

# Tiva™ TM4C129XKCZAD Microcontroller

## DATA SHEET

DS-TM4C129XKCZAD-15863.2743
SPMS448B

Copyright © 2007-2014
Texas Instruments Incorporated

# Copyright

Copyright © 2007-2014 Texas Instruments Incorporated. Tiva and TivaWare are trademarks of Texas Instruments Incorporated. ARM and Thumb are registered trademarks and Cortex is a trademark of ARM Limited. All other trademarks are the property of others.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Texas Instruments Incorporated
108 Wild Basin, Suite 350
Austin, TX 78746
http://www.ti.com/tm4c
http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm

**WARNING – EXPORT NOTICE: Recipient agrees to not knowingly export or re-export, directly or indirectly, any product or technical data (as defined by the U.S., EU, and other Export Administration Regulations) including software, or any controlled product restricted by other applicable national regulations, received from Disclosing party under this Agreement, or any direct product of such technology, to any destination to which such export or re-export is restricted or prohibited by U.S. or other applicable laws, without obtaining prior authorization from U.S. Department of Commerce and other competent Government authorities to the extent required by those laws.**

**According to our best knowledge of the state and end-use of this product or technology, and in compliance with the export control regulations of dual-use goods in force in the origin and exporting countries, this technology is classified as follows:**

- **US ECCN: EAR99**
- **EU ECCN: EAR99**

**And may require export or re-export license for shipping it in compliance with the applicable regulations of certain countries.**

# Table of Contents

**Revision History** ..... **52**

**About This Document** ..... **55**

Audience ..... 55
About This Manual ..... 55
Related Documents ..... 55
Documentation Conventions ..... 56

**1 Architectural Overview** ..... **58**

1.1 Tiva™ C Series Overview ..... 58
1.2 TM4C129XKCZAD Microcontroller Overview ..... 59
1.3 TM4C129XKCZAD Microcontroller Features ..... 62
1.3.1 ARM Cortex-M4F Processor Core ..... 62
1.3.2 On-Chip Memory ..... 64
1.3.3 External Peripheral Interface ..... 66
1.3.4 Cyclical Redundancy Check (CRC) ..... 68
1.3.5 Advanced Encryption Standard (AES) Accelerator ..... 68
1.3.6 Data Encryption Standard (DES) Accelerator ..... 69
1.3.7 Secure Hash Algorithm / Message Digest Algorithm (SHA/MD5) ..... 69
1.3.8 Serial Communications Peripherals ..... 70
1.3.9 System Integration ..... 76
1.3.10 Advanced Motion Control ..... 84
1.3.11 Analog ..... 86
1.3.12 JTAG and ARM Serial Wire Debug ..... 87
1.3.13 Packaging and Temperature ..... 88
1.4 TM4C129XKCZAD Microcontroller Hardware Details ..... 88
1.5 Kits ..... 88
1.6 Support Information ..... 89

**2 The Cortex-M4F Processor** ..... **90**

2.1 Block Diagram ..... 91
2.2 Overview ..... 92
2.2.1 System-Level Interface ..... 92
2.2.2 Integrated Configurable Debug ..... 92
2.2.3 Trace Port Interface Unit (TPIU) ..... 93
2.2.4 Cortex-M4F System Component Details ..... 93
2.3 Programming Model ..... 94
2.3.1 Processor Mode and Privilege Levels for Software Execution ..... 94
2.3.2 Stacks ..... 95
2.3.3 Register Map ..... 95
2.3.4 Register Descriptions ..... 97
2.3.5 Exceptions and Interrupts ..... 113
2.3.6 Data Types ..... 113
2.4 Memory Model ..... 113
2.4.1 Memory Regions, Types and Attributes ..... 117
2.4.2 Memory System Ordering of Memory Accesses ..... 117
2.4.3 Behavior of Memory Accesses ..... 117
2.4.4 Software Ordering of Memory Accesses ..... 118

2.4.5 Bit-Banding ..... 119
2.4.6 Data Storage ..... 121
2.4.7 Synchronization Primitives ..... 122
2.5 Exception Model ..... 123
2.5.1 Exception States ..... 124
2.5.2 Exception Types ..... 124
2.5.3 Exception Handlers ..... 129
2.5.4 Vector Table ..... 129
2.5.5 Exception Priorities ..... 130
2.5.6 Interrupt Priority Grouping ..... 131
2.5.7 Exception Entry and Return ..... 131
2.6 Fault Handling ..... 134
2.6.1 Fault Types ..... 135
2.6.2 Fault Escalation and Hard Faults ..... 135
2.6.3 Fault Status Registers and Fault Address Registers ..... 136
2.6.4 Lockup ..... 136
2.7 Power Management ..... 137
2.7.1 Entering Sleep Modes ..... 137
2.7.2 Wake Up from Sleep Mode ..... 137
2.8 Instruction Set Summary ..... 138
**3 Cortex-M4 Peripherals ..... 145**
3.1 Functional Description ..... 145
3.1.1 System Timer (SysTick) ..... 146
3.1.2 Nested Vectored Interrupt Controller (NVIC) ..... 147
3.1.3 System Control Block (SCB) ..... 148
3.1.4 Memory Protection Unit (MPU) ..... 148
3.1.5 Floating-Point Unit (FPU) ..... 153
3.2 Register Map ..... 157
3.3 System Timer (SysTick) Register Descriptions ..... 160
3.4 NVIC Register Descriptions ..... 164
3.5 System Control Block (SCB) Register Descriptions ..... 174
3.6 Memory Protection Unit (MPU) Register Descriptions ..... 203
3.7 Floating-Point Unit (FPU) Register Descriptions ..... 212
**4 JTAG Interface ..... 218**
4.1 Block Diagram ..... 219
4.2 Signal Description ..... 219
4.3 Functional Description ..... 220
4.3.1 JTAG Interface Pins ..... 220
4.3.2 JTAG TAP Controller ..... 222
4.3.3 Shift Registers ..... 223
4.3.4 Operational Considerations ..... 223
4.4 Initialization and Configuration ..... 226
4.5 Register Descriptions ..... 226
4.5.1 Instruction Register (IR) ..... 227
4.5.2 Data Registers ..... 228
**5 System Control ..... 231**
5.1 Signal Description ..... 231
5.2 Functional Description ..... 231

5.2.1 Device Identification ..... 232
5.2.2 Reset Control ..... 232
5.2.3 Non-Maskable Interrupt ..... 239
5.2.4 Power Control ..... 240
5.2.5 Clock Control ..... 241
5.2.6 System Control ..... 251
5.3 Initialization and Configuration ..... 257
5.4 Register Map ..... 259
5.5 System Control Register Descriptions (System Control Offset) ..... 266
5.6 Cryptographic System Control Register Description (CCM Offset) ..... 553
**6 Processor Support and Exception Module ..... 555**
6.1 Functional Description ..... 555
6.2 Register Map ..... 555
6.3 Register Descriptions ..... 555
**7 Hibernation Module ..... 563**
7.1 Block Diagram ..... 565
7.2 Signal Description ..... 565
7.3 Functional Description ..... 566
7.3.1 Register Access Timing ..... 567
7.3.2 Hibernation Clock Source ..... 567
7.3.3 System Implementation ..... 570
7.3.4 Battery Management ..... 571
7.3.5 Real-Time Clock ..... 571
7.3.6 Tamper ..... 574
7.3.7 Battery-Backed Memory ..... 577
7.3.8 Power Control Using $\overline{\text{HIB}}$ ..... 577
7.3.9 Power Control Using VDD3ON Mode ..... 578
7.3.10 Initiating Hibernate ..... 578
7.3.11 Waking from Hibernate ..... 578
7.3.12 Arbitrary Power Removal ..... 579
7.3.13 Interrupts and Status ..... 580
7.4 Initialization and Configuration ..... 580
7.4.1 Initialization ..... 580
7.4.2 RTC Match Functionality (No Hibernation) ..... 581
7.4.3 RTC Match/Wake-Up from Hibernation ..... 581
7.4.4 External Wake-Up from Hibernation ..... 582
7.4.5 RTC or External Wake-Up from Hibernation ..... 583
7.4.6 Tamper Initialization ..... 583
7.5 Register Map ..... 583
7.6 Register Descriptions ..... 585
**8 Internal Memory ..... 632**
8.1 Block Diagram ..... 632
8.2 Functional Description ..... 634
8.2.1 SRAM ..... 634
8.2.2 ROM ..... 634
8.2.3 Flash Memory ..... 636
8.2.4 EEPROM ..... 647
8.2.5 Bus Matrix Memory Accesses ..... 653

8.3 Register Map ..... 653
8.4 Internal Memory Register Descriptions (Internal Memory Control Offset) ..... 655
8.5 EEPROM Register Descriptions (EEPROM Offset) ..... 681
8.6 Memory Register Descriptions (System Control Offset) ..... 698
**9 Micro Direct Memory Access (µDMA) ..... 708**
9.1 Block Diagram ..... 709
9.2 Functional Description ..... 709
9.2.1 Channel Assignments ..... 710
9.2.2 Priority ..... 711
9.2.3 Arbitration Size ..... 712
9.2.4 Request Types ..... 712
9.2.5 Channel Configuration ..... 713
9.2.6 Transfer Modes ..... 715
9.2.7 Transfer Size and Increment ..... 723
9.2.8 Peripheral Interface ..... 723
9.2.9 Software Request ..... 724
9.2.10 Interrupts and Errors ..... 724
9.3 Initialization and Configuration ..... 724
9.3.1 Module Initialization ..... 724
9.3.2 Configuring a Memory-to-Memory Transfer ..... 725
9.3.3 Configuring a Peripheral for Simple Transmit ..... 726
9.3.4 Configuring a Peripheral for Ping-Pong Receive ..... 728
9.3.5 Configuring Channel Assignments ..... 731
9.4 Register Map ..... 731
9.5 µDMA Channel Control Structure ..... 732
9.6 µDMA Register Descriptions ..... 739
**10 General-Purpose Input/Outputs (GPIOs) ..... 772**
10.1 Signal Description ..... 773
10.2 Pad Capabilities ..... 778
10.3 Functional Description ..... 778
10.3.1 Data Control ..... 780
10.3.2 Interrupt Control ..... 782
10.3.3 Mode Control ..... 783
10.3.4 Commit Control ..... 784
10.3.5 Pad Control ..... 784
10.3.6 Identification ..... 785
10.4 Initialization and Configuration ..... 785
10.5 Register Map ..... 787
10.6 Register Descriptions ..... 790
**11 External Peripheral Interface (EPI) ..... 848**
11.1 EPI Block Diagram ..... 849
11.2 Signal Description ..... 850
11.3 Functional Description ..... 851
11.3.1 Master Access to EPI ..... 852
11.3.2 Non-Blocking Reads ..... 852
11.3.3 DMA Operation ..... 853
11.4 Initialization and Configuration ..... 854
11.4.1 EPI Interface Options ..... 855

11.4.2 SDRAM Mode ..... 855
11.4.3 Host Bus Mode ..... 859
11.4.4 General-Purpose Mode ..... 880
11.5 Register Map ..... 887
11.6 Register Descriptions ..... 889
**12 Cyclical Redundancy Check (CRC) ..... 979**
12.1 Functional Description ..... 979
12.1.1 CRC Support ..... 979
12.2 Initialization and Configuration ..... 981
12.2.1 CRC Initialization and Configuration ..... 981
12.3 Register Map ..... 982
12.4 CRC Module Register Descriptions ..... 982
**13 Advance Encryption Standard Accelerator (AES) ..... 988**
13.1 AES Overview ..... 988
13.2 AES Functional Description ..... 988
13.2.1 AES Block Diagram ..... 989
13.2.2 AES Algorithm ..... 992
13.2.3 AES Operating Modes ..... 993
13.2.4 AES Software Reset ..... 1001
13.2.5 Power Management ..... 1001
13.2.6 Hardware Requests ..... 1001
13.3 AES Performance Information ..... 1002
13.4 AES Module Programming Guide ..... 1004
13.4.1 AES Low - Level Programming Models ..... 1004
13.5 Register Map ..... 1009
13.6 AES Register Descriptions ..... 1011
13.7 AES µDMA Interrupt Register Descriptions (CCM Offset) ..... 1033
**14 Data Encryption Standard Accelerator (DES) ..... 1040**
14.1 DES Functional Description ..... 1040
14.2 DES Block Diagram ..... 1041
14.2.1 µDMA Control ..... 1041
14.2.2 Interrupt Control ..... 1042
14.2.3 Register Interface ..... 1042
14.2.4 DES Engine ..... 1042
14.3 Software Reset ..... 1043
14.4 DES Supported Modes of Operation ..... 1043
14.4.1 ECB Feedback Mode ..... 1043
14.5 DES Module Programming Guide -Low Level Programming Models ..... 1045
14.5.1 Surrounding Modules Global Initialization ..... 1045
14.5.2 Operational Modes Configuration ..... 1046
14.5.3 DES Events Servicing ..... 1048
14.6 Register Map ..... 1049
14.7 DES Register Description ..... 1050
14.8 DES µDMA Interrupt Register Descriptions (CCM Offset) ..... 1064
**15 SHA/MD5 Accelerator ..... 1069**
15.1 SHA/MD5 Functional Description ..... 1069
15.1.1 SHA/MD5 Block Diagram ..... 1069

15.1.2 Power Management ..... 1071
15.1.3 Reset Management ..... 1071
15.1.4 µDMA and Interrupt Requests ..... 1071
15.1.5 Operation Description ..... 1072
15.1.6 SHA/MD5 Performance Information ..... 1078
15.1.7 SHA/MD5 Programming Guide ..... 1079
15.2 SHA/MD5 Register Map ..... 1083
15.3 SHA/MD5 Register Descriptions ..... 1085
15.4 SHA/MD5 µDMA Control Register Descriptions (Encryption Control Offset) ..... 1099
**16 General-Purpose Timers ..... 1104**
16.1 Block Diagram ..... 1105
16.2 Signal Description ..... 1106
16.3 Functional Description ..... 1107
16.3.1 GPTM Reset Conditions ..... 1108
16.3.2 Timer Clock Source ..... 1108
16.3.3 Timer Modes ..... 1109
16.3.4 Wait-for-Trigger Mode ..... 1118
16.3.5 Synchronizing GP Timer Blocks ..... 1119
16.3.6 DMA Operation ..... 1120
16.3.7 ADC Operation ..... 1120
16.3.8 Accessing Concatenated 16/32-Bit GPTM Register Values ..... 1120
16.4 Initialization and Configuration ..... 1121
16.4.1 One-Shot/Periodic Timer Mode ..... 1121
16.4.2 Real-Time Clock (RTC) Mode ..... 1122
16.4.3 Input Edge-Count Mode ..... 1122
16.4.4 Input Edge Time Mode ..... 1123
16.4.5 PWM Mode ..... 1123
16.5 Register Map ..... 1124
16.6 Register Descriptions ..... 1125
**17 Watchdog Timers ..... 1178**
17.1 Block Diagram ..... 1179
17.2 Functional Description ..... 1179
17.2.1 Register Access Timing ..... 1180
17.3 Initialization and Configuration ..... 1180
17.4 Register Map ..... 1180
17.5 Register Descriptions ..... 1181
**18 Analog-to-Digital Converter (ADC) ..... 1203**
18.1 Block Diagram ..... 1204
18.2 Signal Description ..... 1205
18.3 Functional Description ..... 1206
18.3.1 Sample Sequencers ..... 1207
18.3.2 Module Control ..... 1207
18.3.3 Hardware Sample Averaging Circuit ..... 1213
18.3.4 Analog-to-Digital Converter ..... 1213
18.3.5 Differential Sampling ..... 1215
18.3.6 Internal Temperature Sensor ..... 1217
18.3.7 Digital Comparator Unit ..... 1218
18.4 Initialization and Configuration ..... 1223

18.4.1 Module Initialization ..... 1223
18.4.2 Sample Sequencer Configuration ..... 1224
18.5 Register Map ..... 1224
18.6 Register Descriptions ..... 1227
**19 Universal Asynchronous Receivers/Transmitters (UARTs) ..... 1312**
19.1 Block Diagram ..... 1313
19.2 Signal Description ..... 1313
19.3 Functional Description ..... 1315
19.3.1 Transmit/Receive Logic ..... 1316
19.3.2 Baud-Rate Generation ..... 1316
19.3.3 Data Transmission ..... 1317
19.3.4 Serial IR (SIR) ..... 1317
19.3.5 ISO 7816 Support ..... 1319
19.3.6 Modem Handshake Support ..... 1319
19.3.7 9-Bit UART Mode ..... 1320
19.3.8 FIFO Operation ..... 1321
19.3.9 Interrupts ..... 1321
19.3.10 Loopback Operation ..... 1322
19.3.11 DMA Operation ..... 1322
19.4 Initialization and Configuration ..... 1323
19.5 Register Map ..... 1324
19.6 Register Descriptions ..... 1326
**20 Quad Synchronous Serial Interface (QSSI) ..... 1378**
20.1 Block Diagram ..... 1378
20.2 Signal Description ..... 1379
20.3 Functional Description ..... 1381
20.3.1 Bit Rate Generation ..... 1381
20.3.2 FIFO Operation ..... 1381
20.3.3 Advanced, Bi- and Quad- SSI Function ..... 1382
20.3.4 SSInFSS Function ..... 1383
20.3.5 High Speed Clock Operation ..... 1384
20.3.6 Interrupts ..... 1384
20.3.7 Frame Formats ..... 1385
20.3.8 DMA Operation ..... 1392
20.4 Initialization and Configuration ..... 1392
20.4.1 Enhanced Mode Configuration ..... 1394
20.5 Register Map ..... 1395
20.6 Register Descriptions ..... 1396
**21 Inter-Integrated Circuit ($I^2C$) Interface ..... 1427**
21.1 Block Diagram ..... 1428
21.2 Signal Description ..... 1429
21.3 Functional Description ..... 1430
21.3.1 $I^2C$ Bus Functional Overview ..... 1430
21.3.2 Available Speed Modes ..... 1436
21.3.3 Interrupts ..... 1438
21.3.4 Loopback Operation ..... 1439
21.3.5 FIFO and µDMA Operation ..... 1439
21.3.6 Command Sequence Flow Charts ..... 1441

21.4 Initialization and Configuration ..... 1449
21.4.1 Configure the I2C Module to Transmit a Single Byte as a Master ..... 1449
21.4.2 Configure the I2C Master to High Speed Mode ..... 1450
21.5 Register Map ..... 1451
21.6 Register Descriptions (I2C Master) ..... 1453
21.7 Register Descriptions (I2C Slave) ..... 1482
21.8 Register Descriptions (I2C Status and Control) ..... 1499
**22 1-Wire Master Module ..... 1508**
22.1 Block Diagram ..... 1508
22.2 Signal Description ..... 1509
22.3 Functional Description ..... 1509
22.3.1 1-Wire Protocol ..... 1509
22.3.2 Transport Protocol ..... 1512
22.3.3 Overdrive ..... 1513
22.3.4 Timing Override ..... 1513
22.3.5 Command Protocol ..... 1514
22.3.6 Search (Enumeration) and Sub-Byte ..... 1515
22.3.7 Interrupts ..... 1515
22.3.8 DMA ..... 1515
22.3.9 1-Wire Timing ..... 1517
22.4 Initialization and Configuration ..... 1517
22.5 Register Map ..... 1518
22.6 1-Wire Master Register Descriptions ..... 1519
**23 Controller Area Network (CAN) Module ..... 1537**
23.1 Block Diagram ..... 1538
23.2 Signal Description ..... 1538
23.3 Functional Description ..... 1539
23.3.1 Initialization ..... 1540
23.3.2 Operation ..... 1540
23.3.3 Transmitting Message Objects ..... 1541
23.3.4 Configuring a Transmit Message Object ..... 1542
23.3.5 Updating a Transmit Message Object ..... 1543
23.3.6 Accepting Received Message Objects ..... 1543
23.3.7 Receiving a Data Frame ..... 1544
23.3.8 Receiving a Remote Frame ..... 1544
23.3.9 Receive/Transmit Priority ..... 1545
23.3.10 Configuring a Receive Message Object ..... 1545
23.3.11 Handling of Received Message Objects ..... 1546
23.3.12 Handling of Interrupts ..... 1548
23.3.13 Test Mode ..... 1549
23.3.14 Bit Timing Configuration Error Considerations ..... 1551
23.3.15 Bit Time and Bit Rate ..... 1551
23.3.16 Calculating the Bit Timing Parameters ..... 1553
23.4 Register Map ..... 1556
23.5 CAN Register Descriptions ..... 1557
**24 Ethernet Controller ..... 1588**
24.1 Block Diagram ..... 1589

24.2 Signal Description ..... 1589
24.3 Functional Description ..... 1591
24.3.1 Ethernet Clock Control ..... 1591
24.3.2 MII/RMII Interface Signals ..... 1594
24.3.3 DMA Controller ..... 1595
24.3.4 TX/RX Controller ..... 1619
24.3.5 MAC Operation ..... 1623
24.3.6 IEEE 1588 and Advanced Timestamp Function ..... 1625
24.3.7 Frame Filtering ..... 1634
24.3.8 Source Address, VLAN, and CRC Insertion, Replacement or Deletion ..... 1635
24.3.9 Checksum Offload Engine ..... 1637
24.3.10 MAC Management Counters ..... 1638
24.3.11 Power Management Module ..... 1639
24.3.12 Serial Management Interface ..... 1642
24.3.13 Reduced Media Independent Interface (RMII) ..... 1642
24.3.14 Interrupt Configuration ..... 1642
24.4 Ethernet PHY ..... 1642
24.4.1 Integrated PHY Block Diagram ..... 1643
24.4.2 Functional Description ..... 1643
24.4.3 Interface Configuration ..... 1649
24.5 Initialization and Configuration ..... 1650
24.5.1 Ethernet PHY Initialization ..... 1651
24.6 Register Map ..... 1653
24.7 Ethernet MAC Register Descriptions ..... 1657
24.8 Ethernet PHY Register Descriptions ..... 1776
**25 Universal Serial Bus (USB) Controller ..... 1831**
25.1 Block Diagram ..... 1832
25.2 Signal Description ..... 1832
25.3 Register Map ..... 1833
**26 LCD Controller ..... 1840**
26.1 Block Diagram ..... 1840
26.2 Signal Description ..... 1841
26.3 Functional Description ..... 1843
26.3.1 Clocking ..... 1843
26.3.2 LCD DMA Engine ..... 1845
26.3.3 LIDD Bus Operation ..... 1847
26.3.4 Raster Control ..... 1848
26.3.5 LCD Frame Buffer ..... 1851
26.3.6 Palette RAM ..... 1851
26.3.7 Palette ..... 1857
26.3.8 Gray-Scaler/Serializer - Passive (STN) Mode ..... 1857
26.3.9 Gray-Scaler/Serializer - Active (TFT) Mode ..... 1857
26.3.10 Color/Grayscale Intensities and Modulation Rates ..... 1857
26.3.11 Summary of Color Depth ..... 1858
26.3.12 Output Format ..... 1858
26.3.13 Subpicture Feature ..... 1859
26.4 Interrupts ..... 1860
26.5 Bus Transaction Modes ..... 1861

26.6 Initialization and Configuration ..... 1861
26.7 Register Map ..... 1862
26.8 Register Descriptions ..... 1863
**27 Analog Comparators ..... 1909**
27.1 Block Diagram ..... 1910
27.2 Signal Description ..... 1910
27.3 Functional Description ..... 1911
27.3.1 Internal Reference Programming ..... 1912
27.4 Initialization and Configuration ..... 1914
27.5 Register Map ..... 1915
27.6 Register Descriptions ..... 1915
**28 Pulse Width Modulator (PWM) ..... 1925**
28.1 Block Diagram ..... 1926
28.2 Signal Description ..... 1928
28.3 Functional Description ..... 1928
28.3.1 Clock Configuration ..... 1928
28.3.2 PWM Timer ..... 1929
28.3.3 PWM Comparators ..... 1929
28.3.4 PWM Signal Generator ..... 1930
28.3.5 Dead-Band Generator ..... 1931
28.3.6 Interrupt/ADC-Trigger Selector ..... 1931
28.3.7 Synchronization Methods ..... 1932
28.3.8 Fault Conditions ..... 1933
28.3.9 Output Control Block ..... 1934
28.4 Initialization and Configuration ..... 1934
28.5 Register Map ..... 1935
28.6 Register Descriptions ..... 1938
**29 Quadrature Encoder Interface (QEI) ..... 2004**
29.1 Block Diagram ..... 2004
29.2 Signal Description ..... 2006
29.3 Functional Description ..... 2006
29.4 Initialization and Configuration ..... 2009
29.5 Register Map ..... 2009
29.6 Register Descriptions ..... 2010
**30 Pin Diagram ..... 2027**
**31 Signal Tables ..... 2028**
31.1 Signals by Pin Number ..... 2029
31.2 Signals by Signal Name ..... 2048
31.3 Signals by Function, Except for GPIO ..... 2064
31.4 GPIO Pins and Alternate Functions ..... 2081
31.5 Possible Pin Assignments for Alternate Functions ..... 2086
31.6 Connections for Unused Signals ..... 2093
**32 Electrical Characteristics ..... 2095**
32.1 Maximum Ratings ..... 2095
32.2 Operating Characteristics ..... 2096
32.3 Recommended Operating Conditions ..... 2097
32.3.1 DC Operating Conditions ..... 2097

32.3.2 Recommended GPIO Operating Characteristics ..... 2097
32.4 Load Conditions ..... 2100
32.5 JTAG and Boundary Scan ..... 2101
32.6 Power and Brown-Out ..... 2103
32.6.1 $V_{DDA}$ Levels ..... 2103
32.6.2 $V_{DD}$ Levels ..... 2104
32.6.3 $V_{DDC}$ Levels ..... 2105
32.6.4 Response ..... 2106
32.7 Reset ..... 2108
32.8 On-Chip Low Drop-Out (LDO) Regulator ..... 2111
32.9 Clocks ..... 2112
32.9.1 PLL Specifications ..... 2112
32.9.2 PIOSC Specifications ..... 2114
32.9.3 Low-Frequency Internal Oscillator Specifications ..... 2114
32.9.4 Hibernation Clock Source Specifications ..... 2114
32.9.5 Main Oscillator Specifications ..... 2115
32.9.6 System Clock Specification with ADC Operation ..... 2119
32.9.7 System Clock Specification with USB Operation ..... 2119
32.10 Sleep Modes ..... 2120
32.11 Hibernation Module ..... 2122
32.12 Flash Memory ..... 2124
32.13 EEPROM ..... 2125
32.14 Input/Output Pin Characteristics ..... 2126
32.14.1 Types of I/O Pins and ESD Protection ..... 2128
32.15 External Peripheral Interface (EPI) ..... 2130
32.16 Analog-to-Digital Converter (ADC) ..... 2138
32.17 Synchronous Serial Interface (SSI) ..... 2144
32.18 Inter-Integrated Circuit ($I^2C$) Interface ..... 2147
32.19 Ethernet Controller ..... 2148
32.19.1 DC Characteristics ..... 2148
32.19.2 Clock Characteristics ..... 2148
32.19.3 AC Characteristics ..... 2149
32.20 Universal Serial Bus (USB) Controller ..... 2156
32.21 LCD Controller ..... 2158
32.21.1 LCD Interface Display Driver (LIDD Mode) ..... 2158
32.21.2 LCD Raster Mode ..... 2168
32.22 Analog Comparator ..... 2175
32.23 Pulse-Width Modulator (PWM) ..... 2177
32.24 Current Consumption ..... 2178

**A Package Information ..... 2183**
A.1 Orderable Devices ..... 2183
A.2 Device Nomenclature ..... 2183
A.3 Device Markings ..... 2183
A.4 Packaging Diagram ..... 2185

# List of Figures


Figure 1-1. Tiva™ TM4C129XKCZAD Microcontroller High-Level Block Diagram ..... 61
Figure 2-1. CPU Block Diagram ..... 92
Figure 2-2. TPIU Block Diagram ..... 93
Figure 2-3. Cortex-M4F Register Set ..... 96
Figure 2-4. Bit-Band Mapping ..... 121
Figure 2-5. Data Storage ..... 122
Figure 2-6. Vector Table ..... 130
Figure 2-7. Exception Stack Frame ..... 133
Figure 3-1. SRD Use Example ..... 151
Figure 3-2. FPU Register Bank ..... 154
Figure 4-1. JTAG Module Block Diagram ..... 219
Figure 4-2. Test Access Port State Machine ..... 223
Figure 4-3. IDCODE Register Format ..... 229
Figure 4-4. BYPASS Register Format ..... 229
Figure 4-5. Boundary Scan Register Format ..... 229
Figure 5-1. Basic $\overline{\text{RST}}$ Configuration ..... 235
Figure 5-2. External Circuitry to Extend Power-On Reset ..... 235
Figure 5-3. Reset Circuit Controlled by Switch ..... 235
Figure 5-4. Power Architecture ..... 241
Figure 5-5. Main Clock Tree ..... 244
Figure 5-6. Module Clock Selection ..... 253
Figure 7-1. Hibernation Module Block Diagram ..... 565
Figure 7-2. Using a Crystal as the Hibernation Clock Source with a Single Battery Source ..... 569
Figure 7-3. Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode ..... 569
Figure 7-4. Using a Regulator for Both $V_{DD}$ and $V_{BAT}$ ..... 570
Figure 7-5. Counter Behavior with a TRIM Value of 0x8002 ..... 574
Figure 7-6. Counter Behavior with a TRIM Value of 0x7FFC ..... 574
Figure 7-7. Tamper Block Diagram ..... 574
Figure 7-8. Tamper Pad with Glitch Filtering ..... 575
Figure 8-1. Internal Memory Block Diagram ..... 633
Figure 8-2. Flash Memory Configuration ..... 637
Figure 8-3. Single 256-Bit Prefetch Buffer Set ..... 638
Figure 8-4. Four 256-Bit Prefetch Buffer Configuration ..... 638
Figure 8-5. Single Cycle Access, 0 Wait States ..... 639
Figure 8-6. Prefetch Fills from Flash ..... 640
Figure 8-7. Mirror Mode Function ..... 641
Figure 9-1. µDMA Block Diagram ..... 709
Figure 9-2. Example of Ping-Pong µDMA Transaction ..... 716
Figure 9-3. Memory Scatter-Gather, Setup and Configuration ..... 718
Figure 9-4. Memory Scatter-Gather, µDMA Copy Sequence ..... 719
Figure 9-5. Peripheral Scatter-Gather, Setup and Configuration ..... 721
Figure 9-6. Peripheral Scatter-Gather, µDMA Copy Sequence ..... 722
Figure 10-1. Digital I/O Pads ..... 779
Figure 10-2. Analog/Digital I/O Pads ..... 780
Figure 10-3. GPIODATA Write Example ..... 781

Figure 10-4. GPIODATA Read Example ..... 781
Figure 11-1. EPI Block Diagram ..... 850
Figure 11-2. SDRAM Non-Blocking Read Cycle ..... 858
Figure 11-3. SDRAM Normal Read Cycle ..... 858
Figure 11-4. SDRAM Write Cycle ..... 859
Figure 11-5. iRDY Access Stalls, IRDYDLY==01, 10, 11 ..... 870
Figure 11-6. iRDY Signal Connection ..... 870
Figure 11-7. PSRAM Burst Read ..... 872
Figure 11-8. PSRAM Burst Write ..... 873
Figure 11-9. Read Delay During Refresh Event ..... 874
Figure 11-10. Write Delay During Refresh Event ..... 874
Figure 11-11. Example Schematic for Muxed Host-Bus 16 Mode ..... 875
Figure 11-12. Host-Bus Read Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0 ..... 878
Figure 11-13. Host-Bus Write Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0 ..... 878
Figure 11-14. Host-Bus Write Cycle with Multiplexed Address and Data, MODE = 0x0, WRHIGH = 0, RDHIGH = 0 ..... 879
Figure 11-15. Host-Bus Write Cycle with Multiplexed Address and Data and ALE with Dual or Quad CSn ..... 879
Figure 11-16. Continuous Read Mode Accesses ..... 879
Figure 11-17. Write Followed by Read to External FIFO ..... 880
Figure 11-18. Two-Entry FIFO ..... 880
Figure 11-19. Single-Cycle Single Write Access, FRM50=0, FRMCNT=0, WR2CYC=0 ..... 883
Figure 11-20. Two-Cycle Read, Write Accesses, FRM50=0, FRMCNT=0, WR2CYC=1 ..... 884
Figure 11-21. Read Accesses, FRM50=0, FRMCNT=0 ..... 884
Figure 11-22. FRAME Signal Operation, FRM50=0 and FRMCNT=0 ..... 885
Figure 11-23. FRAME Signal Operation, FRM50=0 and FRMCNT=1 ..... 885
Figure 11-24. FRAME Signal Operation, FRM50=0 and FRMCNT=2 ..... 885
Figure 11-25. FRAME Signal Operation, FRM50=1 and FRMCNT=0 ..... 885
Figure 11-26. FRAME Signal Operation, FRM50=1 and FRMCNT=1 ..... 886
Figure 11-27. FRAME Signal Operation, FRM50=1 and FRMCNT=2 ..... 886
Figure 11-28. EPI Clock Operation, CLKGATE=1, WR2CYC=0 ..... 886
Figure 11-29. EPI Clock Operation, CLKGATE=1, WR2CYC=1 ..... 887
Figure 13-1. AES Block Diagram ..... 989
Figure 13-2. AES - ECB Feedback Mode ..... 993
Figure 13-3. AES - CBC Feedback Mode ..... 994
Figure 13-4. AES Encryption With CTR/ICM Mode ..... 994
Figure 13-5. AES - CFB Feedback Mode ..... 995
Figure 13-6. AES - F8 Mode ..... 996
Figure 13-7. AES - XTS Operation ..... 996
Figure 13-8. AES - F9 Operation ..... 997
Figure 13-9. AES - CBC-MAC Authentication Mode ..... 998
Figure 13-10. AES - GCM Operation ..... 999
Figure 13-11. AES - CCM Operation ..... 1000
Figure 13-12. AES Polling Mode ..... 1007
Figure 13-13. AES Interrupt Service ..... 1009
Figure 14-1. DES Block Diagram ..... 1041
Figure 14-2. DES - ECB Feedback Mode ..... 1044
Figure 14-3. DES3DES - CBC Feedback Mode ..... 1044

Figure 14-4. DES3DES-CFB Feedback Mode ..... 1045
Figure 14-5. DES Polling Mode ..... 1047
Figure 14-6. DES Interrupt Service ..... 1048
Figure 14-7. DES Context Input Event Service ..... 1049
Figure 15-1. SHA/MD5 Module Block Diagram ..... 1070
Figure 15-2. SHA/MD5 Polling Mode ..... 1081
Figure 15-3. SHA/MD5 Interrupt Subroutine ..... 1083
Figure 16-1. GPTM Module Block Diagram ..... 1105
Figure 16-2. Input Edge-Count Mode Example, Counting Down ..... 1113
Figure 16-3. 16-Bit Input Edge-Time Mode Example ..... 1115
Figure 16-4. 16-Bit PWM Mode Example ..... 1117
Figure 16-5. CCP Output, GPTMTnMATCHR > GPTMTnILR ..... 1117
Figure 16-6. CCP Output, GPTMTnMATCHR = GPTMTnILR ..... 1118
Figure 16-7. CCP Output, GPTMTnILR > GPTMTnMATCHR ..... 1118
Figure 16-8. Timer Daisy Chain ..... 1119
Figure 17-1. WDT Module Block Diagram ..... 1179
Figure 18-1. Implementation of Two ADC Blocks ..... 1204
Figure 18-2. ADC Module Block Diagram ..... 1205
Figure 18-3. ADC Sample Phases ..... 1210
Figure 18-4. Doubling the ADC Sample Rate ..... 1211
Figure 18-5. Skewed Sampling ..... 1212
Figure 18-6. Sample Averaging Example ..... 1213
Figure 18-7. ADC Input Equivalency ..... 1214
Figure 18-8. ADC Voltage Reference ..... 1214
Figure 18-9. ADC Conversion Result ..... 1215
Figure 18-10. Differential Voltage Representation ..... 1217
Figure 18-11. Internal Temperature Sensor Characteristic ..... 1218
Figure 18-12. Low-Band Operation (CIC=0x0 and/or CTC=0x0) ..... 1221
Figure 18-13. Mid-Band Operation (CIC=0x1 and/or CTC=0x1) ..... 1222
Figure 18-14. High-Band Operation (CIC=0x3 and/or CTC=0x3) ..... 1223
Figure 19-1. UART Module Block Diagram ..... 1313
Figure 19-2. UART Character Frame ..... 1316
Figure 19-3. IrDA Data Modulation ..... 1318
Figure 20-1. QSSI Module with Advanced, Bi-SSI and Quad-SSI Support ..... 1379
Figure 20-2. TI Synchronous Serial Frame Format (Single Transfer) ..... 1386
Figure 20-3. TI Synchronous Serial Frame Format (Continuous Transfer) ..... 1387
Figure 20-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 ..... 1388
Figure 20-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 ..... 1388
Figure 20-6. Freescale SPI Frame Format with SPO=0 and SPH=1 ..... 1389
Figure 20-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 ..... 1390
Figure 20-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 ..... 1390
Figure 20-9. Freescale SPI Frame Format with SPO=1 and SPH=1 ..... 1391
Figure 21-1. $I^2C$ Block Diagram ..... 1428
Figure 21-2. $I^2C$ Bus Configuration ..... 1430
Figure 21-3. START and STOP Conditions ..... 1431
Figure 21-4. Complete Data Transfer with a 7-Bit Address ..... 1432
Figure 21-5. R/S Bit in First Byte ..... 1432
Figure 21-6. Data Validity During Bit Transfer on the $I^2C$ Bus ..... 1432

Figure 21-7. High-Speed Data Format ..... 1438
Figure 21-8. Master Single TRANSMIT ..... 1442
Figure 21-9. Master Single RECEIVE ..... 1443
Figure 21-10. Master TRANSMIT of Multiple Data Bytes ..... 1444
Figure 21-11. Master RECEIVE of Multiple Data Bytes ..... 1445
Figure 21-12. Master RECEIVE with Repeated START after Master TRANSMIT ..... 1446
Figure 21-13. Master TRANSMIT with Repeated START after Master RECEIVE ..... 1447
Figure 21-14. Standard High Speed Mode Master Transmit ..... 1448
Figure 21-15. Slave Command Sequence ..... 1449
Figure 22-1. 1-Wire Block Diagram ..... 1508
Figure 22-2. 1-Wire Reset Protocol ..... 1510
Figure 22-3. 1-Wire Master Transmitting a 1 ..... 1511
Figure 22-4. 1-Wire Master Transmitting a 0' ..... 1511
Figure 22-5. 1-Wire Master Receiving a 1 Signal from a Slave ..... 1512
Figure 22-6. 1-Wire Master Receiving a 0 Signal from a Slave ..... 1512
Figure 23-1. CAN Controller Block Diagram ..... 1538
Figure 23-2. CAN Data/Remote Frame ..... 1539
Figure 23-3. Message Objects in a FIFO Buffer ..... 1548
Figure 23-4. CAN Bit Time ..... 1552
Figure 24-1. Ethernet MAC with Integrated PHY Interface ..... 1589
Figure 24-2. Ethernet MAC and PHY Clock Structure ..... 1592
Figure 24-3. MII Clock Structure ..... 1593
Figure 24-4. RMII Clock Structure ..... 1594
Figure 24-5. Enhanced Transmit Descriptor Structure ..... 1599
Figure 24-6. Enhanced Receive Descriptor Structure ..... 1604
Figure 24-7. TX DMA Default Operation Using Descriptors ..... 1611
Figure 24-8. TX DMA OSF Mode Operation Using Descriptors ..... 1613
Figure 24-9. RX DMA Operation Flow ..... 1616
Figure 24-10. Networked Time Synchronization ..... 1626
Figure 24-11. System Time Update Using Fine Correction Method ..... 1628
Figure 24-12. Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction ..... 1631
Figure 24-13. Wake-Up Frame Filter Register Bank ..... 1639
Figure 24-14. Integrated PHY Diagram ..... 1643
Figure 24-15. Interface to Ethernet Jack ..... 1650
Figure 25-1. USB Module Block Diagram ..... 1832
Figure 26-1. LCD Block Diagram ..... 1841
Figure 26-2. Input and Output Clocks ..... 1844
Figure 26-3. LCD Raster Data Path ..... 1850
Figure 26-4. Palette RAM Structure for 1, 2, and 4 Bits Per Pixel ..... 1852
Figure 26-5. 24 bpp Packed Data Format ..... 1854
Figure 26-6. 24 bpp Unpacked Data Format ..... 1854
Figure 26-7. 24 bpp Color RGB Remapping on LCDDATA[23:0] ..... 1855
Figure 26-8. 16 bpp Data Format ..... 1855
Figure 26-9. 16 bpp Color Component Ordering ..... 1855
Figure 26-10. 12 bpp Data Format ..... 1856
Figure 26-11. 12 bpp Color Component Ordering ..... 1856
Figure 26-12. 1/2/4/8 bpp Dither Output ..... 1857

Figure 26-13. Monchrome and Color Output ..... 1859
Figure 26-14. Example of Subpicture ..... 1860
Figure 26-15. Subpicture Output with HOLS Bit Variations ..... 1860
Figure 27-1. Analog Comparator Module Block Diagram ..... 1910
Figure 27-2. Structure of Comparator Unit ..... 1911
Figure 27-3. Comparator Internal Reference Structure ..... 1912
Figure 28-1. PWM Module Diagram ..... 1927
Figure 28-2. PWM Generator Block Diagram ..... 1927
Figure 28-3. PWM Count-Down Mode ..... 1930
Figure 28-4. PWM Count-Up/Down Mode ..... 1930
Figure 28-5. PWM Generation Example In Count-Up/Down Mode ..... 1931
Figure 28-6. PWM Dead-Band Generator ..... 1931
Figure 29-1. QEI Block Diagram ..... 2005
Figure 29-2. QEI Input Signal Logic ..... 2006
Figure 29-3. Quadrature Encoder and Velocity Predivider Operation ..... 2008
Figure 30-1. 212-Ball BGA Package Pin Diagram (Top View) ..... 2027
Figure 32-1. Load Conditions ..... 2100
Figure 32-2. JTAG Test Clock Input Timing ..... 2102
Figure 32-3. JTAG Test Access Port (TAP) Timing ..... 2102
Figure 32-4. Power and Brown-Out Assertions vs $V_{DDA}$ Levels ..... 2104
Figure 32-5. Power and Brown-Out Assertions vs $V_{DD}$ Levels ..... 2105
Figure 32-6. POK Assertion vs $V_{DDC}$ ..... 2106
Figure 32-7. POR-BOR $V_{DD}$ Glitch Response ..... 2106
Figure 32-8. POR-BOR $V_{DD}$ Droop Response ..... 2107
Figure 32-9. Digital Power-On Reset Timing ..... 2108
Figure 32-10. Brown-Out Reset Timing ..... 2109
Figure 32-11. External Reset Timing ($\overline{RST}$) ..... 2109
Figure 32-12. Software Reset Timing ..... 2109
Figure 32-13. Watchdog Reset Timing ..... 2109
Figure 32-14. MOSC Failure Reset Timing ..... 2110
Figure 32-15. Hibernation Module Timing ..... 2123
Figure 32-16. ESD Protection ..... 2128
Figure 32-17. ESD Protection for Non-Power Pins (Except $\overline{WAKE}$ Signal) ..... 2129
Figure 32-18. SDRAM Initialization and Load Mode Register Timing ..... 2131
Figure 32-19. SDRAM Read Timing ..... 2131
Figure 32-20. SDRAM Write Timing ..... 2132
Figure 32-21. Host-Bus 8/16 Asynchronous Mode Read Timing ..... 2133
Figure 32-22. Host-Bus 8/16 Asynchronous Mode Write Timing ..... 2133
Figure 32-23. Host-Bus 8/16 Mode Asynchronous Muxed Read Timing ..... 2134
Figure 32-24. Host-Bus 8/16 Mode Asynchronous Muxed Write Timing ..... 2134
Figure 32-25. General-Purpose Mode Read and Write Timing ..... 2135
Figure 32-26. PSRAM Single Burst Read ..... 2136
Figure 32-27. PSRAM Single Burst Write ..... 2137
Figure 32-28. ADC External Reference Filtering ..... 2143
Figure 32-29. ADC Input Equivalency ..... 2143
Figure 32-30. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement ..... 2145
Figure 32-31. Master Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1 ..... 2145

Figure 32-32. Slave Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1 ..... 2146
Figure 32-33. I2C Timing ..... 2147
Figure 32-34. MOSC Crystal Characteristics for Ethernet ..... 2148
Figure 32-35. Single-Ended MOSC Characteristics for Ethernet ..... 2149
Figure 32-36. EN0RREF_CLK 50-MHz Oscillator Characteristics ..... 2149
Figure 32-37. Reset Timing ..... 2150
Figure 32-38. Station Management Write and Read Timing ..... 2151
Figure 32-39. 100 Mb/s MII Transmit Timing ..... 2151
Figure 32-40. 100 Mb/s MII Receive Timing ..... 2152
Figure 32-41. 100 Base-TX Transmit Timing ..... 2152
Figure 32-42. 10 Mb/s MII Transmit Timing ..... 2153
Figure 32-43. 10 Mb/s MII Receive Timing ..... 2153
Figure 32-44. 10Base-TX Normal Link Pulse Timing ..... 2154
Figure 32-45. Auto-Negotiation Fast Link Pulse Timing ..... 2154
Figure 32-46. 100Base-TX Signal Detect Timing ..... 2155
Figure 32-47. RMII Transmit Timing ..... 2155
Figure 32-48. RMII Receive Timing ..... 2155
Figure 32-49. ULPI Interface Timing Diagram ..... 2157
Figure 32-50. Command Write in Hitachi Mode ..... 2160
Figure 32-51. Data Write in Hitachi Mode ..... 2161
Figure 32-52. Command Read in Hitachi Mode ..... 2161
Figure 32-53. Data Read in Hitachi Mode ..... 2162
Figure 32-54. Motorola 6800 Graphic Display Mode Write (Synchronous & Asynchronous Operation) ..... 2163
Figure 32-55. Motorola 6800 Graphic Display Mode Read (Synchronous & Asynchronous Operation) ..... 2164
Figure 32-56. Motorola 6800 Graphic Display Mode Status (Synchronous & Asynchronous Operation) ..... 2165
Figure 32-57. Micro-Interface Graphic Display Intel Write ..... 2166
Figure 32-58. Micro-Interface Graphic Display Intel Read ..... 2167
Figure 32-59. Micro-Interface Graphic Display Intel Status ..... 2168
Figure 32-60. LCD Raster-Mode Display Format ..... 2170
Figure 32-61. LCD Raster-Mode Active ..... 2171
Figure 32-62. LCD Raster-Mode Passive ..... 2172
Figure 32-63. LCD Raster-Mode Control Signal Activation ..... 2173
Figure 32-64. LCD Raster-Mode Control Signal Deactivation ..... 2174
Figure A-1. Key to Part Numbers ..... 2183
Figure A-2. TM4C129XKCZAD 212-Ball BGA Package Diagram ..... 2185

# List of Tables


Table 1. Revision History ..... 52
Table 2. Documentation Conventions ..... 56
Table 1-1. TM4C129XKCZAD Microcontroller Features ..... 59
Table 2-1. Summary of Processor Mode, Privilege Level, and Stack Use ..... 95
Table 2-2. Processor Register Map ..... 96
Table 2-3. PSR Register Combinations ..... 102
Table 2-4. Memory Map ..... 113
Table 2-5. Memory Access Behavior ..... 117
Table 2-6. SRAM Memory Bit-Banding Regions ..... 119
Table 2-7. Peripheral Memory Bit-Banding Regions ..... 120
Table 2-8. Exception Types ..... 125
Table 2-9. Interrupts ..... 126
Table 2-10. Exception Return Behavior ..... 134
Table 2-11. Faults ..... 135
Table 2-12. Fault Status and Fault Address Registers ..... 136
Table 2-13. Cortex-M4F Instruction Summary ..... 138
Table 3-1. Core Peripheral Register Regions ..... 145
Table 3-2. Memory Attributes Summary ..... 149
Table 3-3. TEX, S, C, and B Bit Field Encoding ..... 151
Table 3-4. Cache Policy for Memory Attribute Encoding ..... 152
Table 3-5. AP Bit Field Encoding ..... 152
Table 3-6. Memory Region Attributes for Tiva™ C Series Microcontrollers ..... 153
Table 3-7. QNaN and SNaN Handling ..... 156
Table 3-8. Peripherals Register Map ..... 157
Table 3-9. Interrupt Priority Levels ..... 182
Table 3-10. Example SIZE Field Values ..... 210
Table 4-1. JTAG_SWD_SWO Signals (212BGA) ..... 219
Table 4-2. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion ..... 221
Table 4-3. JTAG Instruction Register Commands ..... 227
Table 5-1. System Control & Clocks Signals (212BGA) ..... 231
Table 5-2. Reset Sources ..... 232
Table 5-3. Clock Source Options ..... 242
Table 5-4. Clock Source State Following POR ..... 243
Table 5-5. System Clock Frequency ..... 247
Table 5-6. System Divisor Factors for $f_{vco}$=480 MHz ..... 249
Table 5-7. Actual PLL Frequency ..... 250
Table 5-8. Peripheral Memory Power Control ..... 255
Table 5-9. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage ..... 255
Table 5-10. MOSC Configurations ..... 258
Table 5-11. System Control Register Map ..... 259
Table 5-12. MEMTIM0 Register Configuration versus Frequency ..... 289
Table 5-13. MOSC Configurations ..... 293
Table 5-14. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage ..... 312
Table 5-15. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage ..... 315
Table 5-16. Module Power Control ..... 474
Table 5-17. Module Power Control ..... 476

Table 5-18. Module Power Control ..... 479
Table 5-19. Module Power Control ..... 485
Table 5-20. Module Power Control ..... 487
Table 5-21. Module Power Control ..... 489
Table 5-22. Module Power Control ..... 491
Table 5-23. Module Power Control ..... 494
Table 5-24. Module Power Control ..... 496
Table 5-25. Module Power Control ..... 500
Table 5-26. Module Power Control ..... 502
Table 5-27. Module Power Control ..... 504
Table 5-28. Module Power Control ..... 506
Table 5-29. Module Power Control ..... 508
Table 5-30. Module Power Control ..... 510
Table 5-31. Module Power Control ..... 512
Table 5-32. Module Power Control ..... 514
Table 5-33. Module Power Control ..... 516
Table 5-34. Module Power Control ..... 518
Table 5-35. Module Power Control ..... 520
Table 5-36. Module Power Control ..... 522
Table 6-1. System Exception Register Map ..... 555
Table 7-1. Hibernate Signals (212BGA) ..... 566
Table 7-2. HIB Clock Source Configurations ..... 568
Table 7-3. Hibernation Module Register Map ..... 584
Table 8-1. MEMTIM0 Register Configuration versus Frequency ..... 637
Table 8-2. Flash Memory Protection Policy Combinations ..... 642
Table 8-3. User-Programmable Flash Memory Resident Registers ..... 646
Table 8-4. MEMTIM0 Register Configuration versus Frequency ..... 648
Table 8-5. Master Memory Access Availability ..... 653
Table 8-6. Flash Register Map ..... 653
Table 9-1. µDMA Channel Assignments ..... 710
Table 9-2. Request Type Support ..... 712
Table 9-3. Control Structure Memory Map ..... 714
Table 9-4. Channel Control Structure ..... 714
Table 9-5. µDMA Read Example: 8-Bit Peripheral ..... 723
Table 9-6. µDMA Interrupt Assignments ..... 724
Table 9-7. Channel Control Structure Offsets for Channel 30 ..... 725
Table 9-8. Channel Control Word Configuration for Memory Transfer Example ..... 726
Table 9-9. Channel Control Structure Offsets for Channel 7 ..... 727
Table 9-10. Channel Control Word Configuration for Peripheral Transmit Example ..... 727
Table 9-11. Primary and Alternate Channel Control Structure Offsets for Channel 8 ..... 729
Table 9-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example ..... 729
Table 9-13. µDMA Register Map ..... 731
Table 10-1. GPIO Pins With Special Considerations ..... 773
Table 10-2. GPIO Pins and Alternate Functions (212BGA) ..... 773
Table 10-3. GPIO Drive Strength Options ..... 785
Table 10-4. GPIO Pad Configuration Examples ..... 786
Table 10-5. GPIO Interrupt Configuration Example ..... 787

Table 10-6. GPIO Pins With Special Considerations ..... 788
Table 10-7. GPIO Register Map ..... 789
Table 10-8. GPIO Pins With Special Considerations ..... 803
Table 10-9. GPIO Pins With Special Considerations ..... 809
Table 10-10. GPIO Pins With Special Considerations ..... 811
Table 10-11. GPIO Pins With Special Considerations ..... 814
Table 10-12. GPIO Pins With Special Considerations ..... 820
Table 10-13. GPIO Drive Strength Options ..... 833
Table 11-1. External Peripheral Interface Signals (212BGA) ..... 850
Table 11-2. EPI Interface Options ..... 855
Table 11-3. EPI SDRAM x16 Signal Connections ..... 856
Table 11-4. CSCFGEXT + CSCFG Encodings ..... 860
Table 11-5. Dual- and Quad- Chip Select Address Mappings ..... 861
Table 11-6. Chip Select Configuration Register Assignment ..... 862
Table 11-7. Capabilities of Host Bus 8 and Host Bus 16 Modes ..... 862
Table 11-8. EPI Host-Bus 8 Signal Connections ..... 864
Table 11-9. EPI Host-Bus 16 Signal Connections ..... 866
Table 11-10. PSRAM Fixed Latency Wait State Configuration ..... 872
Table 11-11. Data Phase Wait State Programming ..... 876
Table 11-12. EPI General-Purpose Signal Connections ..... 882
Table 11-13. External Peripheral Interface (EPI) Register Map ..... 887
Table 11-14. CSCFGEXT + CSCFG Encodings ..... 913
Table 11-15. CSCFGEXT + CSCFG Encodings ..... 919
Table 12-1. Endian Configuration ..... 980
Table 12-2. Endian Configuration with Bit Reversal ..... 980
Table 12-3. CCM Register Map ..... 982
Table 13-1. Key-Block-Round Combinations ..... 992
Table 13-2. Interrupts and Events ..... 1001
Table 13-3. AES Module Performance (Input/Output Block Size = 128) ..... 1002
Table 13-4. AES Module Packet Mode Switch Overhead ..... 1003
Table 13-5. AES Register Map ..... 1009
Table 13-6. AES Key Register Descriptions ..... 1012
Table 14-1. Key Repartition ..... 1041
Table 14-2. DES Reset Description ..... 1043
Table 14-3. DES Global Initialization ..... 1045
Table 14-4. DES Algorithm Type Configuration ..... 1046
Table 14-5. 3DES Algorithm Type Configuration ..... 1046
Table 14-6. DES Interrupt Mode ..... 1047
Table 14-7. DES DMA Mode ..... 1047
Table 14-8. DES Register Map ..... 1049
Table 14-9. DES Key Register Mapping ..... 1051
Table 15-1. Interrupts and Events ..... 1072
Table 15-2. SHA/MD5 Module Algorithm Selection ..... 1072
Table 15-3. Outer Digest Registers ..... 1073
Table 15-4. Inner Digest Registers ..... 1074
Table 15-5. SHA Digest Processed in Three Passes ..... 1076
Table 15-6. SHA Digest Processed in One Pass ..... 1076
Table 15-7. SHA/MD5 Performance ..... 1078

Table 15-8. Continuing a Prior HMAC ..... 1080
Table 15-9. SHA-1 Apply on the Key ..... 1081
Table 15-10. Interrupt Mode ..... 1081
Table 15-11. DMA Mode ..... 1082
Table 15-12. SHA/MD5 Register Map ..... 1083
Table 15-13. SHA/MD5 Inner/Outer Digest/HMAC Key Register Mapping ..... 1085
Table 16-1. Available CCP Pins ..... 1105
Table 16-2. General-Purpose Timers Signals (212BGA) ..... 1106
Table 16-3. General-Purpose Timer Capabilities ..... 1108
Table 16-4. Counter Values When the Timer is Enabled in Periodic or One-Shot Modes ..... 1109
Table 16-5. 16-Bit Timer With Prescaler Configurations ..... 1111
Table 16-6. Counter Values When the Timer is Enabled in RTC Mode ..... 1111
Table 16-7. Counter Values When the Timer is Enabled in Input Edge-Count Mode ..... 1112
Table 16-8. Counter Values When the Timer is Enabled in Input Event-Count Mode ..... 1114
Table 16-9. Counter Values When the Timer is Enabled in PWM Mode ..... 1115
Table 16-10. Timeout Actions for GPTM Modes ..... 1119
Table 16-11. Timers Register Map ..... 1124
Table 17-1. Watchdog Timers Register Map ..... 1181
Table 18-1. ADC Signals (212BGA) ..... 1205
Table 18-2. Samples and FIFO Depth of Sequencers ..... 1207
Table 18-3. Sample and Hold Width in ADC Clocks ..... 1209
Table 18-4. $R_S$ and $F_{CONV}$ Values with Varying $N_{SH}$ Values and $F_{ADC}$ = 16 MHz ..... 1210
Table 18-5. $R_S$ and $F_{CONV}$ Values with Varying $N_{SH}$ Values and $F_{ADC}$ = 32 MHz ..... 1210
Table 18-6. Differential Sampling Pairs ..... 1216
Table 18-7. ADC Register Map ..... 1224
Table 18-8. Sample and Hold Width in ADC Clocks ..... 1278
Table 18-9. Sample and Hold Width in ADC Clocks ..... 1290
Table 18-10. Sample and Hold Width in ADC Clocks ..... 1298
Table 19-1. UART Signals (212BGA) ..... 1314
Table 19-2. Flow Control Mode ..... 1320
Table 19-3. UART Register Map ..... 1325
Table 20-1. SSI Signals (212BGA) ..... 1380
Table 20-2. QSSI Transaction Encodings ..... 1383
Table 20-3. SSInFss Functionality ..... 1384
Table 20-4. Legacy Mode TI, Freescale SPI Frame Format Features ..... 1386
Table 20-5. SSI Register Map ..... 1395
Table 21-1. I2C Signals (212BGA) ..... 1429
Table 21-2. Examples of $I^2C$ Master Timer Period Versus Speed Mode ..... 1436
Table 21-3. Examples of $I^2C$ Master Timer Period in High-Speed Mode ..... 1437
Table 21-4. Inter-Integrated Circuit ($I^2C$) Interface Register Map ..... 1452
Table 21-5. Write Field Decoding for I2CMCS[6:0] ..... 1460
Table 22-1. 1-Wire Signals (212BGA) ..... 1509
Table 22-2. Active Time Periods in Overdrive Mode ..... 1513
Table 22-3. Bit Field Definitions for 1-Wire Timing and Override (ONEWIRETIM) Register ..... 1513
Table 22-4. OWIRE Register Map ..... 1518
Table 23-1. Controller Area Network Signals (212BGA) ..... 1539
Table 23-2. Message Object Configurations ..... 1544
Table 23-3. CAN Protocol Ranges ..... 1552

Table 23-4. CANBIT Register Values ..... 1552
Table 23-5. CAN Register Map ..... 1556
Table 24-1. Ethernet Signals (212BGA) ..... 1590
Table 24-2. MII and RMII Interface Signals ..... 1594
Table 24-3. Enhanced Transmit Descriptor 0 (TDES0) ..... 1599
Table 24-4. Enhanced Transmit Descriptor 1 (TDES1) ..... 1602
Table 24-5. Enhanced Transmit Descriptor 2 (TDES2) ..... 1603
Table 24-6. Enhanced Transmit Descriptor 3 (TDES3) ..... 1603
Table 24-7. Enhanced Transmit Descriptor 6 (TDES6) ..... 1603
Table 24-8. Enhanced Transmit Descriptor 7 (TDES7) ..... 1603
Table 24-9. Enhanced Receive Descriptor 0 (RDES0) ..... 1604
Table 24-10. RDES0 Checksum Offload bits ..... 1606
Table 24-11. Enhanced Receive Descriptor 1 (RDES1) ..... 1607
Table 24-12. Enhanced Receive Descriptor 2 (RDES2) ..... 1607
Table 24-13. Enhanced Receive Descriptor 3 (RDES3) ..... 1607
Table 24-14. Enhanced Received Descriptor 4 (RDES4) ..... 1608
Table 24-15. Enhanced Receive Descriptor 6 (RDES6) ..... 1609
Table 24-16. Enhanced Receive Descriptor 7 (RDES7) ..... 1609
Table 24-17. TX MAC Flow Control ..... 1622
Table 24-18. RX MAC Flow Control ..... 1622
Table 24-19. VLAN Match Status ..... 1635
Table 24-20. CRC Replacement Based on Bit 27 and Bit 24 of TDES0 ..... 1637
Table 24-21. Forced Mode Configurations ..... 1644
Table 24-22. Advertised Mode Configurations ..... 1644
Table 24-23. EMACPC to PHY Register Mapping ..... 1652
Table 24-24. Ethernet Register Map ..... 1653
Table 24-25. PPSCTRL Bit Field Values ..... 1736
Table 25-1. USB Signals (212BGA) ..... 1833
Table 25-2. List of Registers ..... 1834
Table 26-1. LCD Signals (212BGA) ..... 1841
Table 26-2. LCD External Signal Details ..... 1842
Table 26-3. LCD Alternate Signal Functions in LIDD Mode ..... 1843
Table 26-4. LIDD I/O Name Map ..... 1848
Table 26-5. Operation Modes Supported by Raster Controller ..... 1849
Table 26-6. Type Encoding in Palette Entry 0 Buffer ..... 1852
Table 26-7. Intensities and Modulation Rates ..... 1857
Table 26-8. Number of Colors/Shades of Gray Available on Screen ..... 1858
Table 26-9. LCD Register Map ..... 1862
Table 26-10. SYSCLK to Pixel Clock (LCDCP) Frequency Conversion Table ..... 1866
Table 26-11. LCD Alternate Signal Functions in LIDD Mode ..... 1869
Table 27-1. Analog Comparators Signals (212BGA) ..... 1910
Table 27-2. Internal Reference Voltage and ACREFCTL Field Values ..... 1912
Table 27-3. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 0 ..... 1913
Table 27-4. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 1 ..... 1914
Table 27-5. Analog Comparators Register Map ..... 1915
Table 28-1. PWM Signals (212BGA) ..... 1928

Table 28-2. PWM Register Map ..... 1935
Table 29-1. QEI Signals (212BGA) ..... 2006
Table 29-2. QEI Register Map ..... 2010
Table 31-1. GPIO Pins With Special Considerations ..... 2028
Table 31-2. Signals by Pin Number ..... 2029
Table 31-3. Signals by Signal Name ..... 2048
Table 31-4. Signals by Function, Except for GPIO ..... 2064
Table 31-5. GPIO Pins and Alternate Functions ..... 2081
Table 31-6. Possible Pin Assignments for Alternate Functions ..... 2086
Table 31-7. Connections for Unused Signals (212-Ball BGA) ..... 2093
Table 32-1. Absolute Maximum Ratings ..... 2095
Table 32-2. ESD Absolute Maximum Ratings ..... 2095
Table 32-3. Temperature Characteristics ..... 2096
Table 32-4. 212 BGA Power Dissipation ..... 2096
Table 32-5. Thermal Characteristics ..... 2096
Table 32-6. Recommended DC Operating Conditions ..... 2097
Table 32-7. Recommended FAST GPIO Pad Operating Conditions ..... 2097
Table 32-8. Recommended Slow GPIO Pad Operating Conditions ..... 2098
Table 32-9. GPIO Current Restrictions ..... 2098
Table 32-10. Maximum GPIO Package Side Assignments ..... 2099
Table 32-11. Load Conditions ..... 2100
Table 32-12. JTAG Characteristics ..... 2101
Table 32-13. Power and Brown-Out Levels ..... 2103
Table 32-14. Reset Characteristics ..... 2108
Table 32-15. LDO Regulator Characteristics ..... 2111
Table 32-16. Phase Locked Loop (PLL) Characteristics ..... 2112
Table 32-17. System Divisor Factors for $f_{vco}$=480 MHz ..... 2113
Table 32-18. Actual PLL Frequency ..... 2113
Table 32-19. PIOSC Clock Characteristics ..... 2114
Table 32-20. Low-Frequency Oscillator Characteristics ..... 2114
Table 32-21. Hibernation Internal Low Frequency Oscillator Clock Characteristics ..... 2114
Table 32-22. Hibernation External Oscillator (XOSC) Input Characteristics ..... 2114
Table 32-23. Main Oscillator Input Characteristics ..... 2115
Table 32-24. Crystal Parameters ..... 2117
Table 32-25. System Clock Characteristics with ADC Operation ..... 2119
Table 32-26. System Clock Characteristics with USB Operation ..... 2119
Table 32-27. Wake from Sleep Characteristics ..... 2120
Table 32-28. Wake from Deep Sleep Characteristics ..... 2120
Table 32-29. Hibernation Module Battery Characteristics ..... 2122
Table 32-30. Hibernation Module Characteristics ..... 2122
Table 32-31. Hibernation Module Tamper I/O Characteristics ..... 2122
Table 32-32. Flash Memory Characteristics ..... 2124
Table 32-33. EEPROM Characteristics ..... 2125
Table 32-34. Fast GPIO Module Characteristics ..... 2126
Table 32-35. Slow GPIO Module Characteristics ..... 2127
Table 32-36. Pad Voltage/Current Characteristics for Hibernate $\overline{\text{WAKE}}$ Pin ..... 2128
Table 32-37. Non-Power I/O Pad Voltage/Current Characteristics ..... 2129
Table 32-38. EPI Interface Load Conditions ..... 2130

Table 32-39. EPI SDRAM Characteristics ..... 2130
Table 32-40. EPI SDRAM Interface Characteristics ..... 2130
Table 32-41. EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics ..... 2132
Table 32-42. EPI General-Purpose Interface Characteristics ..... 2134
Table 32-43. EPI PSRAM Interface Characteristics ..... 2135
Table 32-44. ADC Electrical Characteristics for ADC at 1 Msps ..... 2138
Table 32-45. ADC Electrical Characteristics for ADC at 2 Msps ..... 2140
Table 32-46. SSI Characteristics ..... 2144
Table 32-47. Bi- and Quad-SSI Characteristics ..... 2146
Table 32-48. $I^2C$ Characteristics ..... 2147
Table 32-49. Ethernet PHY DC Characteristics ..... 2148
Table 32-50. MOSC 25-MHz Crystal Specification ..... 2148
Table 32-51. MOSC Single-Ended 25-MHz Oscillator Specification[a] ..... 2148
Table 32-52. EN0RREF_CLK 50-MHz Oscillator Specification ..... 2149
Table 32-53. Ethernet Controller Enable and Software Reset Timing ..... 2149
Table 32-54. MII Serial Management Timing ..... 2150
Table 32-55. 100 Mb/s MII Transmit Timing ..... 2151
Table 32-56. 100 Mb/s MII Receive Timing ..... 2151
Table 32-57. 100Base-TX Transmit Timing ($t_{R/F}$ and Jitter) ..... 2152
Table 32-58. 10 Mb/s MII Transmit Timing ..... 2153
Table 32-59. 10 Mb/s MII Receive Timing ..... 2153
Table 32-60. 10Base-T Normal Link Pulse Timing ..... 2153
Table 32-61. Auto-Negotiation Fast Link Pulse (FLP) Timing ..... 2154
Table 32-62. 100Base-TX Signal Detect Timing ..... 2154
Table 32-63. RMII Transmit Timing ..... 2155
Table 32-64. RMII Receive Timing ..... 2155
Table 32-65. ULPI Interface Timing ..... 2156
Table 32-66. LCD Controller Load Capacitance Limits ..... 2158
Table 32-67. LCD Switching Characteristics ..... 2159
Table 32-68. Timing Requirements for LCDDATA in LIDD Mode ..... 2159
Table 32-69. Switching Characteristics Over Recommended Operating Characteristics for LCD Raster Mode ..... 2168
Table 32-70. Analog Comparator Characteristics ..... 2175
Table 32-71. Analog Comparator Voltage Reference Characteristics ..... 2175
Table 32-72. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 0 ..... 2175
Table 32-73. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 1 ..... 2176
Table 32-74. PWM Timing Characteristics ..... 2177
Table 32-75. Current Consumption ..... 2178
Table 32-76. Peripheral Current Consumption ..... 2182

# List of Registers

**The Cortex-M4F Processor ..... 90**
Register 1: Cortex General-Purpose Register 0 (R0) ..... 98
Register 2: Cortex General-Purpose Register 1 (R1) ..... 98
Register 3: Cortex General-Purpose Register 2 (R2) ..... 98
Register 4: Cortex General-Purpose Register 3 (R3) ..... 98
Register 5: Cortex General-Purpose Register 4 (R4) ..... 98
Register 6: Cortex General-Purpose Register 5 (R5) ..... 98
Register 7: Cortex General-Purpose Register 6 (R6) ..... 98
Register 8: Cortex General-Purpose Register 7 (R7) ..... 98
Register 9: Cortex General-Purpose Register 8 (R8) ..... 98
Register 10: Cortex General-Purpose Register 9 (R9) ..... 98
Register 11: Cortex General-Purpose Register 10 (R10) ..... 98
Register 12: Cortex General-Purpose Register 11 (R11) ..... 98
Register 13: Cortex General-Purpose Register 12 (R12) ..... 98
Register 14: Stack Pointer (SP) ..... 99
Register 15: Link Register (LR) ..... 100
Register 16: Program Counter (PC) ..... 101
Register 17: Program Status Register (PSR) ..... 102
Register 18: Priority Mask Register (PRIMASK) ..... 106
Register 19: Fault Mask Register (FAULTMASK) ..... 107
Register 20: Base Priority Mask Register (BASEPRI) ..... 108
Register 21: Control Register (CONTROL) ..... 109
Register 22: Floating-Point Status Control (FPSC) ..... 111
**Cortex-M4 Peripherals ..... 145**
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010 ..... 161
Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014 ..... 163
Register 3: SysTick Current Value Register (STCURRENT), offset 0x018 ..... 164
Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100 ..... 165
Register 5: Interrupt 32-63 Set Enable (EN1), offset 0x104 ..... 165
Register 6: Interrupt 64-95 Set Enable (EN2), offset 0x108 ..... 165
Register 7: Interrupt 96-113 Set Enable (EN3), offset 0x10C ..... 165
Register 8: Interrupt 0-31 Clear Enable (DIS0), offset 0x180 ..... 166
Register 9: Interrupt 32-63 Clear Enable (DIS1), offset 0x184 ..... 166
Register 10: Interrupt 64-95 Clear Enable (DIS2), offset 0x188 ..... 166
Register 11: Interrupt 96-113 Clear Enable (DIS3), offset 0x18C ..... 166
Register 12: Interrupt 0-31 Set Pending (PEND0), offset 0x200 ..... 167
Register 13: Interrupt 32-63 Set Pending (PEND1), offset 0x204 ..... 167
Register 14: Interrupt 64-95 Set Pending (PEND2), offset 0x208 ..... 167
Register 15: Interrupt 96-113 Set Pending (PEND3), offset 0x20C ..... 167
Register 16: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280 ..... 168
Register 17: Interrupt 32-63 Clear Pending (UNPEND1), offset 0x284 ..... 168
Register 18: Interrupt 64-95 Clear Pending (UNPEND2), offset 0x288 ..... 168
Register 19: Interrupt 96-113 Clear Pending (UNPEND3), offset 0x28C ..... 168
Register 20: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300 ..... 169
Register 21: Interrupt 32-63 Active Bit (ACTIVE1), offset 0x304 ..... 169

Register 22: Interrupt 64-95 Active Bit (ACTIVE2), offset 0x308 ..... 169
Register 23: Interrupt 96-127 Active Bit (ACTIVE3), offset 0x30C ..... 169
Register 24: Interrupt 0-3 Priority (PRI0), offset 0x400 ..... 170
Register 25: Interrupt 4-7 Priority (PRI1), offset 0x404 ..... 170
Register 26: Interrupt 8-11 Priority (PRI2), offset 0x408 ..... 170
Register 27: Interrupt 12-15 Priority (PRI3), offset 0x40C ..... 170
Register 28: Interrupt 16-19 Priority (PRI4), offset 0x410 ..... 170
Register 29: Interrupt 20-23 Priority (PRI5), offset 0x414 ..... 170
Register 30: Interrupt 24-27 Priority (PRI6), offset 0x418 ..... 170
Register 31: Interrupt 28-31 Priority (PRI7), offset 0x41C ..... 170
Register 32: Interrupt 32-35 Priority (PRI8), offset 0x420 ..... 170
Register 33: Interrupt 36-39 Priority (PRI9), offset 0x424 ..... 170
Register 34: Interrupt 40-43 Priority (PRI10), offset 0x428 ..... 170
Register 35: Interrupt 44-47 Priority (PRI11), offset 0x42C ..... 170
Register 36: Interrupt 48-51 Priority (PRI12), offset 0x430 ..... 170
Register 37: Interrupt 52-55 Priority (PRI13), offset 0x434 ..... 170
Register 38: Interrupt 56-59 Priority (PRI14), offset 0x438 ..... 170
Register 39: Interrupt 60-63 Priority (PRI15), offset 0x43C ..... 170
Register 40: Interrupt 64-67 Priority (PRI16), offset 0x440 ..... 172
Register 41: Interrupt 68-71 Priority (PRI17), offset 0x444 ..... 172
Register 42: Interrupt 72-75 Priority (PRI18), offset 0x448 ..... 172
Register 43: Interrupt 76-79 Priority (PRI19), offset 0x44C ..... 172
Register 44: Interrupt 80-83 Priority (PRI20), offset 0x450 ..... 172
Register 45: Interrupt 84-87 Priority (PRI21), offset 0x454 ..... 172
Register 46: Interrupt 88-91 Priority (PRI22), offset 0x458 ..... 172
Register 47: Interrupt 92-95 Priority (PRI23), offset 0x45C ..... 172
Register 48: Interrupt 96-99 Priority (PRI24), offset 0x460 ..... 172
Register 49: Interrupt 100-103 Priority (PRI25), offset 0x464 ..... 172
Register 50: Interrupt 104-107 Priority (PRI26), offset 0x468 ..... 172
Register 51: Interrupt 108-111 Priority (PRI27), offset 0x46C ..... 172
Register 52: Interrupt 112-113 Priority (PRI28), offset 0x470 ..... 172
Register 53: Software Trigger Interrupt (SWTRIG), offset 0xF00 ..... 174
Register 54: Auxiliary Control (ACTLR), offset 0x008 ..... 175
Register 55: CPU ID Base (CPUID), offset 0xD00 ..... 177
Register 56: Interrupt Control and State (INTCTRL), offset 0xD04 ..... 178
Register 57: Vector Table Offset (VTABLE), offset 0xD08 ..... 181
Register 58: Application Interrupt and Reset Control (APINT), offset 0xD0C ..... 182
Register 59: System Control (SYSCTRL), offset 0xD10 ..... 184
Register 60: Configuration and Control (CFGCTRL), offset 0xD14 ..... 186
Register 61: System Handler Priority 1 (SYSPRI1), offset 0xD18 ..... 188
Register 62: System Handler Priority 2 (SYSPRI2), offset 0xD1C ..... 189
Register 63: System Handler Priority 3 (SYSPRI3), offset 0xD20 ..... 190
Register 64: System Handler Control and State (SYSHNDCTRL), offset 0xD24 ..... 191
Register 65: Configurable Fault Status (FAULTSTAT), offset 0xD28 ..... 195
Register 66: Hard Fault Status (HFAULTSTAT), offset 0xD2C ..... 201
Register 67: Memory Management Fault Address (MMADDR), offset 0xD34 ..... 202
Register 68: Bus Fault Address (FAULTADDR), offset 0xD38 ..... 203
Register 69: MPU Type (MPUTYPE), offset 0xD90 ..... 204

Register 70: MPU Control (MPUCTRL), offset 0xD94 ..... 205
Register 71: MPU Region Number (MPUNUMBER), offset 0xD98 ..... 207
Register 72: MPU Region Base Address (MPUBASE), offset 0xD9C ..... 208
Register 73: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4 ..... 208
Register 74: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC ..... 208
Register 75: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4 ..... 208
Register 76: MPU Region Attribute and Size (MPUATTR), offset 0xDA0 ..... 210
Register 77: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8 ..... 210
Register 78: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0 ..... 210
Register 79: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8 ..... 210
Register 80: Coprocessor Access Control (CPAC), offset 0xD88 ..... 213
Register 81: Floating-Point Context Control (FPCC), offset 0xF34 ..... 214
Register 82: Floating-Point Context Address (FPCA), offset 0xF38 ..... 216
Register 83: Floating-Point Default Status Control (FPDSC), offset 0xF3C ..... 217

**System Control ..... 231**

Register 1: Device Identification 0 (DID0), offset 0x000 ..... 267
Register 2: Device Identification 1 (DID1), offset 0x004 ..... 269
Register 3: Power-Temp Brown Out Control (PTBOCTL), offset 0x038 ..... 271
Register 4: Raw Interrupt Status (RIS), offset 0x050 ..... 273
Register 5: Interrupt Mask Control (IMC), offset 0x054 ..... 275
Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058 ..... 277
Register 7: Reset Cause (RESC), offset 0x05C ..... 279
Register 8: Power-Temperature Cause (PWRTC), offset 0x060 ..... 282
Register 9: NMI Cause Register (NMIC), offset 0x064 ..... 283
Register 10: Main Oscillator Control (MOSCCTL), offset 0x07C ..... 285
Register 11: Run and Sleep Mode Configuration Register (RSCLKCFG), offset 0x0B0 ..... 287
Register 12: Memory Timing Parameter Register 0 for Main Flash and EEPROM (MEMTIM0), offset 0x0C0 ..... 289
Register 13: Alternate Clock Configuration (ALTCLKCFG), offset 0x138 ..... 292
Register 14: Deep Sleep Clock Configuration Register (DSCLKCFG), offset 0x144 ..... 293
Register 15: Divisor and Source Clock Configuration (DIVSCLK), offset 0x148 ..... 296
Register 16: System Properties (SYSPROP), offset 0x14C ..... 298
Register 17: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150 ..... 301
Register 18: Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154 ..... 303
Register 19: PLL Frequency 0 (PLLFREQ0), offset 0x160 ..... 304
Register 20: PLL Frequency 1 (PLLFREQ1), offset 0x164 ..... 305
Register 21: PLL Status (PLLSTAT), offset 0x168 ..... 306
Register 22: Sleep Power Configuration (SLPPWRCFG), offset 0x188 ..... 307
Register 23: Deep-Sleep Power Configuration (DSLPPWRCFG), offset 0x18C ..... 309
Register 24: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0 ..... 311
Register 25: LDO Sleep Power Control (LDOSPCTL), offset 0x1B4 ..... 312
Register 26: LDO Sleep Power Calibration (LDOSPCAL), offset 0x1B8 ..... 314
Register 27: LDO Deep-Sleep Power Control (LDODPCTL), offset 0x1BC ..... 315
Register 28: LDO Deep-Sleep Power Calibration (LDODPCAL), offset 0x1C0 ..... 317
Register 29: Sleep / Deep-Sleep Power Mode Status (SDPMST), offset 0x1CC ..... 318
Register 30: Reset Behavior Control Register (RESBEHAVCTL), offset 0x1D8 ..... 321
Register 31: Hardware System Service Request (HSSR), offset 0x1F4 ..... 323
Register 32: USB Power Domain Status (USBPDS), offset 0x280 ..... 324

Register 33: USB Memory Power Control (USBMPC), offset 0x284 ..... 325
Register 34: Ethernet MAC Power Domain Status (EMACPDS), offset 0x288 ..... 326
Register 35: Ethernet MAC Memory Power Control (EMACMPC), offset 0x28C ..... 327
Register 36: LCD Power Domain Status (LCDPDS), offset 0x290 ..... 328
Register 37: LCD Memory Power Control (LCDMPC), offset 0x294 ..... 329
Register 38: CAN 0 Power Domain Status (CAN0PDS), offset 0x298 ..... 330
Register 39: CAN 0 Memory Power Control (CAN0MPC), offset 0x29C ..... 331
Register 40: CAN 1 Power Domain Status (CAN1PDS), offset 0x2A0 ..... 332
Register 41: CAN 1 Memory Power Control (CAN1MPC), offset 0x2A4 ..... 333
Register 42: Watchdog Timer Peripheral Present (PPWD), offset 0x300 ..... 334
Register 43: 16/32-Bit General-Purpose Timer Peripheral Present (PPTIMER), offset 0x304 ..... 335
Register 44: General-Purpose Input/Output Peripheral Present (PPGPIO), offset 0x308 ..... 337
Register 45: Micro Direct Memory Access Peripheral Present (PPDMA), offset 0x30C ..... 340
Register 46: EPI Peripheral Present (PPEPI), offset 0x310 ..... 341
Register 47: Hibernation Peripheral Present (PPHIB), offset 0x314 ..... 342
Register 48: Universal Asynchronous Receiver/Transmitter Peripheral Present (PPUART), offset 0x318 ..... 343
Register 49: Synchronous Serial Interface Peripheral Present (PPSSI), offset 0x31C ..... 345
Register 50: Inter-Integrated Circuit Peripheral Present (PPI2C), offset 0x320 ..... 347
Register 51: Universal Serial Bus Peripheral Present (PPUSB), offset 0x328 ..... 349
Register 52: Ethernet PHY Peripheral Present (PPEPHY), offset 0x330 ..... 350
Register 53: Controller Area Network Peripheral Present (PPCAN), offset 0x334 ..... 351
Register 54: Analog-to-Digital Converter Peripheral Present (PPADC), offset 0x338 ..... 352
Register 55: Analog Comparator Peripheral Present (PPACMP), offset 0x33C ..... 353
Register 56: Pulse Width Modulator Peripheral Present (PPPWM), offset 0x340 ..... 354
Register 57: Quadrature Encoder Interface Peripheral Present (PPQEI), offset 0x344 ..... 355
Register 58: Low Pin Count Interface Peripheral Present (PPLPC), offset 0x348 ..... 356
Register 59: Platform Environment Control Interface Peripheral Present (PPPECI), offset 0x350 ..... 357
Register 60: Fan Control Peripheral Present (PPFAN), offset 0x354 ..... 358
Register 61: EEPROM Peripheral Present (PPEEPROM), offset 0x358 ..... 359
Register 62: 32/64-Bit Wide General-Purpose Timer Peripheral Present (PPWTIMER), offset 0x35C ..... 360
Register 63: Remote Temperature Sensor Peripheral Present (PPRTS), offset 0x370 ..... 361
Register 64: CRC and Cryptographic Modules Peripheral Present (PPCCM), offset 0x374 ..... 362
Register 65: LCD Peripheral Present (PPLCD), offset 0x390 ..... 363
Register 66: 1-Wire Peripheral Present (PPOWIRE), offset 0x398 ..... 364
Register 67: Ethernet MAC Peripheral Present (PPEMAC), offset 0x39C ..... 365
Register 68: Power Regulator Bus Peripheral Present (PPPRB), offset 0x3A0 ..... 366
Register 69: Human Interface Master Peripheral Present (PPHIM), offset 0x3A4 ..... 367
Register 70: Watchdog Timer Software Reset (SRWD), offset 0x500 ..... 368
Register 71: 16/32-Bit General-Purpose Timer Software Reset (SRTIMER), offset 0x504 ..... 369
Register 72: General-Purpose Input/Output Software Reset (SRGPIO), offset 0x508 ..... 371
Register 73: Micro Direct Memory Access Software Reset (SRDMA), offset 0x50C ..... 375
Register 74: EPI Software Reset (SREPI), offset 0x510 ..... 376
Register 75: Hibernation Software Reset (SRHIB), offset 0x514 ..... 377
Register 76: Universal Asynchronous Receiver/Transmitter Software Reset (SRUART), offset 0x518 ..... 378
Register 77: Synchronous Serial Interface Software Reset (SRSSI), offset 0x51C ..... 380
Register 78: Inter-Integrated Circuit Software Reset (SRI2C), offset 0x520 ..... 382
Register 79: Universal Serial Bus Software Reset (SRUSB), offset 0x528 ..... 384

Register 80: Ethernet PHY Software Reset (SREPHY), offset 0x530 ..... 385
Register 81: Controller Area Network Software Reset (SRCAN), offset 0x534 ..... 386
Register 82: Analog-to-Digital Converter Software Reset (SRADC), offset 0x538 ..... 387
Register 83: Analog Comparator Software Reset (SRACMP), offset 0x53C ..... 388
Register 84: Pulse Width Modulator Software Reset (SRPWM), offset 0x540 ..... 389
Register 85: Quadrature Encoder Interface Software Reset (SRQEI), offset 0x544 ..... 390
Register 86: EEPROM Software Reset (SREEPROM), offset 0x558 ..... 391
Register 87: CRC and Cryptographic Modules Software Reset (SRCCM), offset 0x574 ..... 392
Register 88: LCD Controller Software Reset (SRLCD), offset 0x590 ..... 393
Register 89: 1-Wire Software Reset (SROWIRE), offset 0x598 ..... 394
Register 90: Ethernet MAC Software Reset (SREMAC), offset 0x59C ..... 395
Register 91: Watchdog Timer Run Mode Clock Gating Control (RCGCWD), offset 0x600 ..... 396
Register 92: 16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER), offset 0x604 ..... 397
Register 93: General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO), offset 0x608 ..... 399
Register 94: Micro Direct Memory Access Run Mode Clock Gating Control (RCGCDMA), offset 0x60C ..... 402
Register 95: EPI Run Mode Clock Gating Control (RCGCEPI), offset 0x610 ..... 403
Register 96: Hibernation Run Mode Clock Gating Control (RCGCHIB), offset 0x614 ..... 404
Register 97: Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART), offset 0x618 ..... 405
Register 98: Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI), offset 0x61C ..... 407
Register 99: Inter-Integrated Circuit Run Mode Clock Gating Control (RCGCI2C), offset 0x620 ..... 408
Register 100: Universal Serial Bus Run Mode Clock Gating Control (RCGCUSB), offset 0x628 ..... 410
Register 101: Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY), offset 0x630 ..... 411
Register 102: Controller Area Network Run Mode Clock Gating Control (RCGCCAN), offset 0x634 ..... 412
Register 103: Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC), offset 0x638 .... 413
Register 104: Analog Comparator Run Mode Clock Gating Control (RCGCACMP), offset 0x63C ..... 414
Register 105: Pulse Width Modulator Run Mode Clock Gating Control (RCGCPWM), offset 0x640 ..... 415
Register 106: Quadrature Encoder Interface Run Mode Clock Gating Control (RCGCQEI), offset 0x644 ..... 416
Register 107: EEPROM Run Mode Clock Gating Control (RCGCEEPROM), offset 0x658 ..... 417
Register 108: CRC and Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM), offset 0x674 ..... 418
Register 109: LCD Controller Run Mode Clock Gating Control (RCGCLCD), offset 0x690 ..... 419
Register 110: 1-Wire Run Mode Clock Gating Control (RCGCOWIRE), offset 0x698 ..... 420
Register 111: Ethernet MAC Run Mode Clock Gating Control (RCGCEMAC), offset 0x69C ..... 421
Register 112: Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD), offset 0x700 ..... 422
Register 113: 16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control (SCGCTIMER), offset 0x704 ..... 423
Register 114: General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO), offset 0x708 ..... 425
Register 115: Micro Direct Memory Access Sleep Mode Clock Gating Control (SCGCDMA), offset 0x70C ..... 428
Register 116: EPI Sleep Mode Clock Gating Control (SCGCEPI), offset 0x710 ..... 429
Register 117: Hibernation Sleep Mode Clock Gating Control (SCGCHIB), offset 0x714 ..... 430

Register 118: Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control (SCGCUART), offset 0x718 ..... 431
Register 119: Synchronous Serial Interface Sleep Mode Clock Gating Control (SCGCSSI), offset 0x71C ..... 433
Register 120: Inter-Integrated Circuit Sleep Mode Clock Gating Control (SCGCI2C), offset 0x720 ..... 434
Register 121: Universal Serial Bus Sleep Mode Clock Gating Control (SCGCUSB), offset 0x728 ..... 436
Register 122: Ethernet PHY Sleep Mode Clock Gating Control (SCGCEPHY), offset 0x730 ..... 437
Register 123: Controller Area Network Sleep Mode Clock Gating Control (SCGCCAN), offset 0x734 ..... 438
Register 124: Analog-to-Digital Converter Sleep Mode Clock Gating Control (SCGCADC), offset 0x738 ..... 439
Register 125: Analog Comparator Sleep Mode Clock Gating Control (SCGCACMP), offset 0x73C ..... 440
Register 126: Pulse Width Modulator Sleep Mode Clock Gating Control (SCGCPWM), offset 0x740 ..... 441
Register 127: Quadrature Encoder Interface Sleep Mode Clock Gating Control (SCGCQEI), offset 0x744 ..... 442
Register 128: EEPROM Sleep Mode Clock Gating Control (SCGCEEPROM), offset 0x758 ..... 443
Register 129: CRC and Cryptographic Modules Sleep Mode Clock Gating Control (SCGCCCM), offset 0x774 ..... 444
Register 130: LCD Controller Sleep Mode Clock Gating Control (SCGCLCD), offset 0x790 ..... 445
Register 131: 1-Wire Sleep Mode Clock Gating Control (SCGCOWIRE), offset 0x798 ..... 446
Register 132: Ethernet MAC Sleep Mode Clock Gating Control (SCGCEMAC), offset 0x79C ..... 447
Register 133: Watchdog Timer Deep-Sleep Mode Clock Gating Control (DCGCWD), offset 0x800 ..... 448
Register 134: 16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCTIMER), offset 0x804 ..... 449
Register 135: General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control (DCGCGPIO), offset 0x808 ..... 451
Register 136: Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control (DCGCDMA), offset 0x80C ..... 454
Register 137: EPI Deep-Sleep Mode Clock Gating Control (DCGCEPI), offset 0x810 ..... 455
Register 138: Hibernation Deep-Sleep Mode Clock Gating Control (DCGCHIB), offset 0x814 ..... 456
Register 139: Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control (DCGCUART), offset 0x818 ..... 457
Register 140: Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control (DCGCSSI), offset 0x81C ..... 459
Register 141: Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control (DCGCI2C), offset 0x820 ..... 460
Register 142: Universal Serial Bus Deep-Sleep Mode Clock Gating Control (DCGCUSB), offset 0x828 ..... 462
Register 143: Ethernet PHY Deep-Sleep Mode Clock Gating Control (DCGCEPHY), offset 0x830 ..... 463
Register 144: Controller Area Network Deep-Sleep Mode Clock Gating Control (DCGCCAN), offset 0x834 ..... 464
Register 145: Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control (DCGCADC), offset 0x838 ..... 465
Register 146: Analog Comparator Deep-Sleep Mode Clock Gating Control (DCGCACMP), offset 0x83C ..... 466
Register 147: Pulse Width Modulator Deep-Sleep Mode Clock Gating Control (DCGCPWM), offset 0x840 ..... 467
Register 148: Quadrature Encoder Interface Deep-Sleep Mode Clock Gating Control (DCGCQEI), offset 0x844 ..... 468
Register 149: EEPROM Deep-Sleep Mode Clock Gating Control (DCGCEEPROM), offset 0x858 ..... 469

Register 150: CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control (DCGCCCM), offset 0x874 ..... 470
Register 151: LCD Controller Deep-Sleep Mode Clock Gating Control (DCGCLCD), offset 0x890 ..... 471
Register 152: 1-Wire Deep-Sleep Mode Clock Gating Control (DCGCOWIRE), offset 0x898 ..... 472
Register 153: Ethernet MAC Deep-Sleep Mode Clock Gating Control (DCGCEMAC), offset 0x89C ..... 473
Register 154: Watchdog Timer Power Control (PCWD), offset 0x900 ..... 474
Register 155: 16/32-Bit General-Purpose Timer Power Control (PCTIMER), offset 0x904 ..... 476
Register 156: General-Purpose Input/Output Power Control (PCGPIO), offset 0x908 ..... 479
Register 157: Micro Direct Memory Access Power Control (PCDMA), offset 0x90C ..... 485
Register 158: External Peripheral Interface Power Control (PCEPI), offset 0x910 ..... 487
Register 159: Hibernation Power Control (PCHIB), offset 0x914 ..... 489
Register 160: Universal Asynchronous Receiver/Transmitter Power Control (PCUART), offset 0x918 ..... 491
Register 161: Synchronous Serial Interface Power Control (PCSSI), offset 0x91C ..... 494
Register 162: Inter-Integrated Circuit Power Control (PCI2C), offset 0x920 ..... 496
Register 163: Universal Serial Bus Power Control (PCUSB), offset 0x928 ..... 500
Register 164: Ethernet PHY Power Control (PCEPHY), offset 0x930 ..... 502
Register 165: Controller Area Network Power Control (PCCAN), offset 0x934 ..... 504
Register 166: Analog-to-Digital Converter Power Control (PCADC), offset 0x938 ..... 506
Register 167: Analog Comparator Power Control (PCACMP), offset 0x93C ..... 508
Register 168: Pulse Width Modulator Power Control (PCPWM), offset 0x940 ..... 510
Register 169: Quadrature Encoder Interface Power Control (PCQEI), offset 0x944 ..... 512
Register 170: EEPROM Power Control (PCEEPROM), offset 0x958 ..... 514
Register 171: CRC and Cryptographic Modules Power Control (PCCCM), offset 0x974 ..... 516
Register 172: LCD Controller Power Control (PCLCD), offset 0x990 ..... 518
Register 173: 1-Wire Power Control (PCOWIRE), offset 0x998 ..... 520
Register 174: Ethernet MAC Power Control (PCEMAC), offset 0x99C ..... 522
Register 175: Watchdog Timer Peripheral Ready (PRWD), offset 0xA00 ..... 524
Register 176: 16/32-Bit General-Purpose Timer Peripheral Ready (PRTIMER), offset 0xA04 ..... 525
Register 177: General-Purpose Input/Output Peripheral Ready (PRGPIO), offset 0xA08 ..... 527
Register 178: Micro Direct Memory Access Peripheral Ready (PRDMA), offset 0xA0C ..... 531
Register 179: EPI Peripheral Ready (PREPI), offset 0xA10 ..... 532
Register 180: Hibernation Peripheral Ready (PRHIB), offset 0xA14 ..... 533
Register 181: Universal Asynchronous Receiver/Transmitter Peripheral Ready (PRUART), offset 0xA18 ..... 534
Register 182: Synchronous Serial Interface Peripheral Ready (PRSSI), offset 0xA1C ..... 536
Register 183: Inter-Integrated Circuit Peripheral Ready (PRI2C), offset 0xA20 ..... 538
Register 184: Universal Serial Bus Peripheral Ready (PRUSB), offset 0xA28 ..... 541
Register 185: Ethernet PHY Peripheral Ready (PREPHY), offset 0xA30 ..... 542
Register 186: Controller Area Network Peripheral Ready (PRCAN), offset 0xA34 ..... 543
Register 187: Analog-to-Digital Converter Peripheral Ready (PRADC), offset 0xA38 ..... 544
Register 188: Analog Comparator Peripheral Ready (PRACMP), offset 0xA3C ..... 545
Register 189: Pulse Width Modulator Peripheral Ready (PRPWM), offset 0xA40 ..... 546
Register 190: Quadrature Encoder Interface Peripheral Ready (PRQEI), offset 0xA44 ..... 547
Register 191: EEPROM Peripheral Ready (PREEPROM), offset 0xA58 ..... 548
Register 192: CRC and Cryptographic Modules Peripheral Ready (PRCCM), offset 0xA74 ..... 549
Register 193: LCD Controller Peripheral Ready (PRLCD), offset 0xA90 ..... 550
Register 194: 1-Wire Peripheral Ready (PROWIRE), offset 0xA98 ..... 551
Register 195: Ethernet MAC Peripheral Ready (PREMAC), offset 0xA9C ..... 552

Register 196: Unique ID 0 (UNIQUEID0), offset 0xF20 ..... 553
Register 197: Unique ID 1 (UNIQUEID1), offset 0xF24 ..... 553
Register 198: Unique ID 2 (UNIQUEID2), offset 0xF28 ..... 553
Register 199: Unique ID 3 (UNIQUEID3), offset 0xF2C ..... 553
Register 200: Cryptographic Modules Clock Gating Request (CCMCGREQ), offset 0x204 ..... 554

**Processor Support and Exception Module ..... 555**
Register 1: System Exception Raw Interrupt Status (SYSEXCRIS), offset 0x000 ..... 556
Register 2: System Exception Interrupt Mask (SYSEXCIM), offset 0x004 ..... 558
Register 3: System Exception Masked Interrupt Status (SYSEXCMIS), offset 0x008 ..... 560
Register 4: System Exception Interrupt Clear (SYSEXCIC), offset 0x00C ..... 562

**Hibernation Module ..... 563**
Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000 ..... 586
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004 ..... 587
Register 3: Hibernation RTC Load (HIBRTCLD), offset 0x00C ..... 588
Register 4: Hibernation Control (HIBCTL), offset 0x010 ..... 589
Register 5: Hibernation Interrupt Mask (HIBIM), offset 0x014 ..... 594
Register 6: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018 ..... 596
Register 7: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C ..... 598
Register 8: Hibernation Interrupt Clear (HIBIC), offset 0x020 ..... 600
Register 9: Hibernation RTC Trim (HIBRTCT), offset 0x024 ..... 602
Register 10: Hibernation RTC Sub Seconds (HIBRTCSS), offset 0x028 ..... 603
Register 11: Hibernation IO Configuration (HIBIO), offset 0x02C ..... 604
Register 12: Hibernation Data (HIBDATA), offset 0x030-0x06F ..... 606
Register 13: Hibernation Calendar Control (HIBCALCTL), offset 0x300 ..... 607
Register 14: Hibernation Calendar 0 (HIBCAL0), offset 0x310 ..... 608
Register 15: Hibernation Calendar 1 (HIBCAL1), offset 0x314 ..... 610
Register 16: Hibernation Calendar Load 0 (HIBCALLD0), offset 0x320 ..... 612
Register 17: Hibernation Calendar Load (HIBCALLD1), offset 0x324 ..... 614
Register 18: Hibernation Calendar Match 0 (HIBCALM0), offset 0x330 ..... 615
Register 19: Hibernation Calendar Match 1 (HIBCALM1), offset 0x334 ..... 617
Register 20: Hibernation Lock (HIBLOCK), offset 0x360 ..... 618
Register 21: HIB Tamper Control (HIBTPCTL), offset 0x400 ..... 619
Register 22: HIB Tamper Status (HIBTPSTAT), offset 0x404 ..... 621
Register 23: HIB Tamper I/O Control (HIBTPIO), offset 0x410 ..... 623
Register 24: HIB Tamper Log 0 (HIBTPLOG0), offset 0x4E0 ..... 627
Register 25: HIB Tamper Log 2 (HIBTPLOG2), offset 0x4E8 ..... 627
Register 26: HIB Tamper Log 4 (HIBTPLOG4), offset 0x4F0 ..... 627
Register 27: HIB Tamper Log 6 (HIBTPLOG6), offset 0x4F8 ..... 627
Register 28: HIB Tamper Log 1 (HIBTPLOG1), offset 0x4E4 ..... 628
Register 29: HIB Tamper Log 3 (HIBTPLOG3), offset 0x4EC ..... 628
Register 30: HIB Tamper Log 5 (HIBTPLOG5), offset 0x4F4 ..... 628
Register 31: HIB Tamper Log 7 (HIBTPLOG7), offset 0x4FC ..... 628
Register 32: Hibernation Peripheral Properties (HIBPP) , offset 0xFC0 ..... 630
Register 33: Hibernation Clock Control (HIBCC), offset 0xFC8 ..... 631

**Internal Memory ..... 632**
Register 1: Flash Memory Address (FMA), offset 0x000 ..... 656
Register 2: Flash Memory Data (FMD), offset 0x004 ..... 657
Register 3: Flash Memory Control (FMC), offset 0x008 ..... 658

Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C ..... 661
Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010 ..... 664
Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 ..... 666
Register 7: Flash Memory Control 2 (FMC2), offset 0x020 ..... 669
Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030 ..... 670
Register 9: Flash Program/Erase Key (FLPEKEY), offset 0x03C ..... 671
Register 10: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C ..... 672
Register 11: Flash Peripheral Properties (FLASHPP), offset 0xFC0 ..... 673
Register 12: SRAM Size (SSIZE), offset 0xFC4 ..... 675
Register 13: Flash Configuration Register (FLASHCONF), offset 0xFC8 ..... 676
Register 14: ROM Third-Party Software (ROMSWMAP), offset 0xFCC ..... 678
Register 15: Flash DMA Address Size (FLASHDMASZ), offset 0xFD0 ..... 680
Register 16: Flash DMA Starting Address (FLASHDMAST), offset 0xFD4 ..... 681
Register 17: EEPROM Size Information (EESIZE), offset 0x000 ..... 682
Register 18: EEPROM Current Block (EEBLOCK), offset 0x004 ..... 683
Register 19: EEPROM Current Offset (EEOFFSET), offset 0x008 ..... 684
Register 20: EEPROM Read-Write (EERDWR), offset 0x010 ..... 685
Register 21: EEPROM Read-Write with Increment (EERDWRINC), offset 0x014 ..... 686
Register 22: EEPROM Done Status (EEDONE), offset 0x018 ..... 687
Register 23: EEPROM Support Control and Status (EESUPP), offset 0x01C ..... 689
Register 24: EEPROM Unlock (EEUNLOCK), offset 0x020 ..... 690
Register 25: EEPROM Protection (EEPROT), offset 0x030 ..... 691
Register 26: EEPROM Password (EEPASS0), offset 0x034 ..... 693
Register 27: EEPROM Password (EEPASS1), offset 0x038 ..... 693
Register 28: EEPROM Password (EEPASS2), offset 0x03C ..... 693
Register 29: EEPROM Interrupt (EEINT), offset 0x040 ..... 694
Register 30: EEPROM Block Hide 0 (EEHIDE0), offset 0x050 ..... 695
Register 31: EEPROM Block Hide 1 (EEHIDE1), offset 0x054 ..... 696
Register 32: EEPROM Block Hide 2 (EEHIDE2), offset 0x058 ..... 696
Register 33: EEPROM Debug Mass Erase (EEDBGME), offset 0x080 ..... 697
Register 34: EEPROM Peripheral Properties (EEPROMPP), offset 0xFC0 ..... 698
Register 35: Reset Vector Pointer (RVP), offset 0x0D4 ..... 699
Register 36: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x200 ..... 700
Register 37: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204 ..... 700
Register 38: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208 ..... 700
Register 39: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C ..... 700
Register 40: Flash Memory Protection Read Enable 4 (FMPRE4), offset 0x210 ..... 700
Register 41: Flash Memory Protection Read Enable 5 (FMPRE5), offset 0x214 ..... 700
Register 42: Flash Memory Protection Read Enable 6 (FMPRE6), offset 0x218 ..... 700
Register 43: Flash Memory Protection Read Enable 7 (FMPRE7), offset 0x21C ..... 700
Register 44: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x400 ..... 702
Register 45: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404 ..... 702
Register 46: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408 ..... 702
Register 47: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C ..... 702
Register 48: Flash Memory Protection Program Enable 4 (FMPPE4), offset 0x410 ..... 702
Register 49: Flash Memory Protection Program Enable 5 (FMPPE5), offset 0x414 ..... 702
Register 50: Flash Memory Protection Program Enable 6 (FMPPE6), offset 0x418 ..... 702
Register 51: Flash Memory Protection Program Enable 7 (FMPPE7), offset 0x41C ..... 702

Register 52: Boot Configuration (BOOTCFG), offset 0x1D0 ..... 704
Register 53: User Register 0 (USER_REG0), offset 0x1E0 ..... 707
Register 54: User Register 1 (USER_REG1), offset 0x1E4 ..... 707
Register 55: User Register 2 (USER_REG2), offset 0x1E8 ..... 707
Register 56: User Register 3 (USER_REG3), offset 0x1EC ..... 707

**Micro Direct Memory Access (µDMA) ..... 708**

Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000 ..... 733
Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004 ..... 734
Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008 ..... 735
Register 4: DMA Status (DMASTAT), offset 0x000 ..... 740
Register 5: DMA Configuration (DMACFG), offset 0x004 ..... 742
Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008 ..... 743
Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C ..... 744
Register 8: DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010 ..... 745
Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014 ..... 746
Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018 ..... 747
Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C ..... 748
Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020 ..... 749
Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024 ..... 750
Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028 ..... 751
Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C ..... 752
Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030 ..... 753
Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034 ..... 754
Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038 ..... 755
Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C ..... 756
Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C ..... 757
Register 21: DMA Channel Assignment (DMACHASGN), offset 0x500 ..... 758
Register 22: DMA Channel Map Select 0 (DMACHMAP0), offset 0x510 ..... 759
Register 23: DMA Channel Map Select 1 (DMACHMAP1), offset 0x514 ..... 760
Register 24: DMA Channel Map Select 2 (DMACHMAP2), offset 0x518 ..... 761
Register 25: DMA Channel Map Select 3 (DMACHMAP3), offset 0x51C ..... 762
Register 26: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0 ..... 763
Register 27: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4 ..... 764
Register 28: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8 ..... 765
Register 29: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC ..... 766
Register 30: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0 ..... 767
Register 31: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0 ..... 768
Register 32: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4 ..... 769
Register 33: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8 ..... 770
Register 34: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC ..... 771

**General-Purpose Input/Outputs (GPIOs) ..... 772**

Register 1: GPIO Data (GPIODATA), offset 0x000 ..... 791
Register 2: GPIO Direction (GPIODIR), offset 0x400 ..... 792
Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404 ..... 793
Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 ..... 794
Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C ..... 796
Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410 ..... 797
Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414 ..... 798

Register 8: GPIO Masked Interrupt Status (GPIOMIS), offset 0x418 ..... 800
Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C ..... 802
Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 ..... 803
Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 ..... 805
Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 ..... 806
Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 ..... 807
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C ..... 808
Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510 ..... 809
Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514 ..... 811
Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 ..... 813
Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C ..... 814
Register 19: GPIO Lock (GPIOLOCK), offset 0x520 ..... 816
Register 20: GPIO Commit (GPIOCR), offset 0x524 ..... 817
Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528 ..... 819
Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C ..... 820
Register 23: GPIO ADC Control (GPIOADCCTL), offset 0x530 ..... 822
Register 24: GPIO DMA Control (GPIODMACTL), offset 0x534 ..... 823
Register 25: GPIO Select Interrupt (GPIOSI), offset 0x538 ..... 824
Register 26: GPIO 12-mA Drive Select (GPIODR12R), offset 0x53C ..... 825
Register 27: GPIO Wake Pin Enable (GPIOWAKEPEN), offset 0x540 ..... 826
Register 28: GPIO Wake Level (GPIOWAKELVL), offset 0x544 ..... 828
Register 29: GPIO Wake Status (GPIOWAKESTAT), offset 0x548 ..... 830
Register 30: GPIO Peripheral Property (GPIOPP), offset 0xFC0 ..... 832
Register 31: GPIO Peripheral Configuration (GPIOPC), offset 0xFC4 ..... 833
Register 32: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 ..... 836
Register 33: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 ..... 837
Register 34: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 ..... 838
Register 35: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC ..... 839
Register 36: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 ..... 840
Register 37: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 ..... 841
Register 38: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 ..... 842
Register 39: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC ..... 843
Register 40: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 ..... 844
Register 41: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 ..... 845
Register 42: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 ..... 846
Register 43: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC ..... 847

**External Peripheral Interface (EPI) ..... 848**

Register 1: EPI Configuration (EPICFG), offset 0x000 ..... 890
Register 2: EPI Main Baud Rate (EPIBAUD), offset 0x004 ..... 892
Register 3: EPI Main Baud Rate (EPIBAUD2), offset 0x008 ..... 894
Register 4: EPI SDRAM Configuration (EPISDRAMCFG), offset 0x010 ..... 896
Register 5: EPI Host-Bus 8 Configuration (EPIHB8CFG), offset 0x010 ..... 898
Register 6: EPI Host-Bus 16 Configuration (EPIHB16CFG), offset 0x010 ..... 903
Register 7: EPI General-Purpose Configuration (EPIGPCFG), offset 0x010 ..... 909
Register 8: EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2), offset 0x014 ..... 912
Register 9: EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2), offset 0x014 ..... 918
Register 10: EPI Address Map (EPIADDRMAP), offset 0x01C ..... 925
Register 11: EPI Read Size 0 (EPIRSIZE0), offset 0x020 ..... 928

Register 12: EPI Read Size 1 (EPIRSIZE1), offset 0x030 ..... 928
Register 13: EPI Read Address 0 (EPIRADDR0), offset 0x024 ..... 929
Register 14: EPI Read Address 1 (EPIRADDR1), offset 0x034 ..... 929
Register 15: EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028 ..... 930
Register 16: EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038 ..... 930
Register 17: EPI Status (EPISTAT), offset 0x060 ..... 932
Register 18: EPI Read FIFO Count (EPIRFIFOCNT), offset 0x06C ..... 934
Register 19: EPI Read FIFO (EPIREADFIFO0), offset 0x070 ..... 935
Register 20: EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074 ..... 935
Register 21: EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078 ..... 935
Register 22: EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C ..... 935
Register 23: EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080 ..... 935
Register 24: EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084 ..... 935
Register 25: EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088 ..... 935
Register 26: EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C ..... 935
Register 27: EPI FIFO Level Selects (EPIFIFOLVL), offset 0x200 ..... 936
Register 28: EPI Write FIFO Count (EPIWFIFOCNT), offset 0x204 ..... 938
Register 29: EPI DMA Transmit Count (EPIDMATXCNT), offset 0x208 ..... 939
Register 30: EPI Interrupt Mask (EPIIM), offset 0x210 ..... 940
Register 31: EPI Raw Interrupt Status (EPIRIS), offset 0x214 ..... 942
Register 32: EPI Masked Interrupt Status (EPIMIS), offset 0x218 ..... 944
Register 33: EPI Error and Interrupt Status and Clear (EPIEISC), offset 0x21C ..... 946
Register 34: EPI Host-Bus 8 Configuration 3 (EPIHB8CFG3), offset 0x308 ..... 948
Register 35: EPI Host-Bus 16 Configuration 3 (EPIHB16CFG3), offset 0x308 ..... 951
Register 36: EPI Host-Bus 8 Configuration 4 (EPIHB8CFG4), offset 0x30C ..... 955
Register 37: EPI Host-Bus 16 Configuration 4 (EPIHB16CFG4), offset 0x30C ..... 958
Register 38: EPI Host-Bus 8 Timing Extension (EPIHB8TIME), offset 0x310 ..... 962
Register 39: EPI Host-Bus 16 Timing Extension (EPIHB16TIME), offset 0x310 ..... 964
Register 40: EPI Host-Bus 8 Timing Extension (EPIHB8TIME2), offset 0x314 ..... 966
Register 41: EPI Host-Bus 16 Timing Extension (EPIHB16TIME2), offset 0x314 ..... 968
Register 42: EPI Host-Bus 8 Timing Extension (EPIHB8TIME3), offset 0x318 ..... 970
Register 43: EPI Host-Bus 16 Timing Extension (EPIHB16TIME3), offset 0x318 ..... 972
Register 44: EPI Host-Bus 8 Timing Extension (EPIHB8TIME4), offset 0x31C ..... 974
Register 45: EPI Host-Bus 16 Timing Extension (EPIHB16TIME4), offset 0x31C ..... 976
Register 46: EPI Host-Bus PSRAM (EPIHBPSRAM), offset 0x360 ..... 978

**Cyclical Redundancy Check (CRC) ..... 979**
Register 1: CRC Control (CRCCTRL), offset 0x400 ..... 983
Register 2: CRC SEED/Context (CRCSEED), offset 0x410 ..... 985
Register 3: CRC Data Input (CRCDIN), offset 0x414 ..... 986
Register 4: CRC Post Processing Result (CRCRSLTPP), offset 0x418 ..... 987

**Advance Encryption Standard Accelerator (AES) ..... 988**
Register 1: AES Key 2_6 (AES_KEY2_6), offset 0x000 ..... 1012
Register 2: AES Key 2_7 (AES_KEY2_7), offset 0x004 ..... 1012
Register 3: AES Key 2_4 (AES_KEY2_4), offset 0x008 ..... 1012
Register 4: AES Key 2_5 (AES_KEY2_5), offset 0x00C ..... 1012
Register 5: AES Key 2_2 (AES_KEY2_2), offset 0x010 ..... 1012
Register 6: AES Key 2_3 (AES_KEY2_3), offset 0x014 ..... 1012
Register 7: AES Key 2_0 (AES_KEY2_0), offset 0x018 ..... 1012

Register 8: AES Key 2_1 (AES_KEY2_1), offset 0x01C ..... 1012
Register 9: AES Key 1_6 (AES_KEY1_6), offset 0x020 ..... 1012
Register 10: AES Key 1_7 (AES_KEY1_7), offset 0x024 ..... 1012
Register 11: AES Key 1_4 (AES_KEY1_4), offset 0x028 ..... 1012
Register 12: AES Key 1_5 (AES_KEY1_5), offset 0x02C ..... 1012
Register 13: AES Key 1_2 (AES_KEY1_2), offset 0x030 ..... 1012
Register 14: AES Key 1_3 (AES_KEY1_3), offset 0x034 ..... 1012
Register 15: AES Key 1_0 (AES_KEY1_0), offset 0x038 ..... 1012
Register 16: AES Key 1_1 (AES_KEY1_1), offset 0x03C ..... 1012
Register 17: AES Initialization Vector Input 0 (AES_IV_IN_0), offset 0x040 ..... 1014
Register 18: AES Initialization Vector Input 1 (AES_IV_IN_1), offset 0x044 ..... 1014
Register 19: AES Initialization Vector Input 2 (AES_IV_IN_2), offset 0x048 ..... 1014
Register 20: AES Initialization Vector Input 3 (AES_IV_IN_3), offset 0x04C ..... 1014
Register 21: AES Control (AES_CTRL), offset 0x050 ..... 1015
Register 22: AES Crypto Data Length 0 (AES_C_LENGTH_0), offset 0x054 ..... 1020
Register 23: AES Crypto Data Length 1 (AES_C_LENGTH_1), offset 0x058 ..... 1020
Register 24: AES Authentication Data Length (AES_AUTH_LENGTH), offset 0x05C ..... 1021
Register 25: AES Data RW Plaintext/Ciphertext 0 (AES_DATA_IN_0), offset 0x060 ..... 1022
Register 26: AES Data RW Plaintext/Ciphertext 1 (AES_DATA_IN_1), offset 0x064 ..... 1022
Register 27: AES Data RW Plaintext/Ciphertext 2 (AES_DATA_IN_2), offset 0x068 ..... 1022
Register 28: AES Data RW Plaintext/Ciphertext 3 (AES_DATA_IN_3), offset 0x06C ..... 1022
Register 29: AES Hash Tag Out 0 (AES_TAG_OUT_0), offset 0x070 ..... 1023
Register 30: AES Hash Tag Out 1 (AES_TAG_OUT_1), offset 0x074 ..... 1023
Register 31: AES Hash Tag Out 2 (AES_TAG_OUT_2), offset 0x078 ..... 1023
Register 32: AES Hash Tag Out 3 (AES_TAG_OUT_3), offset 0x07C ..... 1023
Register 33: AES IP Revision Identifier (AES_REVISION), offset 0x080 ..... 1024
Register 34: AES System Configuration (AES_SYSCONFIG), offset 0x084 ..... 1025
Register 35: AES System Status (AES_SYSSTATUS), offset 0x088 ..... 1028
Register 36: AES Interrupt Status (AES_IRQSTATUS), offset 0x08C ..... 1029
Register 37: AES Interrupt Enable (AES_IRQENABLE), offset 0x090 ..... 1031
Register 38: AES Dirty Bits (AES_DIRTYBITS), offset 0x094 ..... 1033
Register 39: AES DMA Interrupt Mask (AES_DMAIM), offset 0x020 ..... 1034
Register 40: AES DMA Raw Interrupt Status (AES_DMARIS), offset 0x024 ..... 1036
Register 41: AES DMA Masked Interrupt Status (AES_DMAMIS), offset 0x028 ..... 1038
Register 42: AES DMA Interrupt Clear (AES_DMAIC), offset 0x02C ..... 1039

**Data Encryption Standard Accelerator (DES) ..... 1040**

Register 1: DES Key 3 LSW for 192-Bit Key (DES_KEY3_L), offset 0x000 ..... 1051
Register 2: DES Key 3 MSW for 192-Bit Key (DES_KEY3_H), offset 0x004 ..... 1051
Register 3: DES Key 2 LSW for 128-Bit Key (DES_KEY2_L), offset 0x008 ..... 1051
Register 4: DES Key 2 MSW for 128-Bit Key (DES_KEY2_H), offset 0x00C ..... 1051
Register 5: DES Key 1 LSW for 64-Bit Key (DES_KEY1_L), offset 0x010 ..... 1051
Register 6: DES Key 1 MSW for 64-Bit Key (DES_KEY1_H), offset 0x014 ..... 1051
Register 7: DES Initialization Vector (DES_IV_L), offset 0x018 ..... 1052
Register 8: DES Initialization Vector (DES_IV_H), offset 0x01C ..... 1053
Register 9: DES Control (DES_CTRL), offset 0x020 ..... 1054
Register 10: DES Cryptographic Data Length (DES_LENGTH), offset 0x024 ..... 1055
Register 11: DES LSW Data RW (DES_DATA_L), offset 0x028 ..... 1056
Register 12: DES MSW Data RW (DES_DATA_H), offset 0x02C ..... 1057

Register 13: DES Revision Number (DES_REVISION), offset 0x030 ..... 1058
Register 14: DES System Configuration (DES_SYSCONFIG), offset 0x034 ..... 1059
Register 15: DES System Status (DES_SYSSTATUS), offset 0x038 ..... 1061
Register 16: DES Interrupt Status (DES_IRQSTATUS), offset 0x03C ..... 1062
Register 17: DES Interrupt Enable (DES_IRQENABLE), offset 0x040 ..... 1063
Register 18: DES Dirty Bits (DES_DIRTYBITS), offset 0x044 ..... 1064
Register 19: DES DMA Interrupt Mask (DES_DMAIM), offset 0x030 ..... 1065
Register 20: DES DMA Raw Interrupt Status (DES_DMARIS), offset 0x034 ..... 1066
Register 21: DES DMA Masked Interrupt Status (DES_DMAMIS), offset 0x038 ..... 1067
Register 22: DES DMA Interrupt Clear (DES_DMAIC), offset 0x03C ..... 1068

**SHA/MD5 Accelerator ..... 1069**

Register 1: SHA Outer Digest A (SHA_ODIGEST_A), offset 0x000 ..... 1087
Register 2: SHA Outer Digest B (SHA_ODIGEST_B), offset 0x004 ..... 1087
Register 3: SHA Outer Digest C (SHA_ODIGEST_C), offset 0x008 ..... 1087
Register 4: SHA Outer Digest D (SHA_ODIGEST_D), offset 0x00C ..... 1087
Register 5: SHA Outer Digest E (SHA_ODIGEST_E), offset 0x010 ..... 1087
Register 6: SHA Outer Digest F (SHA_ODIGEST_F), offset 0x014 ..... 1087
Register 7: SHA Outer Digest G (SHA_ODIGEST_G), offset 0x018 ..... 1087
Register 8: SHA Outer Digest H (SHA_ODIGEST_H), offset 0x01C ..... 1087
Register 9: SHA Inner Digest A (SHA_IDIGEST_A), offset 0x020 ..... 1087
Register 10: SHA Inner Digest B (SHA_IDIGEST_B), offset 0x024 ..... 1087
Register 11: SHA Inner Digest C (SHA_IDIGEST_C), offset 0x028 ..... 1087
Register 12: SHA Inner Digest D (SHA_IDIGEST_D), offset 0x02C ..... 1087
Register 13: SHA Inner Digest E (SHA_IDIGEST_E), offset 0x030 ..... 1087
Register 14: SHA Inner Digest F (SHA_IDIGEST_F), offset 0x034 ..... 1087
Register 15: SHA Inner Digest G (SHA_IDIGEST_G), offset 0x038 ..... 1087
Register 16: SHA Inner Digest H (SHA_IDIGEST_H), offset 0x03C ..... 1087
Register 17: SHA Digest Count (SHA_DIGEST_COUNT), offset 0x040 ..... 1088
Register 18: SHA Mode (SHA_MODE), offset 0x044 ..... 1089
Register 19: SHA Length (SHA_LENGTH), offset 0x048 ..... 1091
Register 20: SHA Data 0 Input (SHA_DATA_0_IN), offset 0x080 ..... 1092
Register 21: SHA Data 1 Input (SHA_DATA_1_IN), offset 0x084 ..... 1092
Register 22: SHA Data 2 Input (SHA_DATA_2_IN), offset 0x088 ..... 1092
Register 23: SHA Data 3 Input (SHA_DATA_3_IN), offset 0x08C ..... 1092
Register 24: SHA Data 4 Input (SHA_DATA_4_IN), offset 0x090 ..... 1092
Register 25: SHA Data 5 Input (SHA_DATA_5_IN), offset 0x094 ..... 1092
Register 26: SHA Data 6 Input (SHA_DATA_6_IN), offset 0x098 ..... 1092
Register 27: SHA Data 7 Input (SHA_DATA_7_IN), offset 0x09C ..... 1092
Register 28: SHA Data 8 Input (SHA_DATA_8_IN), offset 0x0A0 ..... 1092
Register 29: SHA Data 9 Input (SHA_DATA_9_IN), offset 0x0A4 ..... 1092
Register 30: SHA Data 10 Input (SHA_DATA_10_IN), offset 0x0A8 ..... 1092
Register 31: SHA Data 11 Input (SHA_DATA_11_IN), offset 0x0AC ..... 1092
Register 32: SHA Data 12 Input (SHA_DATA_12_IN), offset 0x0B0 ..... 1092
Register 33: SHA Data 13 Input (SHA_DATA_13_IN), offset 0x0B4 ..... 1092
Register 34: SHA Data 14 Input (SHA_DATA_14_IN), offset 0x0B8 ..... 1092
Register 35: SHA Data 15 Input (SHA_DATA_15_IN), offset 0x0BC ..... 1092
Register 36: SHA Revision (SHA_REVISION), offset 0x100 ..... 1093
Register 37: SHA System Configuration (SHA_SYSCONFIG), offset 0x110 ..... 1094

Register 38: SHA System Status (SHA_SYSSTATUS), offset 0x114 ..... 1096
Register 39: SHA Interrupt Status (SHA_IRQSTATUS), offset 0x118 ..... 1097
Register 40: SHA Interrupt Enable (SHA_IRQENABLE), offset 0x11C ..... 1098
Register 41: SHA DMA Interrupt Mask (SHA_DMAIM), offset 0x010 ..... 1100
Register 42: SHA DMA Raw Interrupt Status (SHA_DMARIS), offset 0x014 ..... 1101
Register 43: SHA DMA Masked Interrupt Status (SHA_DMAMIS), offset 0x018 ..... 1102
Register 44: SHA DMA Interrupt Clear (SHA_DMAIC), offset 0x01C ..... 1103

**General-Purpose Timers ..... 1104**
Register 1: GPTM Configuration (GPTMCFG), offset 0x000 ..... 1126
Register 2: GPTM Timer A Mode (GPTMTAMR), offset 0x004 ..... 1127
Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008 ..... 1132
Register 4: GPTM Control (GPTMCTL), offset 0x00C ..... 1136
Register 5: GPTM Synchronize (GPTMSYNC), offset 0x010 ..... 1140
Register 6: GPTM Interrupt Mask (GPTMIMR), offset 0x018 ..... 1143
Register 7: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C ..... 1146
Register 8: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 ..... 1149
Register 9: GPTM Interrupt Clear (GPTMICR), offset 0x024 ..... 1152
Register 10: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028 ..... 1154
Register 11: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C ..... 1155
Register 12: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030 ..... 1156
Register 13: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034 ..... 1157
Register 14: GPTM Timer A Prescale (GPTMTAPR), offset 0x038 ..... 1158
Register 15: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C ..... 1159
Register 16: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040 ..... 1160
Register 17: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044 ..... 1161
Register 18: GPTM Timer A (GPTMTAR), offset 0x048 ..... 1162
Register 19: GPTM Timer B (GPTMTBR), offset 0x04C ..... 1163
Register 20: GPTM Timer A Value (GPTMTAV), offset 0x050 ..... 1164
Register 21: GPTM Timer B Value (GPTMTBV), offset 0x054 ..... 1165
Register 22: GPTM RTC Predivide (GPTMRTCPD), offset 0x058 ..... 1166
Register 23: GPTM Timer A Prescale Snapshot (GPTMTAPS), offset 0x05C ..... 1167
Register 24: GPTM Timer B Prescale Snapshot (GPTMTBPS), offset 0x060 ..... 1168
Register 25: GPTM DMA Event (GPTMDMAEV), offset 0x06C ..... 1169
Register 26: GPTM ADC Event (GPTMADCEV), offset 0x070 ..... 1172
Register 27: GPTM Peripheral Properties (GPTMPP), offset 0xFC0 ..... 1175
Register 28: GPTM Clock Configuration (GPTMCC), offset 0xFC8 ..... 1177

**Watchdog Timers ..... 1178**
Register 1: Watchdog Load (WDTLOAD), offset 0x000 ..... 1182
Register 2: Watchdog Value (WDTVALUE), offset 0x004 ..... 1183
Register 3: Watchdog Control (WDTCTL), offset 0x008 ..... 1184
Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C ..... 1186
Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 ..... 1187
Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 ..... 1188
Register 7: Watchdog Test (WDTTEST), offset 0x418 ..... 1189
Register 8: Watchdog Lock (WDTLOCK), offset 0xC00 ..... 1190
Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 ..... 1191
Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 ..... 1192
Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 ..... 1193

Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC ..... 1194
Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 ..... 1195
Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 ..... 1196
Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 ..... 1197
Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC ..... 1198
Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 ..... 1199
Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 ..... 1200
Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 ..... 1201
Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3 ), offset 0xFFC ..... 1202

**Analog-to-Digital Converter (ADC) ..... 1203**

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000 ..... 1228
Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004 ..... 1230
Register 3: ADC Interrupt Mask (ADCIM), offset 0x008 ..... 1233
Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C ..... 1236
Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010 ..... 1240
Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014 ..... 1242
Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018 ..... 1247
Register 8: ADC Trigger Source Select (ADCTSSEL), offset 0x01C ..... 1248
Register 9: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 ..... 1250
Register 10: ADC Sample Phase Control (ADCSPC), offset 0x024 ..... 1252
Register 11: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 ..... 1254
Register 12: ADC Sample Averaging Control (ADCSAC), offset 0x030 ..... 1256
Register 13: ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034 ..... 1257
Register 14: ADC Control (ADCCTL), offset 0x038 ..... 1259
Register 15: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 ..... 1260
Register 16: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 ..... 1262
Register 17: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 ..... 1269
Register 18: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 ..... 1269
Register 19: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 ..... 1269
Register 20: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 ..... 1269
Register 21: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C ..... 1270
Register 22: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C ..... 1270
Register 23: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C ..... 1270
Register 24: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC ..... 1270
Register 25: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050 ..... 1272
Register 26: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054 ..... 1274
Register 27: ADC Sample Sequence Extended Input Multiplexer Select 0 (ADCSSEMUX0), offset 0x058 ..... 1276
Register 28: ADC Sample Sequence 0 Sample and Hold Time (ADCSSTSH0), offset 0x05C ..... 1278
Register 29: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 ..... 1280
Register 30: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 ..... 1280
Register 31: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 ..... 1281
Register 32: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 ..... 1281
Register 33: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070 ..... 1285
Register 34: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090 ..... 1285
Register 35: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074 ..... 1286
Register 36: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094 ..... 1286

Register 37: ADC Sample Sequence Extended Input Multiplexer Select 1 (ADCSSEMUX1), offset 0x078 ..... 1288
Register 38: ADC Sample Sequence Extended Input Multiplexer Select 2 (ADCSSEMUX2), offset 0x098 ..... 1288
Register 39: ADC Sample Sequence 1 Sample and Hold Time (ADCSSTSH1), offset 0x07C ..... 1290
Register 40: ADC Sample Sequence 2 Sample and Hold Time (ADCSSTSH2), offset 0x09C ..... 1290
Register 41: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 ..... 1292
Register 42: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 ..... 1293
Register 43: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0 ..... 1295
Register 44: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4 ..... 1296
Register 45: ADC Sample Sequence Extended Input Multiplexer Select 3 (ADCSSEMUX3), offset 0x0B8 ..... 1297
Register 46: ADC Sample Sequence 3 Sample and Hold Time (ADCSSTSH3), offset 0x0BC ..... 1298
Register 47: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00 ..... 1299
Register 48: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00 ..... 1304
Register 49: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04 ..... 1304
Register 50: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08 ..... 1304
Register 51: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C ..... 1304
Register 52: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10 ..... 1304
Register 53: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14 ..... 1304
Register 54: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18 ..... 1304
Register 55: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C ..... 1304
Register 56: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40 ..... 1307
Register 57: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44 ..... 1307
Register 58: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48 ..... 1307
Register 59: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C ..... 1307
Register 60: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50 ..... 1307
Register 61: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54 ..... 1307
Register 62: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58 ..... 1307
Register 63: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C ..... 1307
Register 64: ADC Peripheral Properties (ADCPP), offset 0xFC0 ..... 1308
Register 65: ADC Peripheral Configuration (ADCPC), offset 0xFC4 ..... 1310
Register 66: ADC Clock Configuration (ADCCC), offset 0xFC8 ..... 1311

**Universal Asynchronous Receivers/Transmitters (UARTs) ..... 1312**

Register 1: UART Data (UARTDR), offset 0x000 ..... 1327
Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 ..... 1329
Register 3: UART Flag (UARTFR), offset 0x018 ..... 1332
Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020 ..... 1335
Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 ..... 1336
Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 ..... 1337
Register 7: UART Line Control (UARTLCRH), offset 0x02C ..... 1338
Register 8: UART Control (UARTCTL), offset 0x030 ..... 1340
Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 ..... 1344
Register 10: UART Interrupt Mask (UARTIM), offset 0x038 ..... 1346
Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C ..... 1350
Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040 ..... 1354
Register 13: UART Interrupt Clear (UARTICR), offset 0x044 ..... 1358
Register 14: UART DMA Control (UARTDMACTL), offset 0x048 ..... 1360

Register 15: UART 9-Bit Self Address (UART9BITADDR), offset 0x0A4 ..... 1361
Register 16: UART 9-Bit Self Address Mask (UART9BITAMASK), offset 0x0A8 ..... 1362
Register 17: UART Peripheral Properties (UARTPP), offset 0xFC0 ..... 1363
Register 18: UART Clock Configuration (UARTCC), offset 0xFC8 ..... 1365
Register 19: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 ..... 1366
Register 20: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 ..... 1367
Register 21: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 ..... 1368
Register 22: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC ..... 1369
Register 23: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 ..... 1370
Register 24: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 ..... 1371
Register 25: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 ..... 1372
Register 26: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC ..... 1373
Register 27: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 ..... 1374
Register 28: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 ..... 1375
Register 29: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 ..... 1376
Register 30: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC ..... 1377

**Quad Synchronous Serial Interface (QSSI) ..... 1378**

Register 1: QSSI Control 0 (SSICR0), offset 0x000 ..... 1397
Register 2: QSSI Control 1 (SSICR1), offset 0x004 ..... 1399
Register 3: QSSI Data (SSIDR), offset 0x008 ..... 1401
Register 4: QSSI Status (SSISR), offset 0x00C ..... 1402
Register 5: QSSI Clock Prescale (SSICPSR), offset 0x010 ..... 1404
Register 6: QSSI Interrupt Mask (SSIIM), offset 0x014 ..... 1405
Register 7: QSSI Raw Interrupt Status (SSIRIS), offset 0x018 ..... 1407
Register 8: QSSI Masked Interrupt Status (SSIMIS), offset 0x01C ..... 1409
Register 9: QSSI Interrupt Clear (SSIICR), offset 0x020 ..... 1411
Register 10: QSSI DMA Control (SSIDMACTL), offset 0x024 ..... 1412
Register 11: QSSI Peripheral Properties (SSIPP), offset 0xFC0 ..... 1413
Register 12: QSSI Clock Configuration (SSICC), offset 0xFC8 ..... 1414
Register 13: QSSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 ..... 1415
Register 14: QSSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 ..... 1416
Register 15: QSSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 ..... 1417
Register 16: QSSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC ..... 1418
Register 17: QSSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 ..... 1419
Register 18: QSSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 ..... 1420
Register 19: QSSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 ..... 1421
Register 20: QSSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC ..... 1422
Register 21: QSSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 ..... 1423
Register 22: QSSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 ..... 1424
Register 23: QSSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 ..... 1425
Register 24: QSSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC ..... 1426

**Inter-Integrated Circuit ($I^2C$) Interface ..... 1427**

Register 1: $I^2C$ Master Slave Address (I2CMSA), offset 0x000 ..... 1454
Register 2: $I^2C$ Master Control/Status (I2CMCS), offset 0x004 ..... 1455
Register 3: $I^2C$ Master Data (I2CMDR), offset 0x008 ..... 1464
Register 4: $I^2C$ Master Timer Period (I2CMTPR), offset 0x00C ..... 1465
Register 5: $I^2C$ Master Interrupt Mask (I2CMIMR), offset 0x010 ..... 1467
Register 6: $I^2C$ Master Raw Interrupt Status (I2CMRIS), offset 0x014 ..... 1470

Register 7: I2C Master Masked Interrupt Status (I2CMMIS), offset 0x018 ..... 1473
Register 8: I2C Master Interrupt Clear (I2CMICR), offset 0x01C ..... 1476
Register 9: I2C Master Configuration (I2CMCR), offset 0x020 ..... 1478
Register 10: I2C Master Clock Low Timeout Count (I2CMCLKOCNT), offset 0x024 ..... 1479
Register 11: I2C Master Bus Monitor (I2CMBMON), offset 0x02C ..... 1480
Register 12: I2C Master Burst Length (I2CMBLEN), offset 0x030 ..... 1481
Register 13: I2C Master Burst Count (I2CMBCNT), offset 0x034 ..... 1482
Register 14: I2C Slave Own Address (I2CSOAR), offset 0x800 ..... 1483
Register 15: I2C Slave Control/Status (I2CSCSR), offset 0x804 ..... 1484
Register 16: I2C Slave Data (I2CSDR), offset 0x808 ..... 1487
Register 17: I2C Slave Interrupt Mask (I2CSIMR), offset 0x80C ..... 1488
Register 18: I2C Slave Raw Interrupt Status (I2CSRIS), offset 0x810 ..... 1490
Register 19: I2C Slave Masked Interrupt Status (I2CSMIS), offset 0x814 ..... 1493
Register 20: I2C Slave Interrupt Clear (I2CSICR), offset 0x818 ..... 1496
Register 21: I2C Slave Own Address 2 (I2CSOAR2), offset 0x81C ..... 1498
Register 22: I2C Slave ACK Control (I2CSACKCTL), offset 0x820 ..... 1499
Register 23: I2C FIFO Data (I2CFIFODATA), offset 0xF00 ..... 1500
Register 24: I2C FIFO Control (I2CFIFOCTL), offset 0xF04 ..... 1502
Register 25: I2C FIFO Status (I2CFIFOSTATUS), offset 0xF08 ..... 1504
Register 26: I2C Peripheral Properties (I2CPP), offset 0xFC0 ..... 1506
Register 27: I2C Peripheral Configuration (I2CPC), offset 0xFC4 ..... 1507

**1-Wire Master Module ..... 1508**

Register 1: 1-Wire Control and Status (ONEWIRECS), offset 0x000 ..... 1520
Register 2: 1-Wire Timing Override (ONEWIRETIM), offset 0x004 ..... 1524
Register 3: 1-Wire Data Write (ONEWIREDATW), offset 0x008 ..... 1525
Register 4: 1-Wire Data Read (ONEWIREDATR), offset 0x00C ..... 1526
Register 5: 1-Wire Interrupt Mask (ONEWIREIM), offset 0x100 ..... 1527
Register 6: 1-Wire Raw Interrupt Status (ONEWIRERIS), offset 0x104 ..... 1529
Register 7: 1-Wire Masked Interrupt Status (ONEWIREMIS), offset 0x108 ..... 1531
Register 8: 1-Wire Interrupt Clear (ONEWIREICR), offset 0x10C ..... 1533
Register 9: 1-Wire µDMA Control (ONEWIREDMA), offset 0x120 ..... 1534
Register 10: 1-Wire Peripheral Properties (ONEWIREPP), offset 0xFC0 ..... 1536

**Controller Area Network (CAN) Module ..... 1537**

Register 1: CAN Control (CANCTL), offset 0x000 ..... 1559
Register 2: CAN Status (CANSTS), offset 0x004 ..... 1561
Register 3: CAN Error Counter (CANERR), offset 0x008 ..... 1564
Register 4: CAN Bit Timing (CANBIT), offset 0x00C ..... 1565
Register 5: CAN Interrupt (CANINT), offset 0x010 ..... 1566
Register 6: CAN Test (CANTST), offset 0x014 ..... 1567
Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018 ..... 1569
Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020 ..... 1570
Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080 ..... 1570
Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024 ..... 1571
Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084 ..... 1571
Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028 ..... 1574
Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088 ..... 1574
Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C ..... 1575

Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C ..... 1575
Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030 ..... 1577
Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090 ..... 1577
Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034 ..... 1578
Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094 ..... 1578
Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038 ..... 1580
Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098 ..... 1580
Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C ..... 1583
Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040 ..... 1583
Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044 ..... 1583
Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048 ..... 1583
Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C ..... 1583
Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0 ..... 1583
Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4 ..... 1583
Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8 ..... 1583
Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100 ..... 1584
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104 ..... 1584
Register 32: CAN New Data 1 (CANNWDA1), offset 0x120 ..... 1585
Register 33: CAN New Data 2 (CANNWDA2), offset 0x124 ..... 1585
Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140 ..... 1586
Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144 ..... 1586
Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160 ..... 1587
Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164 ..... 1587

**Ethernet Controller ..... 1588**

Register 1: Ethernet MAC Configuration (EMACCFG), offset 0x000 ..... 1658
Register 2: Ethernet MAC Frame Filter (EMACFRAMEFLTR), offset 0x004 ..... 1665
Register 3: Ethernet MAC Hash Table High (EMACHASHTBLH), offset 0x008 ..... 1669
Register 4: Ethernet MAC Hash Table Low (EMACHASHTBLL), offset 0x00C ..... 1670
Register 5: Ethernet MAC MII Address (EMACMIIADDR), offset 0x010 ..... 1671
Register 6: Ethernet MAC MII Data Register (EMACMIIDATA), offset 0x014 ..... 1673
Register 7: Ethernet MAC Flow Control (EMACFLOWCTL), offset 0x018 ..... 1674
Register 8: Ethernet MAC VLAN Tag (EMACVLANTG), offset 0x01C ..... 1676
Register 9: Ethernet MAC Status (EMACSTATUS), offset 0x024 ..... 1678
Register 10: Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF), offset 0x028 ..... 1681
Register 11: Ethernet MAC PMT Control and Status Register (EMACPMTCTLSTAT), offset 0x02C ..... 1682
Register 12: Ethernet MAC Raw Interrupt Status (EMACRIS), offset 0x038 ..... 1684
Register 13: Ethernet MAC Interrupt Mask (EMACIM), offset 0x03C ..... 1686
Register 14: Ethernet MAC Address 0 High (EMACADDR0H), offset 0x040 ..... 1687
Register 15: Ethernet MAC Address 0 Low Register (EMACADDR0L), offset 0x044 ..... 1688
Register 16: Ethernet MAC Address 1 High (EMACADDR1H), offset 0x048 ..... 1689
Register 17: Ethernet MAC Address 1 Low (EMACADDR1L), offset 0x04C ..... 1691
Register 18: Ethernet MAC Address 2 High (EMACADDR2H), offset 0x050 ..... 1692
Register 19: Ethernet MAC Address 2 Low (EMACADDR2L), offset 0x054 ..... 1694
Register 20: Ethernet MAC Address 3 High (EMACADDR3H), offset 0x058 ..... 1695
Register 21: Ethernet MAC Address 3 Low (EMACADDR3L), offset 0x05C ..... 1697
Register 22: Ethernet MAC Watchdog Timeout (EMACWDOGTO), offset 0x0DC ..... 1698
Register 23: Ethernet MAC MMC Control (EMACMMCCTRL), offset 0x100 ..... 1699
Register 24: Ethernet MAC MMC Receive Raw Interrupt Status (EMACMMCRXRIS), offset 0x104 ..... 1702

Register 25: Ethernet MAC MMC Transmit Raw Interrupt Status (EMACMMCTXRIS), offset 0x108 ..... 1704
Register 26: Ethernet MAC MMC Receive Interrupt Mask (EMACMMCRXIM), offset 0x10C ..... 1706
Register 27: Ethernet MAC MMC Transmit Interrupt Mask (EMACMMCTXIM), offset 0x110 ..... 1708
Register 28: Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB), offset 0x118 ..... 1710
Register 29: Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL), offset 0x14C ..... 1711
Register 30: Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL), offset 0x150 ..... 1712
Register 31: Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG), offset 0x164 ..... 1713
Register 32: Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB), offset 0x180 ..... 1714
Register 33: Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR), offset 0x194 ..... 1715
Register 34: Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR), offset 0x198 ..... 1716
Register 35: Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI), offset 0x1C4 ..... 1717
Register 36: Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP), offset 0x584 .... 1718
Register 37: Ethernet MAC VLAN Hash Table (EMACVLANHASH), offset 0x588 ..... 1720
Register 38: Ethernet MAC Timestamp Control (EMACTIMSTCTRL), offset 0x700 ..... 1721
Register 39: Ethernet MAC Sub-Second Increment (EMACSUBSECINC), offset 0x704 ..... 1725
Register 40: Ethernet MAC System Time - Seconds (EMACTIMSEC), offset 0x708 ..... 1726
Register 41: Ethernet MAC System Time - Nanoseconds (EMACTIMNANO), offset 0x70C ..... 1727
Register 42: Ethernet MAC System Time - Seconds Update (EMACTIMSECU), offset 0x710 ..... 1728
Register 43: Ethernet MAC System Time - Nanoseconds Update (EMACTIMNANOU), offset 0x714 .... 1729
Register 44: Ethernet MAC Timestamp Addend (EMACTIMADD), offset 0x718 ..... 1730
Register 45: Ethernet MAC Target Time Seconds (EMACTARGSEC), offset 0x71C ..... 1731
Register 46: Ethernet MAC Target Time Nanoseconds (EMACTARGNANO), offset 0x720 ..... 1732
Register 47: Ethernet MAC System Time-Higher Word Seconds (EMACHWORDSEC), offset 0x724 .... 1733
Register 48: Ethernet MAC Timestamp Status (EMACTIMSTAT), offset 0x728 ..... 1734
Register 49: Ethernet MAC PPS Control (EMACPPSCTRL), offset 0x72C ..... 1735
Register 50: Ethernet MAC PPS0 Interval (EMACPPS0INTVL), offset 0x760 ..... 1738
Register 51: Ethernet MAC PPS0 Width (EMACPPS0WIDTH), offset 0x764 ..... 1739
Register 52: Ethernet MAC DMA Bus Mode (EMACDMABUSMOD), offset 0xC00 ..... 1740
Register 53: Ethernet MAC Transmit Poll Demand (EMACTXPOLLD), offset 0xC04 ..... 1744
Register 54: Ethernet MAC Receive Poll Demand (EMACRXPOLLD), offset 0xC08 ..... 1745
Register 55: Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR), offset 0xC0C ..... 1746
Register 56: Ethernet MAC Transmit Descriptor List Address (EMACTXDLADDR), offset 0xC10 ..... 1747
Register 57: Ethernet MAC DMA Interrupt Status (EMACDMARIS), offset 0xC14 ..... 1748
Register 58: Ethernet MAC DMA Operation Mode (EMACDMAOPMODE), offset 0xC18 ..... 1754
Register 59: Ethernet MAC DMA Interrupt Mask Register (EMACDMAIM), offset 0xC1C ..... 1759
Register 60: Ethernet MAC Missed Frame and Buffer Overflow Counter (EMACMFBOC), offset 0xC20 ..... 1762
Register 61: Ethernet MAC Receive Interrupt Watchdog Timer (EMACRXINTWDT), offset 0xC24 ..... 1763
Register 62: Ethernet MAC Current Host Transmit Descriptor (EMACHOSTXDESC), offset 0xC48 ..... 1764
Register 63: Ethernet MAC Current Host Receive Descriptor (EMACHOSRXDESC), offset 0xC4C ..... 1765
Register 64: Ethernet MAC Current Host Transmit Buffer Address (EMACHOSTXBA), offset 0xC50 .... 1766
Register 65: Ethernet MAC Current Host Receive Buffer Address (EMACHOSRXBA), offset 0xC54 ..... 1767

Register 66: Ethernet MAC Peripheral Property Register (EMACPP), offset 0xFC0 ..... 1768
Register 67: Ethernet MAC Peripheral Configuration Register (EMACPC), offset 0xFC4 ..... 1769
Register 68: Ethernet MAC Clock Configuration Register (EMACCC), offset 0xFC8 ..... 1773
Register 69: Ethernet PHY Raw Interrupt Status (EPHYRIS), offset 0xFD0 ..... 1774
Register 70: Ethernet PHY Interrupt Mask (EPHYIM), offset 0xFD4 ..... 1775
Register 71: Ethernet PHY Masked Interrupt Status and Clear (EPHYMISC), offset 0xFD8 ..... 1776
Register 72: Ethernet PHY Basic Mode Control - MR0 (EPHYBMCR), address 0x000 ..... 1777
Register 73: Ethernet PHY Basic Mode Status - MR1 (EPHYBMSR), address 0x001 ..... 1779
Register 74: Ethernet PHY Identifier Register 1 - MR2 (EPHYID1), address 0x002 ..... 1782
Register 75: Ethernet PHY Identifier Register 2 - MR3 (EPHYID2), address 0x003 ..... 1783
Register 76: Ethernet PHY Auto-Negotiation Advertisement - MR4 (EPHYANA), address 0x004 ..... 1784
Register 77: Ethernet PHY Auto-Negotiation Link Partner Ability - MR5 (EPHYANLPA), address 0x005 ..... 1786
Register 78: Ethernet PHY Auto-Negotiation Expansion - MR6 (EPHYANER), address 0x006 ..... 1788
Register 79: Ethernet PHY Auto-Negotiation Next Page TX - MR7 (EPHYANNPTR), address 0x007 .... 1789
Register 80: Ethernet PHY Auto-Negotiation Link Partner Ability Next Page - MR8 (EPHYANLNPTR), address 0x008 ..... 1791
Register 81: Ethernet PHY Configuration 1 - MR9 (EPHYCFG1), address 0x009 ..... 1793
Register 82: Ethernet PHY Configuration 2 - MR10 (EPHYCFG2), address 0x00A ..... 1796
Register 83: Ethernet PHY Configuration 3 - MR11 (EPHYCFG3), address 0x00B ..... 1798
Register 84: Ethernet PHY Register Control - MR13 (EPHYREGCTL), address 0x00D ..... 1800
Register 85: Ethernet PHY Address or Data - MR14 (EPHYADDAR), address 0x00E ..... 1802
Register 86: Ethernet PHY Status - MR16 (EPHYSTS), address 0x010 ..... 1803
Register 87: Ethernet PHY Specific Control- MR17 (EPHYSCR), address 0x011 ..... 1806
Register 88: Ethernet PHY MII Interrupt Status 1 - MR18 (EPHYMISR1), address 0x012 ..... 1809
Register 89: Ethernet PHY MII Interrupt Status 2 - MR19 (EPHYMISR2), address 0x013 ..... 1812
Register 90: Ethernet PHY False Carrier Sense Counter - MR20 (EPHYFCSCR), address 0x014 ..... 1815
Register 91: Ethernet PHY Receive Error Count - MR21 (EPHYRXERCNT), address 0x015 ..... 1816
Register 92: Ethernet PHY BIST Control - MR22 (EPHYBISTCR), address 0x016 ..... 1817
Register 93: Ethernet PHY LED Control - MR24 (EPHYLEDCR), address 0x018 ..... 1820
Register 94: Ethernet PHY Control - MR25 (EPHYCTL), address 0x019 ..... 1821
Register 95: Ethernet PHY 10Base-T Status/Control - MR26 (EPHY10BTSC), address 0x01A ..... 1823
Register 96: Ethernet PHY BIST Control and Status 1 - MR27 (EPHYBICSR1), address 0x01B ..... 1825
Register 97: Ethernet PHY BIST Control and Status 2 - MR28 (EPHYBICSR2), address 0x01C ..... 1826
Register 98: Ethernet PHY Cable Diagnostic Control - MR30 (EPHYCDCR), address 0x01E ..... 1827
Register 99: Ethernet PHY Reset Control - MR31 (EPHYRCR), address 0x01F ..... 1828
Register 100: Ethernet PHY LED Configuration - MR37 (EPHYLEDCFG), address 0x025 ..... 1829

**LCD Controller ..... 1840**

Register 1: LCD PID Register Format (LCDPID), offset 0x000 ..... 1864
Register 2: LCD Control (LCDCTL), offset 0x004 ..... 1865
Register 3: LCD LIDD Control (LCDLIDDCTL), offset 0x00C ..... 1867
Register 4: LCD LIDD CS0 Configuration (LIDDCS0CFG), offset 0x010 ..... 1870
Register 5: LIDD CS0 Read/Write Address (LIDDCS0ADDR), offset 0x014 ..... 1871
Register 6: LIDD CS0 Data Read/Write Initiation (LIDDCS0DATA), offset 0x018 ..... 1872
Register 7: LIDD CS1 Configuration (LIDDCS1CFG), offset 0x01C ..... 1873
Register 8: LIDD CS1 Address Read/Write Initiation (LIDDCS1ADDR), offset 0x020 ..... 1874
Register 9: LIDD CS1 Data Read/Write Initiation (LIDDCS1DATA), offset 0x024 ..... 1875
Register 10: LCD Raster Control (LCDRASTRCTL), offset 0x028 ..... 1876

Register 11: LCD Raster Timing 0 (LCDRASTRTIM0), offset 0x02C ..... 1880
Register 12: LCD Raster Timing 1 (LCDRASTRTIM1), offset 0x030 ..... 1881
Register 13: LCD Raster Timing 2 (LCDRASTRTIM2), offset 0x034 ..... 1882
Register 14: LCD Raster Subpanel Display 1 (LCDRASTRSUBP1), offset 0x038 ..... 1885
Register 15: LCD Raster Subpanel Display 2 (LCDRASTRSUBP2), offset 0x03C ..... 1886
Register 16: LCD DMA Control (LCDDMACTL), offset 0x040 ..... 1887
Register 17: LCD DMA Frame Buffer 0 Base Address (LCDDMABAFB0), offset 0x044 ..... 1889
Register 18: LCD DMA Frame Buffer 0 Ceiling Address (LCDDMACAFB0), offset 0x048 ..... 1890
Register 19: LCD DMA Frame Buffer 1 Base Address (LCDDMABAFB1), offset 0x04C ..... 1891
Register 20: LCD DMA Frame Buffer 1 Ceiling Address (LCDDMACAFB1), offset 0x050 ..... 1892
Register 21: LCD System Configuration Register (LCDSYSCFG), offset 0x054 ..... 1893
Register 22: LCD Interrupt Raw Status and Set Register (LCDRISSET), offset 0x058 ..... 1895
Register 23: LCD Interrupt Status and Clear (LCDMISCLR), offset 0x05C ..... 1898
Register 24: LCD Interrupt Mask (LCDIM), offset 0x060 ..... 1901
Register 25: LCD Interrupt Enable Clear (LCDIENC), offset 0x064 ..... 1904
Register 26: LCD Clock Enable (LCDCLKEN), offset 0x06C ..... 1907
Register 27: LCD Clock Resets (LCDCLKRESET), offset 0x070 ..... 1908

**Analog Comparators ..... 1909**

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000 ..... 1916
Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004 ..... 1917
Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008 ..... 1918
Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010 ..... 1919
Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020 ..... 1920
Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040 ..... 1920
Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x060 ..... 1920
Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x024 ..... 1921
Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044 ..... 1921
Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064 ..... 1921
Register 11: Analog Comparator Peripheral Properties (ACMPPP), offset 0xFC0 ..... 1923

**Pulse Width Modulator (PWM) ..... 1925**

Register 1: PWM Master Control (PWMCTL), offset 0x000 ..... 1939
Register 2: PWM Time Base Sync (PWMSYNC), offset 0x004 ..... 1941
Register 3: PWM Output Enable (PWMENABLE), offset 0x008 ..... 1942
Register 4: PWM Output Inversion (PWMINVERT), offset 0x00C ..... 1944
Register 5: PWM Output Fault (PWMFAULT), offset 0x010 ..... 1946
Register 6: PWM Interrupt Enable (PWMINTEN), offset 0x014 ..... 1948
Register 7: PWM Raw Interrupt Status (PWMRIS), offset 0x018 ..... 1950
Register 8: PWM Interrupt Status and Clear (PWMISC), offset 0x01C ..... 1953
Register 9: PWM Status (PWMSTATUS), offset 0x020 ..... 1956
Register 10: PWM Fault Condition Value (PWMFAULTVAL), offset 0x024 ..... 1958
Register 11: PWM Enable Update (PWMENUPD), offset 0x028 ..... 1960
Register 12: PWM0 Control (PWM0CTL), offset 0x040 ..... 1964
Register 13: PWM1 Control (PWM1CTL), offset 0x080 ..... 1964
Register 14: PWM2 Control (PWM2CTL), offset 0x0C0 ..... 1964
Register 15: PWM3 Control (PWM3CTL), offset 0x100 ..... 1964
Register 16: PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044 ..... 1969
Register 17: PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084 ..... 1969
Register 18: PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4 ..... 1969

Register 19: PWM3 Interrupt and Trigger Enable (PWM3INTEN), offset 0x104 ..... 1969
Register 20: PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048 ..... 1972
Register 21: PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088 ..... 1972
Register 22: PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8 ..... 1972
Register 23: PWM3 Raw Interrupt Status (PWM3RIS), offset 0x108 ..... 1972
Register 24: PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C ..... 1974
Register 25: PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C ..... 1974
Register 26: PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC ..... 1974
Register 27: PWM3 Interrupt Status and Clear (PWM3ISC), offset 0x10C ..... 1974
Register 28: PWM0 Load (PWM0LOAD), offset 0x050 ..... 1976
Register 29: PWM1 Load (PWM1LOAD), offset 0x090 ..... 1976
Register 30: PWM2 Load (PWM2LOAD), offset 0x0D0 ..... 1976
Register 31: PWM3 Load (PWM3LOAD), offset 0x110 ..... 1976
Register 32: PWM0 Counter (PWM0COUNT), offset 0x054 ..... 1977
Register 33: PWM1 Counter (PWM1COUNT), offset 0x094 ..... 1977
Register 34: PWM2 Counter (PWM2COUNT), offset 0x0D4 ..... 1977
Register 35: PWM3 Counter (PWM3COUNT), offset 0x114 ..... 1977
Register 36: PWM0 Compare A (PWM0CMPA), offset 0x058 ..... 1978
Register 37: PWM1 Compare A (PWM1CMPA), offset 0x098 ..... 1978
Register 38: PWM2 Compare A (PWM2CMPA), offset 0x0D8 ..... 1978
Register 39: PWM3 Compare A (PWM3CMPA), offset 0x118 ..... 1978
Register 40: PWM0 Compare B (PWM0CMPB), offset 0x05C ..... 1979
Register 41: PWM1 Compare B (PWM1CMPB), offset 0x09C ..... 1979
Register 42: PWM2 Compare B (PWM2CMPB), offset 0x0DC ..... 1979
Register 43: PWM3 Compare B (PWM3CMPB), offset 0x11C ..... 1979
Register 44: PWM0 Generator A Control (PWM0GENA), offset 0x060 ..... 1980
Register 45: PWM1 Generator A Control (PWM1GENA), offset 0x0A0 ..... 1980
Register 46: PWM2 Generator A Control (PWM2GENA), offset 0x0E0 ..... 1980
Register 47: PWM3 Generator A Control (PWM3GENA), offset 0x120 ..... 1980
Register 48: PWM0 Generator B Control (PWM0GENB), offset 0x064 ..... 1983
Register 49: PWM1 Generator B Control (PWM1GENB), offset 0x0A4 ..... 1983
Register 50: PWM2 Generator B Control (PWM2GENB), offset 0x0E4 ..... 1983
Register 51: PWM3 Generator B Control (PWM3GENB), offset 0x124 ..... 1983
Register 52: PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068 ..... 1986
Register 53: PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8 ..... 1986
Register 54: PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8 ..... 1986
Register 55: PWM3 Dead-Band Control (PWM3DBCTL), offset 0x128 ..... 1986
Register 56: PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C ..... 1987
Register 57: PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC ..... 1987
Register 58: PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC ..... 1987
Register 59: PWM3 Dead-Band Rising-Edge Delay (PWM3DBRISE), offset 0x12C ..... 1987
Register 60: PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070 ..... 1988
Register 61: PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0 ..... 1988
Register 62: PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0 ..... 1988
Register 63: PWM3 Dead-Band Falling-Edge-Delay (PWM3DBFALL), offset 0x130 ..... 1988
Register 64: PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074 ..... 1989
Register 65: PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4 ..... 1989
Register 66: PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4 ..... 1989

Register 67: PWM3 Fault Source 0 (PWM3FLTSRC0), offset 0x134 ..... 1989
Register 68: PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078 ..... 1991
Register 69: PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0x0B8 ..... 1991
Register 70: PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0x0F8 ..... 1991
Register 71: PWM3 Fault Source 1 (PWM3FLTSRC1), offset 0x138 ..... 1991
Register 72: PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C ..... 1994
Register 73: PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC ..... 1994
Register 74: PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC ..... 1994
Register 75: PWM3 Minimum Fault Period (PWM3MINFLTPER), offset 0x13C ..... 1994
Register 76: PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800 ..... 1995
Register 77: PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880 ..... 1995
Register 78: PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900 ..... 1995
Register 79: PWM3 Fault Pin Logic Sense (PWM3FLTSEN), offset 0x980 ..... 1995
Register 80: PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804 ..... 1996
Register 81: PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884 ..... 1996
Register 82: PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904 ..... 1996
Register 83: PWM3 Fault Status 0 (PWM3FLTSTAT0), offset 0x984 ..... 1996
Register 84: PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808 ..... 1998
Register 85: PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888 ..... 1998
Register 86: PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908 ..... 1998
Register 87: PWM3 Fault Status 1 (PWM3FLTSTAT1), offset 0x988 ..... 1998
Register 88: PWM Peripheral Properties (PWMPP), offset 0xFC0 ..... 2001
Register 89: PWM Clock Configuration (PWMCC), offset 0xFC8 ..... 2003

**Quadrature Encoder Interface (QEI) ..... 2004**

Register 1: QEI Control (QEICTL), offset 0x000 ..... 2011
Register 2: QEI Status (QEISTAT), offset 0x004 ..... 2014
Register 3: QEI Position (QEIPOS), offset 0x008 ..... 2015
Register 4: QEI Maximum Position (QEIMAXPOS), offset 0x00C ..... 2016
Register 5: QEI Timer Load (QEILOAD), offset 0x010 ..... 2017
Register 6: QEI Timer (QEITIME), offset 0x014 ..... 2018
Register 7: QEI Velocity Counter (QEICOUNT), offset 0x018 ..... 2019
Register 8: QEI Velocity (QEISPEED), offset 0x01C ..... 2020
Register 9: QEI Interrupt Enable (QEIINTEN), offset 0x020 ..... 2021
Register 10: QEI Raw Interrupt Status (QEIRIS), offset 0x024 ..... 2023
Register 11: QEI Interrupt Status and Clear (QEIISC), offset 0x028 ..... 2025

# Revision History

The revision history table notes changes made between the indicated revisions of the TM4C129XKCZAD data sheet.

**Table 1. Revision History**

| Date | Revision | Description |
|---|---|---|
| June 2014 | 15863.2743 | ■ In ADC chapter, clarified section "Sample and Hold Window Control".<br>■ In SSI chapter:<br>– Noted that during idle periods the transmit data line SSInTx is tristated.<br>– Added clarification to uDMA section about wait states.<br>■ In Ethernet chapter:<br>– Corrected functional description of DMA descriptors.<br>– Added description of Receive Checksum Offload Engine.<br>■ In Electrical Characteristics chapter:<br>– In "Power and Brown-Out Levels" table, updated $V_{POR}$ with characterized values.<br>– In "PIOSC Clock Characteristics" table, clarified $F_{PIOSC}$ values.<br>– In "Low-Frequency Internal Oscillator Characteristics" table, updated $F_{LFIOSC}$ with characterized values.<br>– In "Main Oscillator Input Characteristics" table, removed Pending Characterization footnote.<br>– In "ADC Electrical Characteristics for ADC at 1 Msps" table, updated Max value for $VIN_{CM}$.<br>– In "ADC Electrical Characteristics for ADC at 2 Msps" table, updated values for $VIN_{CM}$, $R_S$, $f_{CONV}$, $T_S$, $T_{LT}$, and the Dynamic Characteristics.<br>– In "Current Consumption" table, updated values that were pending.<br>■ In Package Information appendix:<br>– Moved Orderable Part Numbers table to addendum.<br>– Deleted Packaging Materials section and put into separate packaging document.<br>■ Additional minor data sheet clarifications and corrections. |
| April 2014 | 15802.2729 | ■ In the System Control chapter:<br>– Clarified Hibernation Module reset section.<br>– Added clarifications in Deep-Sleep Mode section.<br>– Added reset for **DID1** register.<br>– Corrected description for **RESC** register, and changed bit 6 HIB Reset to reserved.<br>– Added note to DSSYSDIV bit in **DSCLKCFG** register that values 0x0 and 0x1 should not be used.<br>– Added clarification to FLASHPM bit in **DSLPPWRCFG** register when using the LFIOSC as the Deep-Sleep clock source.<br>– Added four registers, **UNIQUEIDn**, which combined provide a 128-bit unique identifier for each device.<br>■ In the Hibernation chapter, added clarification to **Hibernation Control (HIBCTL)** register about External Wake and Interrupt Pin Enable bit.<br>■ In the Internal Memory chapter, added information on soft reset handling to the EEPROM section.<br>■ In the GPIO chapter:<br>– Replaced table GPIO Pins With Non-Zero Reset Values with table GPIO Pins With Special Considerations.<br>– Added note about preventing false interrupts.<br>■ In the Timer chapter, clarified behavior of TnMIE and TnCINTD bits in the **GPTM Timer n Mode (GPTMTnMR)** registers.<br>■ In the ADC chapter:<br>– Corrected ADC maximum sample rate to two million samples/second. |

**Table 1. Revision History *(continued)***

| Date | Revision | Description |
|---|---|---|
| | | – Corrected figure ADC Input Equivalency.<br>– Removed Dither Enable bit and corrected reset for **ADCCTL** register.<br><br>■ In the UART chapter, clarified that for a receive timeout, the RTIM bit in the **UARTIM** register must be set to see the RTMIS and RTRIS status in the **UARTMIS** and **UARTRIS** registers.<br><br>■ In the SSI chapter:<br>– Clarified Receive FIFO operation.<br>– Clarified DMA operation.<br>– Removed End of Transmission (EOT) bit 4 from **QSSI Control 1 (SSICR1)** register.<br><br>■ In the Ethernet chapter, clarified Initialization and Configuration.<br><br>■ In the USB chapter, added important note that when configured as a bus-powered Device, the USB can operate in SUSPEND mode but produces a higher power draw than required to be compliant.<br><br>■ In the Electrical Characteristics chapter:<br>– In Reset Characteristics table, updated internal reset time parameter values.<br>– In PIOSC Clock Characteristics table, updated parameter values.<br>– In Hibernation External Oscillator (XOSC) Input Characteristics table, removed parameter C0 Crystal shunt capacitance.<br>– Updated Crystal Parameters table.<br>– In Hibernation Module Tamper I/O Characteristics table, updated TMPRn pull-up resistor parameter values.<br>– In Flash Memory Characteristics table, updated $T_{PROG64}$ nom value.<br>– In EEPROM Characteristics table, added values for Read access time and removed EEPROM recovery Power-On Reset delay parameter.<br>– In EPI PSRAM Interface Characteristics table, updated Min value for EPI_CLK period.<br>– In ADC Electrical Characteristics at 1 Msps table, updated values for $V_{ADCIN}$ parameter.<br>– Corrected ADC Input Equivalency diagram.<br>– In Bi- and Quad-SSI Characteristics table, added clarifying footnotes.<br>– Added PWM Timing Characteristics table.<br>– Updated Current Consumption table.<br>– In Peripheral Current Consumption table, updated $I_{DDEMAC}$ Nom value.<br><br>■ In Package Information appendix:<br>– Updated Orderable Devices section to reflect silicon revision 3 part numbers.<br>– Added Device Nomenclature section.<br>– Deleted packaging materials section and put into separate document.<br><br>■ Additional minor data sheet clarifications and corrections. |
| December 2013 | 15638.2711 | ■ Changed NDA (Non-Disclosure Agreement) footer to indicate NDA only applies to USB content.<br><br>■ In System Control chapter:<br>– Added sections "Optional Clock Output Signal (DIVSCLK)" and "Hardware System Service Request".<br>– Removed some registers and bits:<br>• LDORDRIS bit from **Raw Interrupt Status (RIS**) register, LDORDIM bit from **Interrupt Mask Control (IMC)** register, and LDORDMIS bit from **Masked Interrupt Status and Clear (MISC)** register<br>• **Deep Sleep Mode Memory Timing Register 0 for Main Flash and EEPROM (DSMEMTIM0)** register<br>• **LDO Power Calibration (LDOPCAL)** register<br>• **LDO Sleep Power Control (LDOSPCTL)** register<br>• LMINERR bit from **Sleep/Deep-Sleep Power Mode Status (SDPMST)** register<br>– Added LDOSME, TSPDE, PIOSCPDE, SRAMSM, SRAMLPM, FLASHLPM, and LDOSEQ bits in **SYSPROP** register.<br><br>■ In Internal Memory chapter: |

**Table 1. Revision History *(continued)***

| Date | Revision | Description |
|---|---|---|
| | | – Added subsections to "Flash Memory" section about Execute-Only Protection, Read-Only Protection and Permanently Disabling Debug.<br>– Removed `INVPL` bit from **EEPROM Done Status (EEDONE)** register.<br>– Updated table "MEMTIM0 Register Configuration vs. Frequency" with lower wait states, and improved performance values.<br>– Added EEPROM initialization code to "EEPROM Initialization and Configuration" section.<br><br>■ In the ADC chapter:<br>– Added section "Sample and Hold Window Control" and clarified section "Sample Phase Control".<br>– Clarified description of **ADC Sample Phase Control (ADCSPC)** register.<br><br>■ Updated Electrical Characteristics chapter based on characterization information received.<br><br>■ Additional minor data sheet clarifications and corrections. |
| October 2013 | 15440.2698 | Initial release of NDA data sheet. |

# About This Document

This data sheet provides reference information for the TM4C129XKCZAD microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M4F core.

## Audience

This manual is intended for system software developers, hardware designers, and application developers.

## About This Manual

This document is organized into sections that correspond to each major feature.

## Related Documents

The following related documents are available on the Tiva™ C Series web site at http://www.ti.com/tiva-c:

- *Tiva™ C Series TM4C129x Silicon Errata (literature number SPMZ850)*
- *TivaWare™ Boot Loader for C Series User's Guide (literature number SPMU301)*
- *TivaWare™ Graphics Library for C Series User's Guide (literature number SPMU300)*
- *TivaWare™ for C Series Release Notes (literature number SPMU299)*
- *TivaWare™ Peripheral Driver Library for C Series User's Guide (literature number SPMU298)*
- *TivaWare™ USB Library for C Series User's Guide (literature number SPMU297)*
- *Tiva™ C Series TM4C129x ROM User's Guide (literature number SPMU363)*

The following related documents may also be useful:

- *ARM® Cortex™-M4 Errata (literature number SPMZ637)*
- *ARM® Cortex™-M4 Technical Reference Manual*
- *ARM® Debug Interface V5 Architecture Specification*
- *ARM® Embedded Trace Macrocell Architecture Specification*
- Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)*
- *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*

This documentation list was current as of publication date. Please check the web site for additional documentation, including application notes and white papers.

# Documentation Conventions

This document uses the conventions shown in Table 2 on page 56.

**Table 2. Documentation Conventions**

| Notation | Meaning |
|---|---|
| **General Register Notation** | |
| **REGISTER** | APB registers are indicated in uppercase bold. For example, **PBORCTL** is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, **SRCRn** represents any (or all) of the three Software Reset Control registers: **SRCR0, SRCR1** , and **SRCR2**. |
| bit | A single bit in a register. |
| bit field | Two or more consecutive and related bits. |
| offset 0x*nnn* | A hexadecimal increment to a register's address, relative to that module's base address as specified in Table 2-4 on page 113. |
| Register *N* | Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software. |
| reserved | Register bits marked *reserved* are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| *yy:xx* | The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 through 31 in that register. |
| **Register Bit/Field Types** | This value in the register bit diagram indicates whether software running on the controller can change the value of the bit field. |
| RC | Software can read this field. The bit or field is cleared by hardware after reading the bit/field. |
| RO | Software can read this field. Always write the chip reset value. |
| RW | Software can read or write this field. |
| RWC | Software can read or write this field. Writing to it with any value clears the register. |
| RW1C | Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged.<br>This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read. |
| RW1S | Software can read or write a 1 to this field. A write of a 0 to a RW1S bit does not affect the bit value in the register. |
| W1C | Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data.<br>This register is typically used to clear the corresponding bit in an interrupt register. |
| WO | Only a write by software is valid; a read of the register returns no meaningful data. |
| **Register Bit/Field Reset Value** | This value in the register bit diagram shows the bit/field value after any reset, unless noted. |
| 0 | Bit cleared to 0 on chip reset. |
| 1 | Bit set to 1 on chip reset. |
| - | Nondeterministic. |
| **Pin/Signal Notation** | |
| [ ] | Pin alternate function; a pin defaults to the signal without the brackets. |
| pin | Refers to the physical connection on the package. |
| signal | Refers to the electrical signal encoding of a pin. |

**Table 2. Documentation Conventions *(continued)***

| Notation | Meaning |
|---|---|
| assert a signal | Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see SIGNAL and $\overline{\text{SIGNAL}}$ below). |
| deassert a signal | Change the value of the signal from the logically True state to the logically False state. |
| $\overline{\text{SIGNAL}}$ | Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert $\overline{\text{SIGNAL}}$ is to drive it Low; to deassert $\overline{\text{SIGNAL}}$ is to drive it High. |
| SIGNAL | Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert SIGNAL is to drive it High; to deassert SIGNAL is to drive it Low. |
| **Numbers** | |
| X | An uppercase X indicates any of several values is allowed, where X can be any legal pattern. For example, a binary value of 0X00 can be either 0100 or 0000, a hex value of 0xX is 0x0 or 0x1, and so on. |
| 0x | Hexadecimal numbers have a prefix of 0x. For example, 0x00FF is the hexadecimal number FF.<br>All other numbers within register tables are assumed to be binary. Within conceptual information, binary numbers are indicated with a b suffix, for example, 1011b, and decimal numbers are written without a prefix or suffix. |

# 1 Architectural Overview

Texas Instrument's Tiva™ C Series microcontrollers provide designers a high-performance ARM® Cortex™-M-based architecture with a broad set of integration capabilities and a strong ecosystem of software and development tools. Targeting performance and flexibility, the Tiva™ C Series architecture offers a 120 MHz Cortex-M with FPU, a variety of integrated memories and multiple programmable GPIO. Tiva™ C Series devices offer consumers compelling cost-effective solutions by integrating application-specific peripherals and providing a comprehensive library of software tools which minimize board costs and design-cycle time. Offering quicker time-to-market and cost savings, the Tiva™ C Series microcontrollers are the leading choice in high-performance 32-bit applications.

This chapter contains an overview of the Tiva™ C Series microcontrollers as well as details on the TM4C129XKCZAD microcontroller:


- "Tiva™ C Series Overview" on page 58
- "TM4C129XKCZAD Microcontroller Overview" on page 59
- "TM4C129XKCZAD Microcontroller Features" on page 62
- "TM4C129XKCZAD Microcontroller Hardware Details" on page 88
- "Kits" on page 88
- "Support Information" on page 89


## 1.1 Tiva™ C Series Overview

The Tiva™ C Series ARM Cortex-M4 microcontrollers provide top performance and advanced integration. The product family is positioned for cost-effective applications requiring significant control processing and connectivity capabilities such as:

- Industrial communication equipment
- Network appliances, gateways & adapters
- Residential & commercial site monitoring & control
- Remote connectivity & monitoring
- Security/access systems
- HMI control panels
- Factory automation control
- Test and measurement equipment
- Fire & security systems
- Motion control & power inversion
- Medical instrumentation
- Gaming equipment
- Electronic point-of-sale (POS) displays
- Smart Energy/Smart Grid solutions
- Intelligent lighting control
- Vehicle tracking

Tiva™ C Series microcontrollers integrate a large variety of rich communication features to enable a new class of highly connected designs with the ability to allow critical, real-time control between performance and power. The microcontrollers feature integrated communication peripherals along with other high-performance analog and digital functions to offer a strong foundation for many different target uses, spanning from human machine interface to networked system management controllers.

In addition, Tiva™ C Series microcontrollers offer the advantages of ARM's widely available development tools, System-on-Chip (SoC) infrastructure, and a large user community. Additionally, these microcontrollers use ARM's Thumb®-compatible Thumb-2 instruction set to reduce memory requirements and, thereby, cost. Finally, the TM4C129XKCZAD microcontroller is code-compatible to all members of the extensive Tiva™ C Series, providing flexibility to fit precise needs.

Texas Instruments offers a complete solution 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.

## 1.2 TM4C129XKCZAD Microcontroller Overview

The TM4C129XKCZAD microcontroller combines complex integration and high performance with the features shown in Table 1-1.

**Table 1-1. TM4C129XKCZAD Microcontroller Features**

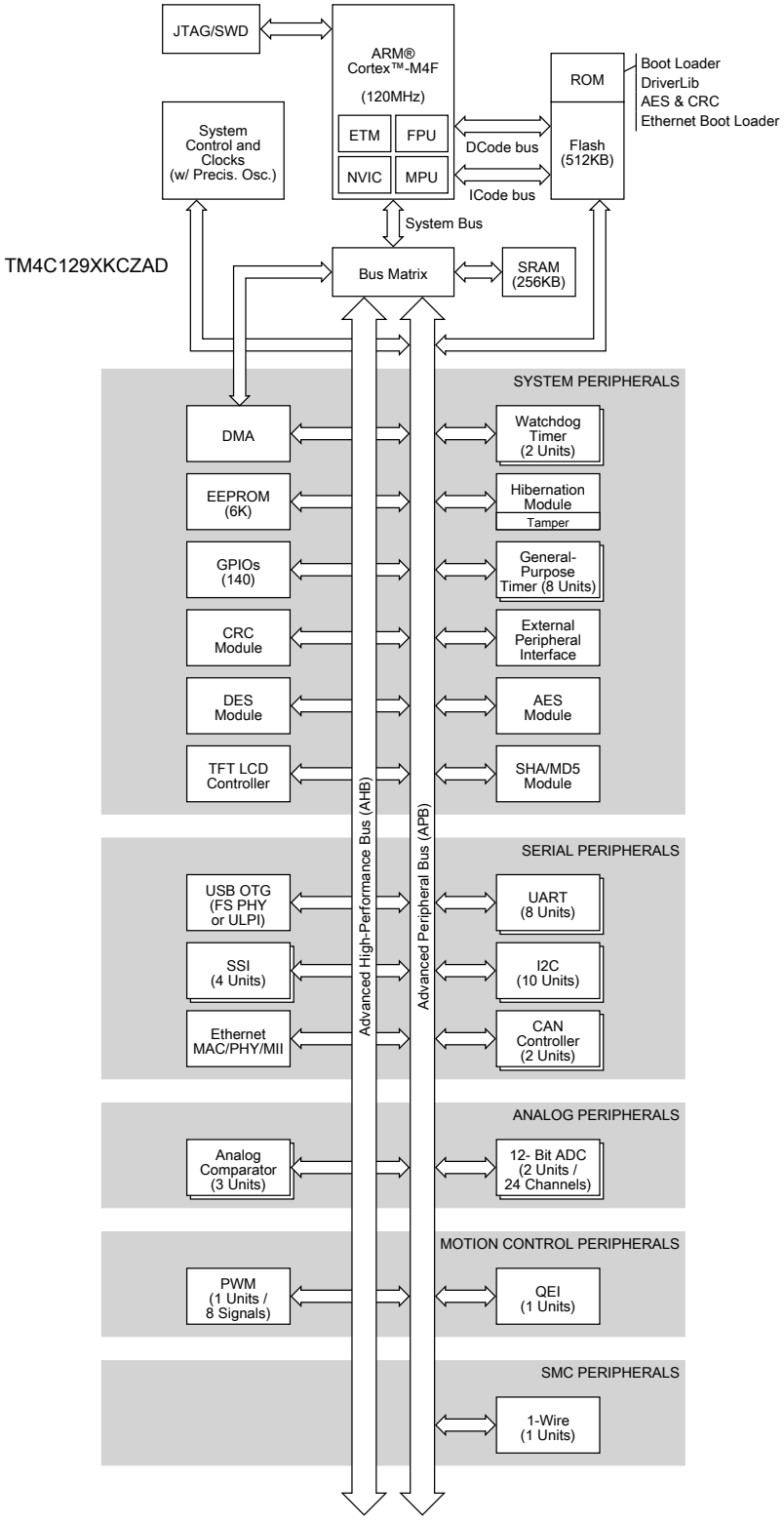
| Feature | Description |
|---|---|
| **Performance** | |
| Core | ARM Cortex-M4F processor core |
| Performance | 120-MHz operation; 150 DMIPS performance |
| Flash | 512 KB Flash memory |
| System SRAM | 256 KB single-cycle System SRAM |
| EEPROM | 6KB of EEPROM |
| Internal ROM | Internal ROM loaded with TivaWare™ for C Series software |
| External Peripheral Interface (EPI) | 8-/16-/32- bit dedicated interface for peripherals and memory |
| **Security** | |
| Cyclical Redundancy Check (CRC) Hardware | 16-/32-bit Hash function that supports four CRC forms |
| Advanced Encryption Standard (AES) | Hardware accelerated data encryption and decryption based on 128-, 192-, and 256-bit keys |
| Data Encryption Standard (DES) | Block cipher implementation with 168-bit effective key length |
| Hardware Accelerated Hash (SHA/MD5) | Advanced hash engine that supports SHA-1, SHA-2 or MD5 Hash computation |
| Tamper | Support for four tamper inputs and configurable tamper event response |
| **Communication Interfaces** | |
| Universal Asynchronous Receivers/Transmitter (UART) | Eight UARTs |
| Quad Synchronous Serial Interface (QSSI) | Four SSI modules with Bi-, Quad- and advanced SSI support |
| Inter-Integrated Circuit ($I^2C$) | Ten $I^2C$ modules with four transmission speeds including high-speed mode |
| Controller Area Network (CAN) | Two CAN 2.0 A/B controllers |
| Ethernet MAC | 10/100 Ethernet MAC with Media Independent Interface (MII) and Reduced MII (RMII) |
| Ethernet PHY | PHY with IEEE 1588 PTP hardware support |
| Universal Serial Bus (USB) | USB 2.0 OTG/Host/Device with ULPI interface option and Link Power Management (LPM) support |
| **System Integration** | |
| Micro Direct Memory Access (µDMA) | ARM® PrimeCell® 32-channel configurable µDMA controller |
| LCD Controller | Configurable LCD controller with passive and active matrix LCD panel support |

**Table 1-1. TM4C129XKCZAD Microcontroller Features *(continued)***

| Feature | Description |
|---|---|
| General-Purpose Timer (GPTM) | Eight 16/32-bit GPTM blocks |
| Watchdog Timer (WDT) | Two watchdog timers |
| Hibernation Module (HIB) | Low-power battery-backed Hibernation module |
| General-Purpose Input/Output (GPIO) | 18 physical GPIO blocks |
| **Advanced Motion Control** | |
| Pulse Width Modulator (PWM) | One PWM module, with four PWM generator blocks and a control block, for a total of 8 PWM outputs. |
| Quadrature Encoder Interface (QEI) | One QEI module |
| **Analog Support** | |
| Analog-to-Digital Converter (ADC) | Two 12-bit ADC modules, each with a maximum sample rate of two million samples/second |
| Analog Comparator Controller | Three independent integrated analog comparators |
| Digital Comparator | 16 digital comparators |
| **System Management** | |
| 1-Wire | One 1-Wire module |
| JTAG and Serial Wire Debug (SWD) | One JTAG module with integrated ARM SWD |
| **Package Information** | |
| Package | 212-ball BGA |
| Operating Range (Ambient) | Industrial (-40°C to 85°C) temperature range<br>Extended (-40°C to 105°C) temperature range |

Figure 1-1 on page 61 shows the features on the TM4C129XKCZAD microcontroller. Note that there are two on-chip buses that connect the core to the peripherals. The Advanced Peripheral Bus (APB) bus is the legacy bus. The Advanced High-Performance Bus (AHB) bus provides better back-to-back access performance than the APB bus.

**Figure 1-1. Tiva™ TM4C129XKCZAD Microcontroller High-Level Block Diagram**



## 1.3 TM4C129XKCZAD Microcontroller Features

The TM4C129XKCZAD microcontroller component features and general function are discussed in more detail in the following section.

### 1.3.1 ARM Cortex-M4F Processor Core

All members of the Tiva™ C Series, including the TM4C129XKCZAD microcontroller, are designed around an ARM Cortex-M processor core. The ARM Cortex-M processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

#### 1.3.1.1 Processor Core (see page 90)

- 32-bit ARM Cortex-M4F architecture optimized for small-footprint embedded applications
- 120-MHz operation; 150 DMIPS performance
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- IEEE754-compliant single-precision Floating-Point Unit (FPU)
- 16-bit SIMD vector processing unit
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing

- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage up to specific frequencies; see “Internal Memory” on page 632 for more information.
- Ultra-low power consumption with integrated sleep modes

#### 1.3.1.2 System Timer (SysTick) (see page 146)

ARM Cortex-M4F 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 the system clock
- A variable rate alarm or signal timer—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/meeting durations

#### 1.3.1.3 Nested Vectored Interrupt Controller (NVIC) (see page 147)

The TM4C129XKCZAD controller includes the ARM Nested Vectored Interrupt Controller (NVIC). The NVIC and Cortex-M4F 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 the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, meaning that back-to-back interrupts can be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 109 interrupts.

- Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining (these values reflect no FPU stacking)
- External non-maskable interrupt signal (NMI) available for immediate execution of NMI handler for safety critical applications
- Dynamically reprioritizable interrupts
- Exceptional interrupt handling via hardware implementation of required register manipulations

#### 1.3.1.4 System Control Block (SCB) (see page 148)

The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

#### 1.3.1.5 Memory Protection Unit (MPU) (see page 148)

The MPU supports the standard ARM7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

#### 1.3.1.6 Floating-Point Unit (FPU) (see page 153)

The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

- 32-bit instructions for single-precision (C float) data-processing operations
- Combined multiply and accumulate instructions for increased precision (Fused MAC)
- Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root
- Hardware support for denormals and all IEEE rounding modes
- 32 dedicated 32-bit single-precision registers, also addressable as 16 double-word registers
- Decoupled three stage pipeline

### 1.3.2 On-Chip Memory

The TM4C129XKCZAD microcontroller is integrated with the following set of on-chip memory and features:

- 256 KB single-cycle SRAM
- 512 KB Flash memory
- 6KB EEPROM
- Internal ROM loaded with TivaWare™ for C Series software:
  - TivaWare™ Peripheral Driver Library
  - TivaWare Boot Loader
  - Advanced Encryption Standard (AES) cryptography tables
  - Cyclic Redundancy Check (CRC) error detection functionality

#### 1.3.2.1 SRAM (see page 634)

The TM4C129XKCZAD microcontroller provides 256 KB of single-cycle on-chip SRAM. The internal SRAM of the device is located at offset 0x2000.0000 of the device memory map.

The SRAM is implemented using four 32-bit wide interleaving SRAM banks (separate SRAM arrays) which allow for increased speed between memory accesses. The SRAM memory provides nearly 2 GB/s memory bandwidth at a 120 MHz clock frequency.

Because read-modify-write (RMW) operations are very time consuming, ARM has introduced *bit-banding* technology in the Cortex-M4F processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

Data can be transferred to and from SRAM by the following masters:

- µDMA
- USB
- LCD Controller

- Ethernet Controller

#### 1.3.2.2 Flash Memory (see page 636)

The TM4C129XKCZAD microcontroller provides 512 KB of on-chip Flash memory. The Flash memory is configured as four banks of 8 K x 128 bits (4 * 128 KB total) which are two-way interleaved. Memory blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

The TM4C129XKCZAD microcontroller provides enhanced performance and power savings by implementation of two sets of instruction prefetch buffers. Each prefetch buffer is 2 x 256 bits and can be combined as a 4 x 256-bit prefetch buffer.

The Flash can also be accessed by the µDMA in Run Mode.

#### 1.3.2.3 ROM (see page 634)

The TM4C129XKCZAD ROM is preprogrammed with the following software and programs:

- TivaWare Peripheral Driver Library
- TivaWare Boot Loader
- Advanced Encryption Standard (AES) cryptography tables
- Cyclic Redundancy Check (CRC) error-detection functionality

The TivaWare Peripheral Driver Library is a royalty-free software library for controlling on-chip peripherals with a boot-loader capability. The library performs both peripheral initialization and control functions, with a choice of polled or interrupt-driven peripheral support. In addition, the library is designed to take full advantage of the stellar interrupt performance of the ARM Cortex-M4F core. No special pragmas or custom assembly code prologue/epilogue functions are required. For applications that require in-field programmability, the royalty-free TivaWare Boot Loader can act as an application loader and support in-field firmware updates.

The Advanced Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government. AES is a strong encryption method with reasonable performance and size. In addition, it is fast in both hardware and software, is fairly easy to implement, and requires little memory. The Texas Instruments encryption package is available with full source code, and is based on Lesser General Public License (LGPL) source. An LGPL means that the code can be used within an application without any copyleft implications for the application (the code does not automatically become open source). Modifications to the package source, however, must be open source.

CRC (Cyclic Redundancy Check) is a technique to validate a span of data has the same contents as when previously checked. This technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that Flash memory contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (for example, XOR all bits) because it catches changes more readily.

**Note:** CRC and AES software programs are available in the TivaWare™ for C Series software for backward-compatibility. A device that has enhanced CRC and AES integrated modules should utilize this hardware for best performance. Please refer to “Cyclical Redundancy

Check (CRC)" on page 979 and "Advance Encryption Standard Accelerator (AES)" on page 988 for more information.

#### 1.3.2.4 EEPROM (see page 647)

The TM4C129XKCZAD microcontroller includes an EEPROM with the following features:

- 6Kbytes of memory accessible as 1536 32-bit words
- 96 blocks of 16 words (64 bytes) each
- Built-in wear leveling
- Access protection per block
- Lock protection option for the whole peripheral as well as per block using 32-bit to 96-bit unlock codes (application selectable)
- Interrupt support for write completion to avoid polling
- Endurance of 500K writes (when writing at fixed offset in every alternate page in circular fashion) to 15M operations (when cycling through two pages ) per each 2-page block.

### 1.3.3 External Peripheral Interface (see page 848)

The External Peripheral Interface (EPI) provides access to external devices using a parallel path. Unlike communications peripherals such as SSI, UART, and $I^2C$, the EPI is designed to act like a bus to external peripherals and memory.

The EPI has the following features:

- 8/16/32-bit dedicated parallel bus for external peripherals and memory
- Memory interface supports contiguous memory access independent of data bus width, thus enabling code execution directly from SDRAM, SRAM and Flash memory
- Blocking and non-blocking reads
- Separates processor from timing details through use of an internal write FIFO
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for read and write
  - Read channel request asserted by programmable levels on the internal Non-Blocking Read FIFO (NBRFIFO)
  - Write channel request asserted by empty on the internal Write FIFO (WFIFO)

The EPI supports three primary functional modes: Synchronous Dynamic Random Access Memory (SDRAM) mode, Traditional Host-Bus mode, and General-Purpose mode. The EPI module also provides custom GPIOs; however, unlike regular GPIOs, the EPI module uses a FIFO in the same way as a communication mechanism and is speed-controlled using clocking.

- Synchronous Dynamic Random Access Memory (SDRAM) mode
  - Supports x16 (single data rate) SDRAM at up to 60 MHz

  - Supports low-cost SDRAMs up to 64 MB (512 megabits)
  - Includes automatic refresh and access to all banks/rows
  - Includes a Sleep/Standby mode to keep contents active with minimal power draw
  - Multiplexed address/data interface for reduced pin count
- Host-Bus mode
  - Traditional x8 and x16 MCU bus interface capabilities
  - Similar device compatibility options as PIC, ATmega, 8051, and others
  - Access to SRAM, NOR Flash memory, and other devices, with up to 1 MB of addressing in non-multiplexed mode and 256 MB in multiplexed mode (512 MB in Host-Bus 16 mode with no byte selects)
  - Support for up to 512 Mb PSRAM in quad chip select mode, with dedicated configuration register read and write enable.
  - Support of both muxed and de-muxed address and data
  - Access to a range of devices supporting the non-address FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) EMPTY and FULL signals
  - Speed controlled, with read and write data wait-state counters
  - Support for read/write burst mode to Host Bus
  - Multiple chip select modes including single, dual, and quad chip selects, with and without ALE
  - External iRDY signal provided for stall capability of reads and writes
  - Manual chip-enable (or use extra address pins)
- General-Purpose mode
  - Wide parallel interfaces for fast communications with CPLDs and FPGAs
  - Data widths up to 32 bits
  - Data rates up to 150 MB/second
  - Optional "address" sizes from 4 bits to 20 bits
  - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
  - 1 to 32 bits, FIFOed with speed control
  - Useful for custom peripherals or for digital data acquisition and actuator controls

### 1.3.4 Cyclical Redundancy Check (CRC) (see page 979)

The TM4C129XKCZAD microcontroller includes a CRC computation module for uses such as message transfer and safety system checks. This module can be used in conjunction with the AES and DES modules. The CRC has the following features:

- Support four major CRC forms:
  - CRC16-CCITT as used by CCITT/ITU X.25
  - CRC16-IBM as used by USB and ANSI
  - CRC32-IEEE as used by IEEE802.3 and MPEG2
  - CRC32C as used by G.Hn
- Allows word and byte feed
- Supports auto-initialization and manual initialization
- Supports MSb and LSb
- Supports CCITT post-processing
- Can be fed by µDMA, Flash memory and code

### 1.3.5 Advanced Encryption Standard (AES) Accelerator (see page 988)

The advanced encryption standard (AES) accelerator module provides hardware-accelerated data encryption and decryption operations based on a binary key. The AES module is a symmetrical cipher modules that supports a 128-bit, 192-bit, or 256-bit key in hardware for both encryption and decryption.

The AES has following features:

- Support for basic AES encrypt and decrypt operations:
  - Galois/Counter Mode (GCM), with basic GHASH operation
  - Counter Mode with CBC-MAC (CCM)
  - XTS Mode
- Availability of the following feedback operating modes:
  - Electronic Code Book Mode (ECB)
  - Cipher Block Chaining Mode (CBC)
  - Counter Mode (CTR)
  - Cipher Feedback Mode (CFB), 128-bit
  - F8 Mode
- Key sizes 128-, 192- and 256-bits

- Support for CBC_MAC and Fedora 9 (F9) authentication modes
- Basic GHASH operation (when selecting no encryption)
- Key scheduling in hardware
- Support for µDMA transfers
- Fully synchronous design

### 1.3.6 Data Encryption Standard (DES) Accelerator (see page 1040)

The DES module provides hardware accelerated data encryption and decryption functions. The module runs either the single DES or the triple DES (3DES) algorithm and supports electronic codebook (ECB), cipher block chaining (CBC), and cipher feedback (CFB) modes of operation.

The DES accelerator includes the following main features:

- DES/3DES encryption and decryption.
- Feedback modes: ECB, CBC, CFB
- Host interrupt or µDMA driven modes of operation. µDMA support for data and context in/result out
- Fully synchronous design
- Internal wide-bus interface

### 1.3.7 Secure Hash Algorithm / Message Digest Algorithm (SHA/MD5) (see page 1069)

The SHA/MD5 module provides hardware-accelerated hash functions and can run:

- MD5 message digest algorithm developed by Ron Rivest in 1991
- SHA-1 algorithm compliant with the FIPS 180-3 standard
- SHA-2 (SHA-224 and SHA-256) algorithm compliant with the FIPS 180-3 standard
- Hash message authentication code (HMAC) operation

The algorithms produce a condensed representation of a message or a data file which can then be used to verify the message integrity.

The SHA/MD5 accelerator module includes the following main features:

- Hashing of 0 to $2^{33}$ - 2 bytes of data (of which $2^{32}$ - 1 bytes are in one pass) using the MD5, SHA-1, SHA-224, or SHA-256 hash algorithm (byte granularity only, no support for bit granularity)
- Automatic HMAC key preprocessing for HMAC keys up to 64 bytes
- Host-assisted HMAC key preprocessing for HMAC keys larger than 64 bytes
- HMAC from precomputes (inner/outer digest) for improved performance on small blocks
- Supports µDMA operation for data and context in/result out transfers
- Supports interrupt to read the digest (signature)

### 1.3.8 Serial Communications Peripherals

The TM4C129XKCZAD controller supports both asynchronous and synchronous serial communications with:

- 10/100 Ethernet MAC with Advanced IEEE 1588 PTP hardware and both Media Independent Interface (MII) and Reduced MII (RMII) support; integrated PHY provided
- Two CAN 2.0 A/B controllers
- USB 2.0 Controller OTG/Host/Device with optional high speed using external PHY through ULPI interface
- Eight UARTs with IrDA, 9-bit and ISO 7816 support.
- 1-Wire Master module with the capability of providing power and data over a single bi-directional wire
- Ten $I^2C$ modules with four transmission speeds including high-speed mode
- Four Quad Synchronous Serial Interface modules (QSSI) with bi- and quad-SSI support

The following sections provide more detail on each of these communications functions.

#### 1.3.8.1 Ethernet MAC and PHY (see page 1588)

The TM4C129XKCZAD Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface with the following features:

- Conforms to the IEEE 802.3 specification
  - 10BASE-T/100BASE-TX IEEE-802.3 compliant
  - Supports 10/100 Mbps data transmission rates
  - Supports full-duplex and half-duplex (CSMA/CD) operation
  - Supports flow control and back pressure
  - Full-featured and enhanced auto-negotiation
  - Supports IEEE 802.1Q VLAN tag detection
- Conforms to IEEE 1588-2002 Timestamp Precision Time Protocol (PTP) protocol and the IEEE 1588-2008 Advanced Timestamp specification
  - Transmit and Receive frame time stamping
  - Precision Time Protocol
  - Flexible pulse per second output
  - Supports coarse and fine correction methods
- Multiple addressing modes
  - Four MAC address filters

  - Programmable 64-bit Hash Filter for multicast address filtering
  - Promiscuous mode support
- Processor offloading
  - Programmable insertion (TX) or deletion (RX) of preamble and start-of-frame data
  - Programmable generation (TX) or deletion (RX) of CRC and pad data
  - IP header and hardware checksum checking (IPv4, IPv6, TCP/UDP/ICMP)
- Highly configurable
  - LED activity selection
  - Supports network statistics with RMON/MIB counters
  - Supports Magic Packet and wakeup frames
- Efficient transfers using integrated Direct Memory Access (DMA)
  - Dual-buffer (ring) or linked-list (chained) descriptors
  - Round-robin or fixed priority arbitration between TX/RX
  - Descriptors support up to 8 kB transfer blocks size
  - Programmable interrupts for flexible system implementation
- Physical media manipulation
  - MDI/MDI-X cross-over support
  - Register-programmable transmit amplitude
  - Automatic polarity correction and 10BASE-T signal reception
- MII and RMII interface support

#### 1.3.8.2 Controller Area Network (CAN) (see page 1537)

Controller Area Network (CAN) is a multicast shared serial-bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically noisy environments and can utilize a differential balanced line like RS-485 or twisted-pair wire. Originally created for automotive purposes, it is now used in many embedded control applications (for example, industrial or medical). Bit rates up to 1 Mbps are possible at network lengths below 40 meters. Decreased bit rates allow longer network distances (for example, 125 Kbps at 500m).

A transmitter sends a message to all CAN nodes (broadcasting). Each node decides on the basis of the identifier received whether it should process the message. The identifier also determines the priority that the message enjoys in competition for bus access. Each CAN message can transmit from 0 to 8 bytes of user information.

The TM4C129XKCZAD microcontroller includes two CAN units with the following features:

- CAN protocol version 2.0 part A/B

- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the CANnTX and CANnRX signals

#### 1.3.8.3 Universal Serial Bus (USB) (see page 1831)

Universal Serial Bus (USB) is a serial bus standard designed to allow peripherals to be connected and disconnected using a standardized interface without rebooting the system.

The TM4C129XKCZAD microcontroller has one USB controller that supports high and full speed multi-point communications and complies with the USB 2.0 standard for high-speed function. The USB controller can have three configurations: USB Device, USB Host, and USB On-The-Go (negotiated on-the-go as host or device when connected to other USB-enabled systems). Support for full-speed communication is provided by using the integrated USB PHY or optionally, a high-speed ULPI interface can communicate to an external PHY.

The USB module has the following features:

- Complies with USB-IF (Implementer's Forum) certification standards
- USB 2.0 high-speed (480 Mbps) operation with the integrated ULPI interface communicating with an external PHY
- Link Power Management support which uses link-state awareness to reduce power usage
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 16 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 7 configurable IN endpoints and 7 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop detection and interrupt
- Integrated USB DMA with bus master capability
  - Up to eight RX Endpoint channels and up to eight TX Endpoint channels are available.
  - Each channel can be separately programmed to operate in different modes
  - Incremental burst transfers of 4-, 8-, 16- or unspecified length supported

### 1.3.8.4 UART (see page 1312)

A Universal Asynchronous Receiver/Transmitter (UART) is an integrated circuit used for RS-232C serial communications, containing a transmitter (parallel-to-serial converter) and a receiver (serial-to-parallel converter), each clocked separately.

The TM4C129XKCZAD microcontroller includes eight fully programmable 16C550-type UARTs. Although the functionality is similar to a 16C550 UART, this UART design is not register compatible. The UART can generate individually masked interrupts from the Rx, Tx, modem flow control, modem status, and error conditions. The module generates a single combined interrupt when any of the interrupts are asserted and are unmasked.

The eight UARTs have the following features:

- Programmable baud-rate generator allowing speeds up to 7.5 Mbps for regular speed (divide by 16) and 15 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing 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/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23 µs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Modem functionality available on the following UARTs:
  - UART0 (modem flow control and modem status)
  - UART1 (modem flow control and modem status)
  - UART2 (modem flow control)

  - UART3 (modem flow control)
  - UART4 (modem flow control)
- EIA-485 9-bit support
- 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 asserted when there is space in the FIFO; burst request asserted at programmed FIFO level
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate baud clock

#### 1.3.8.5 1-Wire Master Module (see page 1508)

1-Wire is a bi-directional serial communication protocol which provides both power and data over a single wire. The 1-Wire Master module can interface with a multiple variety of slaves such as thermometers, mixed-signal devices, memory, and authentication devices.

The TM4C129XKCZAD microcontroller includes a 1-Wire Master module with the following features:

- Support for standard and overdrive speeds, including a late-sample mechanism
- Allows transfers of send, receive and bi-directional bits
- Data size transfers of 1, 2, 3, or 4 bytes with sub-byte search and enumeration support
- Interrupt capability for transaction pacing and line error
- Optional two-wire support for isolated lines and high voltage use
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)

#### 1.3.8.6 $I^2C$ (see page 1427)

The Inter-Integrated Circuit ($I^2C$) bus provides bi-directional data transfer through a two-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 (RAMs and ROMs), 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 manufacture.

Each device on the $I^2C$ bus can be designated as either a master or a slave. $I^2C$ 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 TM4C129XKCZAD microcontroller includes $I^2C$ modules 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 8-entry FIFOs for receive and transmit data
  - FIFOs can be independently assigned to master or slave
- Four transmission speeds:
  - Standard (100 Kbps)
  - Fast-mode (400 Kbps)
  - Fast-mode plus (1 Mbps)
  - High-speed mode (3.33 Mbps)
- Glitch suppression
- SMBus support through software
  - Clock low timeout interrupt
  - Dual slave address capability
  - Quick command capability
- 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
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Ability to execute single data transfers or burst data transfers using the RX and TX FIFOs in the $I^2C$

#### 1.3.8.7 QSSI (see page 1378)

Quad Synchronous Serial Interface (QSSI) is a bi-directional communications interface that converts data between parallel and serial. The QSSI module performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data transmitted to a peripheral device. The QSSI module can be configured as either a master or slave device. As a slave device, the QSSI module can also be configured to disable its output, which allows a master device to be coupled with multiple slave devices. The TX and RX paths are buffered with separate internal FIFOs.

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

The TM4C129XKCZAD microcontroller includes four QSSI modules with the following features:

- Four QSSI channels with Advanced, Bi- and Quad-SSI functionality
- Programmable interface operation for Freescale SPI or Texas Instruments synchronous serial interfaces in Legacy Mode. Support for Freescale interface in Bi- and Quad-SSI mode.
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- 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 FIFO contains 4 entries
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when four or more entries are available to be written in the FIFO
  - Maskable µDMA interrupts for receive and transmit complete
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate baud clock.

### 1.3.9 System Integration

The TM4C129XKCZAD microcontroller provides a variety of standard system functions integrated into the device, including:

- Direct Memory Access Controller (DMA)
- System control and clocks including on-chip precision 16-MHz oscillator
- Eight 32-bit timers (each of which can be configured as two 16-bit timers)

- Lower-power battery-backed Hibernation module
- Real-Time Clock in Hibernation module
- Two Watchdog Timers
  - One timer runs off the main oscillator
  - One timer runs off the precision internal oscillator
- Up to 140 GPIOs, depending on configuration
  - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
  - Independently configurable to 2-, 4-, 8-, 10-, or 12-mA drive capability
  - Up to 4 GPIOs can have 18-mA drive capability

The following sections provide more detail on each of these functions.

#### 1.3.9.1 Direct Memory Access (see page 708)

The TM4C129XKCZAD microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the Cortex-M4F processor, allowing for more efficient use of the processor and the available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. It 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 in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of up to 256 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
  - Flexible channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable priority scheme
  - Optional software-initiated requests for any channel
- Two levels of priority

- Design optimizations for improved bus access performance between µDMA controller and the processor core
  - µDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- 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, half-word, word, or no increment
- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

#### 1.3.9.2 System Control and Clocks (see page 231)

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

- Device identification information: version, part number, SRAM size, Flash memory size, and so on
- Power control
  - On-chip fixed Low Drop-Out (LDO) voltage regulator
  - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
  - Low-power options for microcontroller: Sleep and Deep-Sleep modes with clock gating
  - Low-power options for on-chip modules: software controls shutdown of individual peripherals and memory
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Multiple clock sources for microcontroller system clock. The TM4C129XKCZAD microcontroller is clocked by the system clock (SYSCLK) that is distributed to the processor and integrated peripherals after clock gating. The SYSCLK frequency is based on the frequency of the clock source and a divisor factor. A PLL is provided for the generation of system clock frequencies in excess of the reference clock provided. The reference clocks for the PLL are the PIOSC and the main crystal oscillator. The following clock sources are provided to the TM4C129XKCZAD microcontroller:
  - 16-MHz Precision Oscillator (PIOSC)
  - Main Oscillator (MOSC): A frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins.

  - Low Frequency Internal Oscillator (LFIOSC): On-chip resource used during power-saving modes
  - Hibernate RTC oscillator (RTCOSC) clock that can be configured to be the 32.768-kHz external oscillator source from the Hibernation (HIB) module or the HIB Low Frequency clock source (HIB LFIOSC), which is located within the Hibernation Module.
- Flexible reset sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out reset (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Hibernation module event
  - MOSC failure
- 128-bit unique identifier for individual device identification

#### 1.3.9.3 Programmable Timers (see page 1104)

Programmable timers can be used to count or time external events that drive the Timer input pins. Each 16/32-bit GPTM block provides two 16-bit timers/counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger analog-to-digital (ADC) conversions and DMA transfers.

The General-Purpose Timer Module (GPTM) contains eight 16/32-bit GPTM blocks with the following functional options:

- 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
  - 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
  - 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
  - The System Clock or a global Alternate Clock (ALTCLK) resource can be used as timer clock source. The global ALTCLK can be:
    - PIOSC
    - Hibernation Module Real-time clock output (RTCOSC)

    - Low-frequency internal oscillator (LFIOSC)

- Count up or down
- Sixteen 16/32-bit Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- Timer synchronization allows selected timers to start counting on the same clock cycle
- ADC event trigger
- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode)
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each timer
  - Burst request generated on timer interrupt

#### 1.3.9.4 CCP Pins (see page 1112)

Capture Compare PWM pins (CCP) can be used by the General-Purpose Timer Module to time/count external events using the CCP pin as an input. Alternatively, the GPTM can generate a simple PWM output on the CCP pin.

The TM4C129XKCZAD microcontroller includes 16/32-bit CCP pins that can be programmed to operate in the following modes:

- Capture: The GP Timer is incremented/decremented by programmed events on the CCP input. The GP Timer captures and stores the current timer value when a programmed event occurs.
- Compare: The GP Timer is incremented/decremented by programmed events on the CCP input. The GP Timer compares the current value with a stored value and generates an interrupt when a match occurs.
- PWM: The GP Timer is incremented/decremented by the system clock. A PWM signal is generated based on a match between the counter value and a value stored in a match register and is output on the CCP pin.

#### 1.3.9.5 Hibernation Module (HIB) (see page 563)

The Hibernation module provides logic to switch power off to the main processor and peripherals and to wake on external or time-based events. The Hibernation module includes power-sequencing logic and has the following features:

- 32-bit real-time seconds counter (RTC) with 1/32,768 second resolution and a 15-bit sub-seconds counter
  - 32-bit RTC seconds match register and a 15-bit sub seconds match for timed wake-up and interrupt generation with 1/32,768 second resolution
  - RTC predivider trim for making fine adjustments to the clock rate

- Hardware Calendar Function
  - Year, Month, Day, Day of Week, Hours, Minutes, Seconds
  - Four-year leap compensation
  - 24-hour or AM/PM configuration
- Two mechanisms for power control
  - System power control using discrete external regulator
  - On-chip power control using internal switches under register control
- $V_{DD}$ supplies power when valid, even if $V_{BAT} > V_{DD}$
- Dedicated pin for waking using an external signal
- Capability to configure external reset ($\overline{RST}$) pin and/or up to four GPIO port pins as wake source, with programmable wake level
- Tamper Functionality
  - Support for four tamper inputs
  - Configurable level, weak pull-up, and glitch filter
  - Configurable tamper event response
  - Logging of up to four tamper events
  - Optional BBRAM erase on tamper detection
  - Tamper wake from hibernate capability
  - Hibernation clock input failure detect with a switch to the internal oscillator on detection
- RTC operational and hibernation memory valid as long as $V_{DD}$ or $V_{BAT}$ is valid
- Low-battery detection, signaling, and interrupt generation, with optional wake on low battery
- GPIO pin state can be retained during hibernation
- Clock source from an internal low frequency oscillator (HIB LFIOSC) or a 32.768-kHz external crystal or oscillator
- Sixteen 32-bit words of battery-backed memory to save state during hibernation
- Programmable interrupts for:
  - RTC match
  - External wake
  - Low battery

### 1.3.9.6 Watchdog Timers (see page 1178)

A watchdog timer is used to regain control when a system has failed due to a software error or to the failure of an external device to respond in the expected way. The TM4C129XKCZAD Watchdog Timer can generate an interrupt, a non-maskable interrupt, or a reset when a time-out value is reached. In addition, the Watchdog Timer is ARM FiRM-compliant and can be configured to generate an interrupt to the microcontroller on its first time-out, and to generate a reset signal on its second timeout. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

The TM4C129XKCZAD microcontroller has two Watchdog Timer modules: Watchdog Timer 0 uses the system clock for its timer clock; Watchdog Timer 1 uses the PIOSC as its timer clock. The Watchdog Timer module has the following features:

- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking and optional NMI function
- Lock register protection from runaway software
- Reset generation logic with an enable/disable
- User-enabled stalling when the microcontroller asserts the CPU Halt flag during debug

### 1.3.9.7 Programmable GPIOs (see page 772)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections. The TM4C129XKCZAD GPIO module is comprised of 18 physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-140 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 2028 for the signals available to each GPIO pin).

- Up to 140 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 3.3-V-tolerant in input configuration
- Advanced High Performance Bus accesses all ports:
  - Ports A-H and J; Ports K-N and P-T
- Fast toggle capable of a change every clock cycle for ports on AHB
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
  - Per-pin interrupts available on Port P and Port Q

- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence or a µDMA transfer
- Pin state can be retained during Hibernation mode; pins on port P can be programmed to wake on level in Hibernation mode
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, 6-mA, 8-mA, 10-mA and 12-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for 8-mA, 10-mA and 12-mA pad drive
  - Open drain enables
  - Digital input enables

#### 1.3.9.8 LCD Controller (see page 1840)

The TM4C129XKCZAD controller integrates an LCD Controller which supports the following features:

- Character-based panels
  - Support for two character panels (CS0 and CS1) with independent and programmable bus timing parameters when in asynchronous Hitachi, Motorola, and Intel modes
  - Support for one character panel (CS0) with programmable bus timing parameters when in synchronous Motorola and Intel modes
  - Can be used as a generic 16-bit address/data interleaved MPU bus master with no external stall
- Passive matrix LCD panels
  - Panel types including STN, DSTN, and C-DSTN
  - AC Bias Control
- Active matrix LCD panels
  - Panel types including TN TFT
  - 1, 2, 4, or 8 bits per pixel with palette RAM and 16 or 24 bits per pixel without palette RAM
- OLED Panels
  - Passive Matrix (PM OLED) with frame buffer and controller IC inside the panel
  - Active Matrix (AM OLED)
- Bus mastering capability from either system SRAM or EPI memory.

### 1.3.10 Advanced Motion Control

The TM4C129XKCZAD microcontroller provides motion control functions integrated into the device, including:

- Eight advanced PWM outputs for motion and energy applications
- Four fault inputs to promote low-latency shutdown
- One Quadrature Encoder Input (QEI)

The following provides more detail on these motion control functions.

#### 1.3.10.1 PWM (see page 1925)

The TM4C129XKCZAD microcontroller contains one PWM module, with four PWM generator blocks and a control block, for a total of 8 PWM outputs. Pulse width modulation (PWM) is a powerful technique for digitally encoding analog signal levels. High-resolution counters are used to generate a square wave, and the duty cycle of the square wave is modulated to encode an analog signal. Typical applications include switching power supplies and motor control. The TM4C129XKCZAD PWM module consists of four PWM generator block and a control block. Each PWM generator block contains one timer (16-bit down or up/down counter), two comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector. Each PWM generator block produces two PWM signals that can either be independent signals or a single pair of complementary signals with dead-band delays inserted.

Each PWM generator has the following features:

- Four fault-condition handling inputs to quickly provide low-latency shutdown and prevent damage to the motor being controlled
- One 16-bit counter
  - Runs in Down or Up/Down mode
  - Output frequency controlled by a 16-bit load value
  - Load value updates can be synchronized
  - Produces output signals at zero and load value
- Two PWM comparators
  - Comparator value updates can be synchronized
  - Produces output signals on match
- PWM signal generator
  - Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals
  - Produces two independent PWM signals
- Dead-band generator

  - Produces two PWM signals with programmable dead-band delays suitable for driving a half-H bridge
  - Can be bypassed, leaving input PWM signals unmodified
- Can initiate an ADC sample sequence

The control block determines the polarity of the PWM signals and which signals are passed through to the pins. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins. The PWM control block has the following options:

- PWM output enable of each PWM signal
- Optional output inversion of each PWM signal (polarity control)
- Optional fault handling for each PWM signal
- Synchronization of timers in the PWM generator blocks
- Synchronization of timer/comparator updates across the PWM generator blocks
- Extended PWM synchronization of timer/comparator updates across the PWM generator blocks
- Interrupt status summary of the PWM generator blocks
- Extended PWM fault handling, with multiple fault signals, programmable polarities, and filtering
- PWM generators can be operated independently or synchronized with other generators

#### 1.3.10.2 QEI (see page 2004)

A quadrature encoder, also known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring both the number of pulses and the relative phase of the two signals, the position, direction of rotation, and speed can be tracked. In addition, a third channel, or index signal, can be used to reset the position counter. The TM4C129XKCZAD quadrature encoder with index (QEI) module interprets the code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel. The input frequency of the QEI inputs may be as high as 1/4 of the processor frequency (for example, 30 MHz for a 120-MHz system).

The TM4C129XKCZAD microcontroller includes one QEI module providing control of one motor with the following features:

- Position integrator that tracks the encoder position
- Programmable noise filter on the inputs
- Velocity capture using built-in timer
- The input frequency of the QEI inputs may be as high as 1/4 of the processor frequency (for example, 12.5 MHz for a 50-MHz system)
- Interrupt generation on:
  - Index pulse
  - Velocity-timer expiration

- Direction change
- Quadrature error detection

## 1.3.11 Analog

The TM4C129XKCZAD microcontroller provides analog functions integrated into the device, including:

- Two 12-bit Analog-to-Digital Converters (ADC), with a total of 24 analog input channels and each with a sample rate of two million samples/second
- Three analog comparators
- On-chip voltage regulator

The following provides more detail on these analog functions.

### 1.3.11.1 ADC (see page 1203)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. The TM4C129XKCZAD ADC module features 12-bit conversion resolution and supports 24 input channels plus an internal temperature sensor. Four buffered sample sequencers allow rapid sampling of up to 24 analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. Each ADC module has a digital comparator function that allows the conversion value to be diverted to a comparison unit that provides eight digital comparators.

The TM4C129XKCZAD microcontroller provides two ADC modules, each with the following features:

- 24 shared analog input channels
- 12-bit precision ADC
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of two million samples/second
- Optional, programmable phase delay
- Sample and hold window programmability
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - PWM

  - GPIO

- Hardware averaging of up to 64 samples
- Eight digital comparators
- Converter uses two external reference signals (VREFA+ and VREFA-) or VDDA and GNDA as the voltage reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each sample sequencer
  - ADC module uses burst requests for DMA
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate ADC clock

#### 1.3.11.2 Analog Comparators (see page 1909)

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result. The TM4C129XKCZAD microcontroller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts or triggers to the ADC to cause it to start capturing a sample sequence. The interrupt generation and ADC triggering logic is separate. This means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The TM4C129XKCZAD microcontroller provides three independent integrated analog comparators with the following functions:

- Compare external pin input to external pin input or to internal programmable voltage reference
- Compare a test voltage against any one of the following voltages:
  - An individual external reference voltage
  - A shared single external reference voltage
  - A shared internal reference voltage

### 1.3.12 JTAG and ARM Serial Wire Debug (see page 218)

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and 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 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. Texas Instruments replaces the ARM SW-DP and JTAG-DP with the ARM Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module providing all the normal JTAG debug and test functionality plus real-time access to system

memory without halting the core or requiring any target resident code. The SWJ-DP interface has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, and EXTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trace (DWT) unit for implementing watchpoints, trigger resources, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
  - Embedded Trace Macrocell (ETM) for instruction trace capture
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

### 1.3.13 Packaging and Temperature

- 212-ball RoHS-compliant BGA package
- Industrial (-40°C to 85°C) ambient temperature range
- Extended (-40°C to 105°C) ambient temperature range

## 1.4 TM4C129XKCZAD Microcontroller Hardware Details

Details on the pins and package can be found in the following sections:

- “Pin Diagram” on page 2027
- “Signal Tables” on page 2028
- “Electrical Characteristics” on page 2095
- “Package Information” on page 2183

## 1.5 Kits

The Tiva™ C Series provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware and comprehensive documentation including hardware design files
- Evaluation Kits provide a low-cost and effective means of evaluating TM4C129XKCZAD microcontrollers before purchase

- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box

See the Tiva series website at http://www.ti.com/tiva-c for the latest tools available, or ask your distributor.

## 1.6 Support Information

For support on Tiva™ C Series products, contact the TI Worldwide Product Information Center nearest you.

# 2 The Cortex-M4F Processor

The ARM® Cortex™-M4F processor provides a high-performance, low-cost platform that meets the system requirements of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts. Features include:

- 32-bit ARM® Cortex™-M4F architecture optimized for small-footprint embedded applications
- 120-MHz operation; 150 DMIPS performance
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices, typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- IEEE754-compliant single-precision Floating-Point Unit (FPU)
- 16-bit SIMD vector processing unit
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast digital-signal-processing orientated multiply accumulate
- Saturating arithmetic for signal processing
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage up to specific frequencies; see “Internal Memory” on page 632 for more information.
- Ultra-low power consumption with integrated sleep modes

The Tiva™ C Series microcontrollers builds on this core to bring high-performance 32-bit computing to cost-conscious applications requiring significant control processing and connectivity capabilities such as:

- Low power, hand-held smart devices
- Gaming equipment
- Network appliances and switches
- Home and commercial site monitoring and control
- Electronic point-of-sale (POS) machines
- Motion control
- Medical instrumentation
- Remote connectivity and monitoring
- Test and measurement equipment
- Factory automation
- Fire and security
- Smart Energy/Smart Grid solutions
- Intelligent lighting control
- Transportation

This chapter provides information on the Tiva™ C Series implementation of the Cortex-M4F processor, including the programming model, the memory model, the exception model, fault handling, and power management.

For technical details on the instruction set, see the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)*.

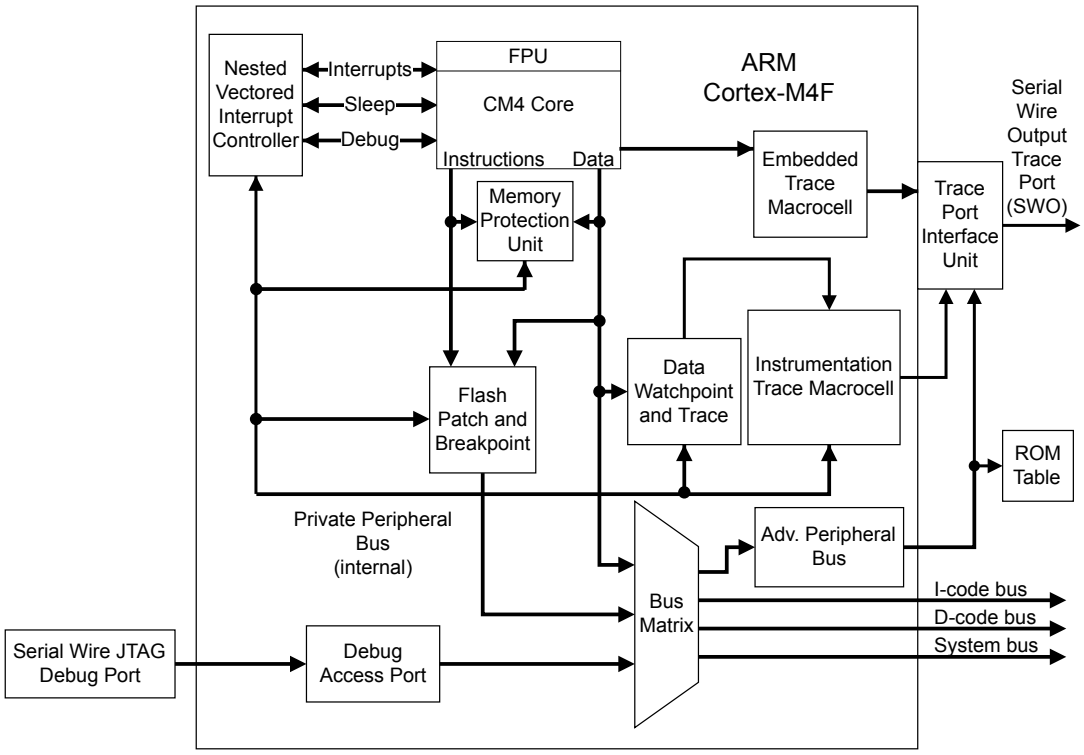
## 2.1 Block Diagram

The Cortex-M4F processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including IEEE754-compliant single-precision floating-point computation, a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M4F processor implements tightly coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M4F processor implements a version of the Thumb® instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M4F instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M4F processor closely integrates a nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The TM4C129XKCZAD NVIC includes a non-maskable interrupt (NMI) and provides eight interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing interrupt latency. The hardware stacking of registers and the ability to suspend load-multiple and store-multiple operations further reduce interrupt latency. Interrupt handlers do not require any assembler stubs which removes code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, including Deep-sleep mode, which enables the entire device to be rapidly powered down.

**Figure 2-1. CPU Block Diagram**




# 2.2 Overview

## 2.2.1 System-Level Interface

The Cortex-M4F processor provides multiple interfaces using AMBA® technology to provide high-speed, low-latency memory accesses. The core supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks, and thread-safe Boolean data handling.

The Cortex-M4F processor has a memory protection unit (MPU) that provides fine-grain memory control, enabling applications to implement security privilege levels and separate code, data and stack on a task-by-task basis.

## 2.2.2 Integrated Configurable Debug

The Cortex-M4F processor implements a complete hardware debug solution, providing high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices. The Tiva™ C Series implementation replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module. See the *ARM® Debug Interface V5 Architecture Specification* for details on SWJ-DP.

For system trace, the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system trace events, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.

The Embedded Trace Macrocell (ETM) delivers unrivaled instruction trace capture in an area smaller than traditional trace units, enabling full instruction trace. For more details on the ARM ETM, see the *ARM® Embedded Trace Macrocell Architecture Specification*.

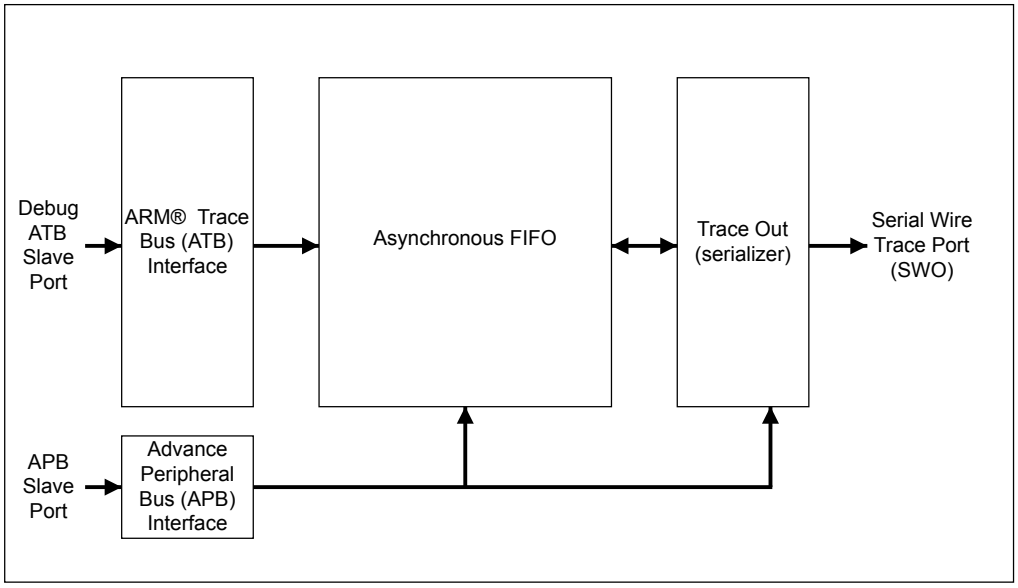
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions for up to eight words of program code in the code memory region. This FPB enables applications stored in a read-only area of Flash memory to be patched in another area of on-chip SRAM or Flash memory. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration.

For more information on the Cortex-M4F debug capabilities, see the*ARM® Debug Interface V5 Architecture Specification*.

### 2.2.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M4F trace data from the ITM, and an off-chip Trace Port Analyzer, as shown in Figure 2-2 on page 93.

**Figure 2-2. TPIU Block Diagram**




### 2.2.4 Cortex-M4F System Component Details

The Cortex-M4F includes the following system components:

- SysTick

  A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter (see “System Timer (SysTick)” on page 146).

- Nested Vectored Interrupt Controller (NVIC)

An embedded interrupt controller that supports low latency interrupt processing (see “Nested Vectored Interrupt Controller (NVIC)” on page 147).

- System Control Block (SCB)

  The programming model interface to the processor. The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions (see “System Control Block (SCB)” on page 148).

- Memory Protection Unit (MPU)

  Improves system reliability by defining the memory attributes for different memory regions. The MPU provides up to eight different regions and an optional predefined background region (see “Memory Protection Unit (MPU)” on page 148).

- Floating-Point Unit (FPU)

  Fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square-root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions (see “Floating-Point Unit (FPU)” on page 153).

## 2.3 Programming Model

This section describes the Cortex-M4F programming model. In addition to the individual core register descriptions, information about the processor modes and privilege levels for software execution and stacks is included.

### 2.3.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M4F has two modes of operation:

- Thread mode

  Used to execute application software. The processor enters Thread mode when it comes out of reset.

- Handler mode

  Used to handle exceptions. When the processor has finished exception processing, it returns to Thread mode.

In addition, the Cortex-M4F has two privilege levels:

- Unprivileged

  In this mode, software has the following restrictions:

  – Limited access to the `MSR` and `MRS` instructions and no use of the `CPS` instruction

  – No access to the system timer, NVIC, or system control block

  – Possibly restricted access to memory or peripherals

- Privileged

  In this mode, software can use all the instructions and has access to all resources.

In Thread mode, the **CONTROL** register (see page 109) controls whether software execution is privileged or unprivileged. In Handler mode, software execution is always privileged.

Only privileged software can write to the **CONTROL** register to change the privilege level for software execution in Thread mode. Unprivileged software can use the SVC instruction to make a supervisor call to transfer control to privileged software.

### 2.3.2 Stacks

The processor uses a full descending stack, meaning that the stack pointer indicates the last stacked item on the memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks: the main stack and the process stack, with a pointer for each held in independent registers (see the **SP** register on page 99).

In Thread mode, the **CONTROL** register (see page 109) controls whether the processor uses the main stack or the process stack. In Handler mode, the processor always uses the main stack. The options for processor operations are shown in Table 2-1 on page 95.

**Table 2-1. Summary of Processor Mode, Privilege Level, and Stack Use**

| Processor Mode | Use | Privilege Level | Stack Used |
|---|---|---|---|
| Thread | Applications | Privileged or unprivileged [a] | Main stack or process stack [a] |
| Handler | Exception handlers | Always privileged | Main stack |

a. See **CONTROL** (page 109).

### 2.3.3 Register Map

Figure 2-3 on page 96 shows the Cortex-M4F register set. Table 2-2 on page 96 lists the Core registers. The core registers are not memory mapped and are accessed by register name, so the base address is n/a (not applicable) and there is no offset.

**Figure 2-3. Cortex-M4F Register Set**

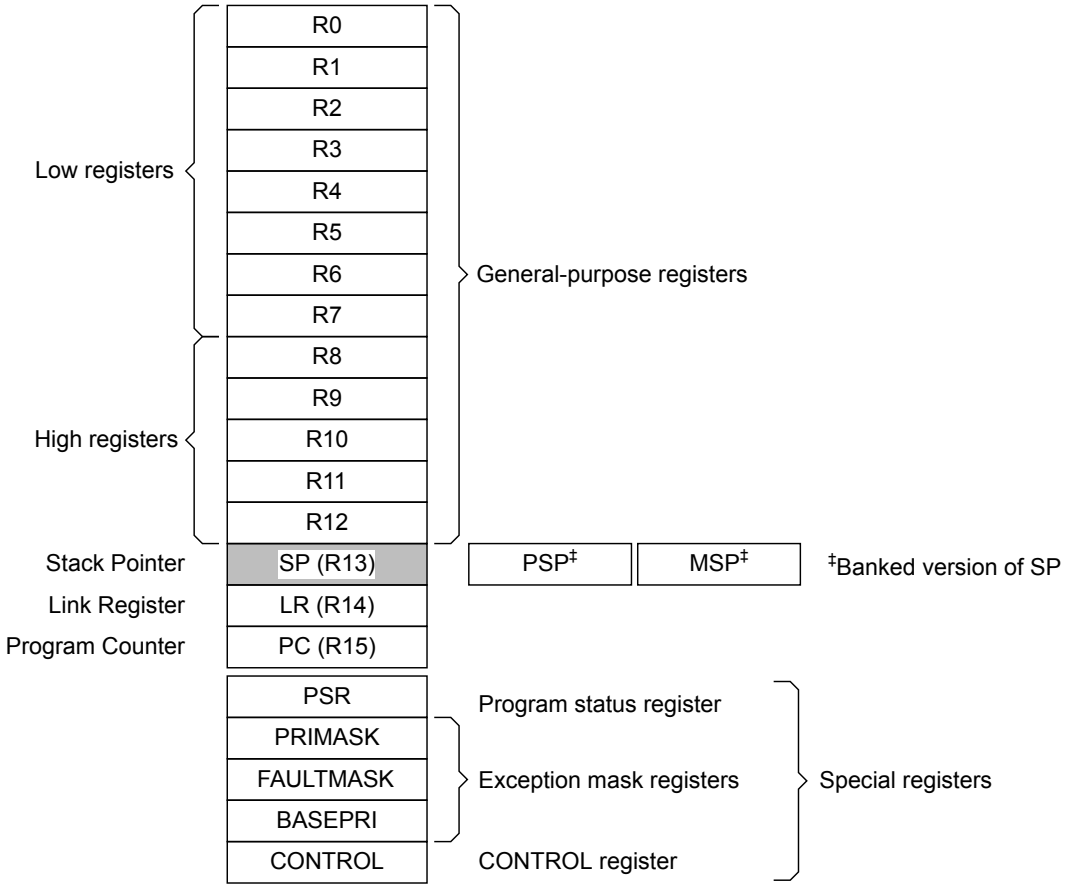



**Table 2-2. Processor Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| - | R0 | RW | - | Cortex General-Purpose Register 0 | 98 |
| - | R1 | RW | - | Cortex General-Purpose Register 1 | 98 |
| - | R2 | RW | - | Cortex General-Purpose Register 2 | 98 |
| - | R3 | RW | - | Cortex General-Purpose Register 3 | 98 |
| - | R4 | RW | - | Cortex General-Purpose Register 4 | 98 |
| - | R5 | RW | - | Cortex General-Purpose Register 5 | 98 |
| - | R6 | RW | - | Cortex General-Purpose Register 6 | 98 |
| - | R7 | RW | - | Cortex General-Purpose Register 7 | 98 |
| - | R8 | RW | - | Cortex General-Purpose Register 8 | 98 |
| - | R9 | RW | - | Cortex General-Purpose Register 9 | 98 |
| - | R10 | RW | - | Cortex General-Purpose Register 10 | 98 |
| - | R11 | RW | - | Cortex General-Purpose Register 11 | 98 |

**Table 2-2. Processor Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| - | R12 | RW | - | Cortex General-Purpose Register 12 | 98 |
| - | SP | RW | - | Stack Pointer | 99 |
| - | LR | RW | 0xFFFF.FFFF | Link Register | 100 |
| - | PC | RW | - | Program Counter | 101 |
| - | PSR | RW | 0x0100.0000 | Program Status Register | 102 |
| - | PRIMASK | RW | 0x0000.0000 | Priority Mask Register | 106 |
| - | FAULTMASK | RW | 0x0000.0000 | Fault Mask Register | 107 |
| - | BASEPRI | RW | 0x0000.0000 | Base Priority Mask Register | 108 |
| - | CONTROL | RW | 0x0000.0000 | Control Register | 109 |
| - | FPSC | RW | - | Floating-Point Status Control | 111 |

### 2.3.4 Register Descriptions

This section lists and describes the Cortex-M4F registers, in the order shown in Figure 2-3 on page 96. The core registers are not memory mapped and are accessed by register name rather than offset.

**Note:** The register type shown in the register descriptions refers to type during program execution in Thread mode and Handler mode. Debug access can differ.

### Register 1: Cortex General-Purpose Register 0 (R0)

### Register 2: Cortex General-Purpose Register 1 (R1)

### Register 3: Cortex General-Purpose Register 2 (R2)

### Register 4: Cortex General-Purpose Register 3 (R3)

### Register 5: Cortex General-Purpose Register 4 (R4)

### Register 6: Cortex General-Purpose Register 5 (R5)

### Register 7: Cortex General-Purpose Register 6 (R6)

### Register 8: Cortex General-Purpose Register 7 (R7)

### Register 9: Cortex General-Purpose Register 8 (R8)

### Register 10: Cortex General-Purpose Register 9 (R9)

### Register 11: Cortex General-Purpose Register 10 (R10)

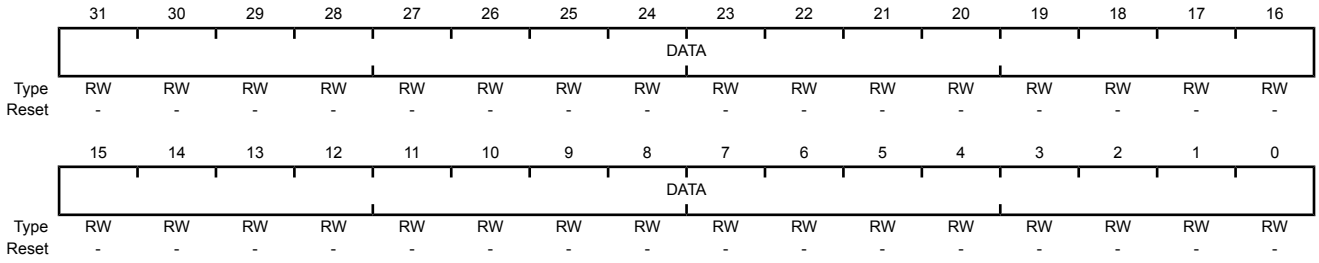
### Register 12: Cortex General-Purpose Register 11 (R11)

### Register 13: Cortex General-Purpose Register 12 (R12)

The **Rn** registers are 32-bit general-purpose registers for data operations and can be accessed from either privileged or unprivileged mode.

Cortex General-Purpose Register 0 (R0)

Type RW, reset -



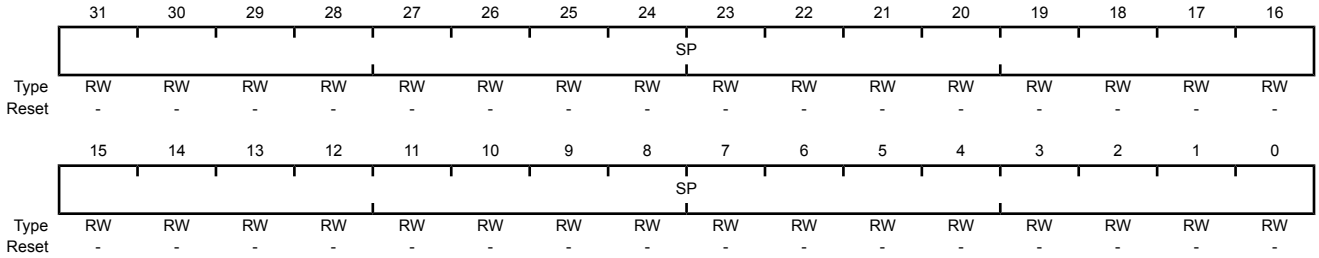

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RW | - | Register data. |

### Register 14: Stack Pointer (SP)

The **Stack Pointer (SP)** is register R13. In Thread mode, the function of this register changes depending on the ASP bit in the **Control Register (CONTROL)** register. When the ASP bit is clear, this register is the **Main Stack Pointer (MSP)**. When the ASP bit is set, this register is the **Process Stack Pointer (PSP)**. On reset, the ASP bit is clear, and the processor loads the **MSP** with the value from address 0x0000.0000. The **MSP** can only be accessed in privileged mode; the **PSP** can be accessed in either privileged or unprivileged mode.

Stack Pointer (SP)

Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SP | RW | - | This field is the address of the stack pointer. |

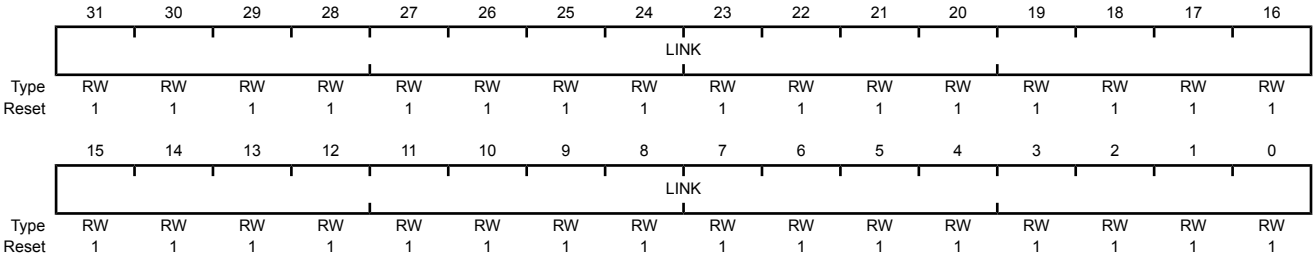
### Register 15: Link Register (LR)

The **Link Register (LR)** is register R14, and it stores the return information for subroutines, function calls, and exceptions. The Link Register can be accessed from either privileged or unprivileged mode.

EXC_RETURN is loaded into the **LR** on exception entry. See Table 2-10 on page 134 for the values and description.

Link Register (LR)

Type RW, reset 0xFFFF.FFFF



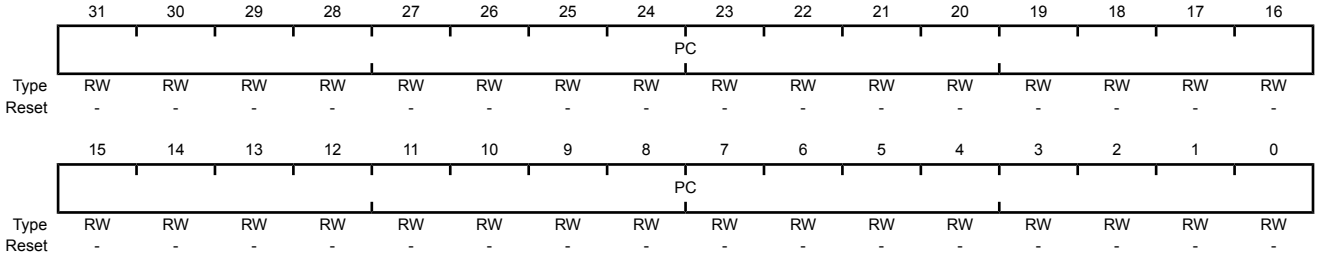

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LINK | RW | 0xFFFF.FFFF | This field is the return address. |

### Register 16: Program Counter (PC)

The **Program Counter (PC)** is register R15, and it contains the current program address. On reset, the processor loads the **PC** with the value of the reset vector, which is at address 0x0000.0004. Bit 0 of the reset vector is loaded into the THUMB bit of the **EPSR** at reset and must be 1. The **PC** register can be accessed in either privileged or unprivileged mode.

Program Counter (PC)

Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PC | RW | - | This field is the current program address. |

### Register 17: Program Status Register (PSR)

**Note:** This register is also referred to as **xPSR**.

The **Program Status Register (PSR)** has three functions, and the register bits are assigned to the different functions:

- **Application Program Status Register (APSR)**, bits 31:27, bits 19:16
- **Execution Program Status Register (EPSR)**, bits 26:24, 15:10
- **Interrupt Program Status Register (IPSR)**, bits 7:0

The **PSR**, **IPSR**, and **EPSR** registers can only be accessed in privileged mode; the **APSR** register can be accessed in either privileged or unprivileged mode.

**APSR** contains the current state of the condition flags from previous instruction executions.

**EPSR** contains the Thumb state bit and the execution state bits for the If-Then (`IT`) instruction or the Interruptible-Continuable Instruction (`ICI`) field for an interrupted load multiple or store multiple instruction. Attempts to read the **EPSR** directly through application software using the `MSR` instruction always return zero. Attempts to write the **EPSR** using the `MSR` instruction in application software are always ignored. Fault handlers can examine the **EPSR** value in the stacked **PSR** to determine the operation that faulted (see “Exception Entry and Return” on page 131).

**IPSR** contains the exception type number of the current Interrupt Service Routine (ISR).

These registers can be accessed individually or as a combination of any two or all three registers, using the register name as an argument to the `MSR` or `MRS` instructions. For example, all of the registers can be read using **PSR** with the `MRS` instruction, or **APSR** only can be written to using **APSR** with the `MSR` instruction. page 102 shows the possible register combinations for the **PSR**. See the `MRS` and `MSR` instruction descriptions in the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information about how to access the program status registers.

**Table 2-3. PSR Register Combinations**

| Register | Type | Combination |
|---|---|---|
| **PSR** | RW[a, b] | **APSR**, **EPSR**, and **IPSR** |
| **IEPSR** | RO | **EPSR** and **IPSR** |
| **IAPSR** | RW[a] | **APSR** and **IPSR** |
| **EAPSR** | RW[b] | **APSR** and **EPSR** |

a. The processor ignores writes to the **IPSR** bits.

b. Reads of the **EPSR** bits return zero, and the processor ignores writes to these bits.

Program Status Register (PSR)

Type RW, reset 0x0100.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N | Z | C | V | Q | ICI / IT | | THUMB | reserved | | | | GE | | | |
| Type | RW | RW | RW | RW | RW | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ICI / IT | | | | | | reserved | | ISRNUM | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | N | RW | 0 | **APSR** Negative or Less Flag<br><br>Value Description<br>1 The previous operation result was negative or less than.<br>0 The previous operation result was positive, zero, greater than, or equal.<br><br>The value of this bit is only meaningful when accessing **PSR** or **APSR**. |
| 30 | Z | RW | 0 | **APSR** Zero Flag<br><br>Value Description<br>1 The previous operation result was zero.<br>0 The previous operation result was non-zero.<br><br>The value of this bit is only meaningful when accessing **PSR** or **APSR**. |
| 29 | C | RW | 0 | **APSR** Carry or Borrow Flag<br><br>Value Description<br>1 The previous add operation resulted in a carry bit or the previous subtract operation did not result in a borrow bit.<br>0 The previous add operation did not result in a carry bit or the previous subtract operation resulted in a borrow bit.<br><br>The value of this bit is only meaningful when accessing **PSR** or **APSR**. |
| 28 | V | RW | 0 | **APSR** Overflow Flag<br><br>Value Description<br>1 The previous operation resulted in an overflow.<br>0 The previous operation did not result in an overflow.<br><br>The value of this bit is only meaningful when accessing **PSR** or **APSR**. |
| 27 | Q | RW | 0 | **APSR** DSP Overflow and Saturation Flag<br><br>Value Description<br>1 DSP Overflow or saturation has occurred when using a SIMD instruction.<br>0 DSP overflow or saturation has not occurred since reset or since the bit was last cleared.<br><br>The value of this bit is only meaningful when accessing **PSR** or **APSR**.<br>This bit is cleared by software using an `MRS` instruction. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 26:25 | ICI / IT | RO | 0x0 | **EPSR** ICI / IT status<br>These bits, along with bits 15:10, contain the Interruptible-Continuable Instruction (`ICI`) field for an interrupted load multiple or store multiple instruction or the execution state bits of the `IT` instruction.<br>When **EPSR** holds the `ICI` execution state, bits 26:25 are zero.<br>The If-Then block contains up to four instructions following an `IT` instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information.<br>The value of this field is only meaningful when accessing **PSR** or **EPSR**. Note that these **EPSR** bits cannot be accessed using `MRS` and `MSR` instructions but the definitions are provided to allow the stacked (E)PSR value to be decoded within an exception handler. |
| 24 | THUMB | RO | 1 | **EPSR** Thumb State<br>This bit indicates the Thumb state and should always be set.<br>The following can clear the `THUMB` bit:<br>■ The `BLX`, `BX` and `POP{PC}` instructions<br>■ Restoration from the stacked **xPSR** value on an exception return<br>■ Bit 0 of the vector value on an exception entry or reset<br>Attempting to execute instructions when this bit is clear results in a fault or lockup. See “Lockup” on page 136 for more information.<br>The value of this bit is only meaningful when accessing **PSR** or **EPSR**. |
| 23:20 | reserved | RO | 0x00 | 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. |
| 19:16 | GE | RW | 0x0 | Greater Than or Equal Flags<br>See the description of the `SEL` instruction in the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information.<br>The value of this field is only meaningful when accessing **PSR** or **APSR**. |

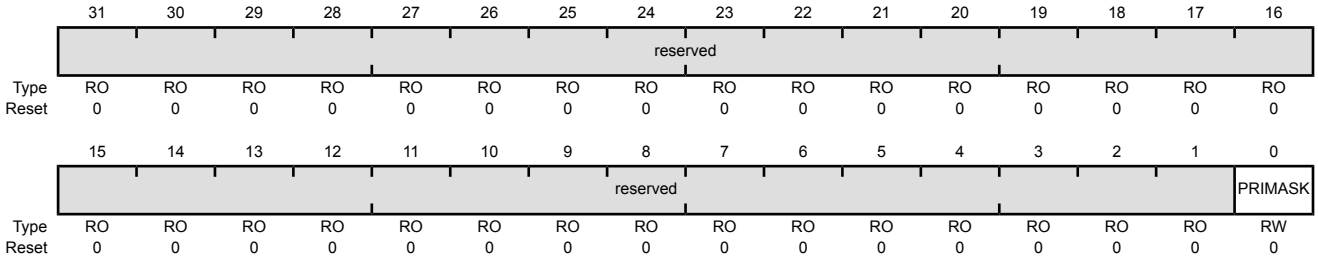
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:10 | ICI / IT | RO | 0x0 | **EPSR** ICI / IT status<br>These bits, along with bits 26:25, contain the Interruptible-Continuable Instruction (`ICI`) field for an interrupted load multiple or store multiple instruction or the execution state bits of the `IT` instruction.<br>When an interrupt occurs during the execution of an `LDM`, `STM`, `PUSH` `POP`, `VLDM`, `VSTM`, `VPUSH`, or `VPOP` instruction, the processor stops the load multiple or store multiple instruction operation temporarily and stores the next register operand in the multiple operation to bits 15:12. After servicing the interrupt, the processor returns to the register pointed to by bits 15:12 and resumes execution of the multiple load or store instruction. When **EPSR** holds the `ICI` execution state, bits 11:10 are zero.<br>The If-Then block contains up to four instructions following a 16-bit `IT` instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information.<br>The value of this field is only meaningful when accessing **PSR** or **EPSR**. |
| 9:8 | reserved | RO | 0x0 | 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. |
| 7:0 | ISRNUM | RO | 0x00 | **IPSR** ISR Number<br>This field contains the exception type number of the current Interrupt Service Routine (ISR).<br>Value / Description<br>0x00 Thread mode<br>0x01 Reserved<br>0x02 NMI<br>0x03 Hard fault<br>0x04 Memory management fault<br>0x05 Bus fault<br>0x06 Usage fault<br>0x07-0x0A Reserved<br>0x0B SVCall<br>0x0C Reserved for Debug<br>0x0D Reserved<br>0x0E PendSV<br>0x0F SysTick<br>0x10 Interrupt Vector 0<br>0x11 Interrupt Vector 1<br>... ...<br>0x81 Interrupt Vector 113<br>See “Exception Types” on page 124 for more information.<br>The value of this field is only meaningful when accessing **PSR** or **IPSR**. |

### Register 18: Priority Mask Register (PRIMASK)

The **PRIMASK** register prevents activation of all exceptions with programmable priority. Reset, non-maskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the **PRIMASK** register, and the CPS instruction may be used to change the value of the **PRIMASK** register. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 124.

Priority Mask Register (PRIMASK)

Type RW, reset 0x0000.0000



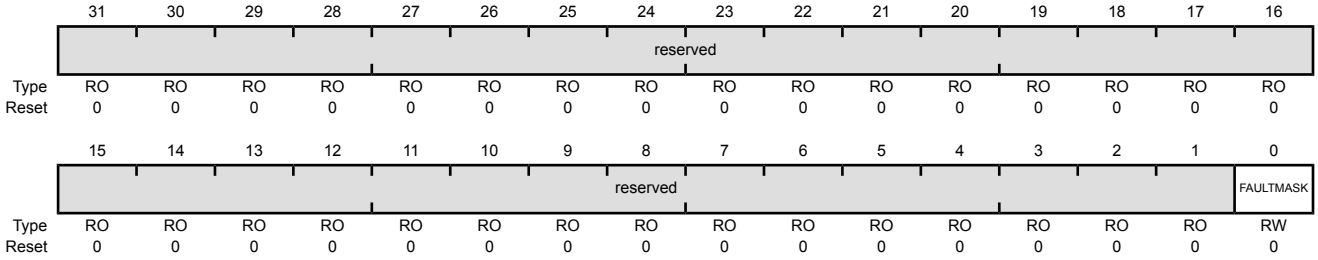

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | PRIMASK | RW | 0 | Priority Mask<br>Value Description<br>1 Prevents the activation of all exceptions with configurable priority.<br>0 No effect. |

### Register 19: Fault Mask Register (FAULTMASK)

The **FAULTMASK** register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the **FAULTMASK** register, and the CPS instruction may be used to change the value of the **FAULTMASK** register. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 124.

Fault Mask Register (FAULTMASK)

Type RW, reset 0x0000.0000



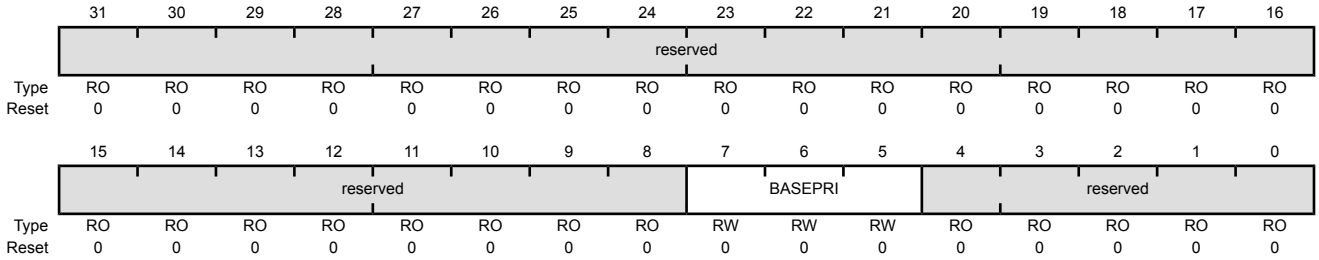

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | FAULTMASK | RW | 0 | Fault Mask<br><br>Value Description<br>1 Prevents the activation of all exceptions except for NMI.<br>0 No effect.<br><br>The processor clears the FAULTMASK bit on exit from any exception handler except the NMI handler. |

### Register 20: Base Priority Mask Register (BASEPRI)

The **BASEPRI** register defines the minimum priority for exception processing. When **BASEPRI** is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the **BASEPRI** value. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. For more information on exception priority levels, see “Exception Types” on page 124.

Base Priority Mask Register (BASEPRI)

Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:5 | BASEPRI | RW | 0x0 | Base Priority<br>Any exception that has a programmable priority level with the same or lower priority as the value of this field is masked. The **PRIMASK** register can be used to mask all exceptions with programmable priority levels. Higher priority exceptions have lower priority levels.<br><br>Value Description<br>0x0 All exceptions are unmasked.<br>0x1 All exceptions with priority level 1-7 are masked.<br>0x2 All exceptions with priority level 2-7 are masked.<br>0x3 All exceptions with priority level 3-7 are masked.<br>0x4 All exceptions with priority level 4-7 are masked.<br>0x5 All exceptions with priority level 5-7 are masked.<br>0x6 All exceptions with priority level 6-7 are masked.<br>0x7 All exceptions with priority level 7 are masked. |
| 4:0 | reserved | RO | 0x0 | 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. |

### Register 21: Control Register (CONTROL)

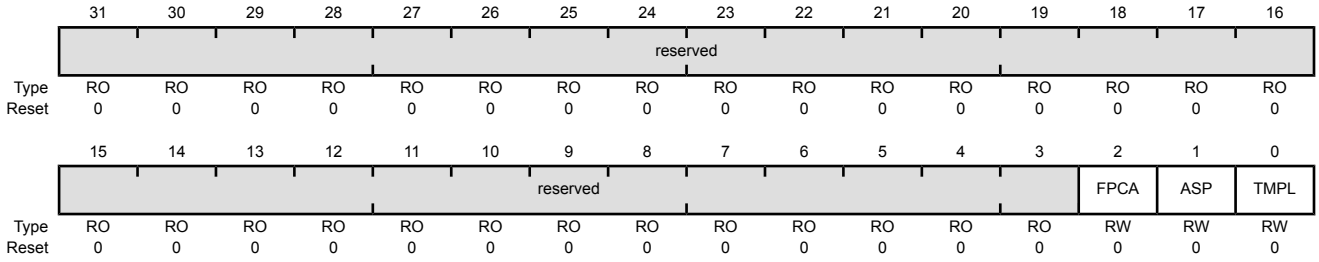
The **CONTROL** register controls the stack used and the privilege level for software execution when the processor is in Thread mode, and indicates whether the FPU state is active. This register is only accessible in privileged mode.

Handler mode always uses the **MSP**, so the processor ignores explicit writes to the ASP bit of the **CONTROL** register when in Handler mode. The exception entry and return mechanisms automatically update the **CONTROL** register based on the EXC_RETURN value (see Table 2-10 on page 134). In an OS environment, threads running in Thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, Thread mode uses the **MSP**. To switch the stack pointer used in Thread mode to the **PSP**, either use the MSR instruction to set the ASP bit, as detailed in the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)*, or perform an exception return to Thread mode with the appropriate EXC_RETURN value, as shown in Table 2-10 on page 134.

**Note:** When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction, ensuring that instructions after the ISB execute use the new stack pointer. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)*.

Control Register (CONTROL)

Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | 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. |
| 2 | FPCA | RW | 0 | Floating-Point Context Active<br><br>Value Description<br>1 Floating-point context active<br>0 No floating-point context active<br><br>The Cortex-M4F uses this bit to determine whether to preserve floating-point state when processing an exception. |

**Important:** Two bits control when FPCA can be enabled: the ASPEN bit in the **Floating-Point Context Control (FPCC)** register and the DISFPCA bit in the **Auxiliary Control (ACTLR)** register.

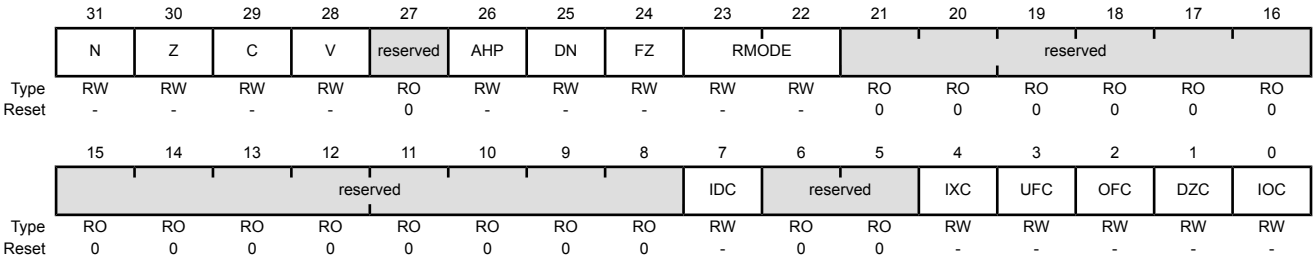
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | ASP | RW | 0 | Active Stack Pointer<br><br>Value Description<br>1 The **PSP** is the current stack pointer.<br>0 The **MSP** is the current stack pointer<br><br>In Handler mode, this bit reads as zero and ignores writes. The Cortex-M4F updates this bit automatically on exception return. |
| 0 | TMPL | RW | 0 | Thread Mode Privilege Level<br><br>Value Description<br>1 Unprivileged software can be executed in Thread mode.<br>0 Only privileged software can be executed in Thread mode. |

### Register 22: Floating-Point Status Control (FPSC)

The **FPSC** register provides all necessary user-level control of the floating-point system.

Floating-Point Status Control (FPSC)

Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | N | RW | - | Negative Condition Code Flag<br>Floating-point comparison operations update this condition code flag. |
| 30 | Z | RW | - | Zero Condition Code Flag<br>Floating-point comparison operations update this condition code flag. |
| 29 | C | RW | - | Carry Condition Code Flag<br>Floating-point comparison operations update this condition code flag. |
| 28 | V | RW | - | Overflow Condition Code Flag<br>Floating-point comparison operations update this condition code flag. |
| 27 | reserved | RO | 0 | 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. |
| 26 | AHP | RW | - | Alternative Half-Precision<br>When set, alternative half-precision format is selected. When clear, IEEE half-precision format is selected.<br>The AHP bit in the **FPDSC** register holds the default value for this bit. |
| 25 | DN | RW | - | Default NaN Mode<br>When set, any operation involving one or more NaNs returns the Default NaN. When clear, NaN operands propagate through to the output of a floating-point operation.<br>The DN bit in the **FPDSC** register holds the default value for this bit. |
| 24 | FZ | RW | - | Flush-to-Zero Mode<br>When set, Flush-to-Zero mode is enabled. When clear, Flush-to-Zero mode is disabled and the behavior of the floating-point system is fully compliant with the IEEE 754 standard.<br>The FZ bit in the **FPDSC** register holds the default value for this bit. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23:22 | RMODE | RW | - | Rounding Mode<br>The specified rounding mode is used by almost all floating-point instructions.<br>The RMODE bit in the **FPDSC** register holds the default value for this bit.<br>Value Description<br>0x0 Round to Nearest (RN) mode<br>0x1 Round towards Plus Infinity (RP) mode<br>0x2 Round towards Minus Infinity (RM) mode<br>0x3 Round towards Zero (RZ) mode |
| 21:8 | reserved | RO | 0x0 | 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. |
| 7 | IDC | RW | - | Input Denormal Cumulative Exception<br>When set, indicates this exception has occurred since 0 was last written to this bit. |
| 6:5 | reserved | RO | 0x0 | 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. |
| 4 | IXC | RW | - | Inexact Cumulative Exception<br>When set, indicates this exception has occurred since 0 was last written to this bit. |
| 3 | UFC | RW | - | Underflow Cumulative Exception<br>When set, indicates this exception has occurred since 0 was last written to this bit. |
| 2 | OFC | RW | - | Overflow Cumulative Exception<br>When set, indicates this exception has occurred since 0 was last written to this bit. |
| 1 | DZC | RW | - | Division by Zero Cumulative Exception<br>When set, indicates this exception has occurred since 0 was last written to this bit. |
| 0 | IOC | RW | - | Invalid Operation Cumulative Exception<br>When set, indicates this exception has occurred since 0 was last written to this bit. |

### 2.3.5 Exceptions and Interrupts

The Cortex-M4F processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See “Exception Entry and Return” on page 131 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” on page 147 for more information.

### 2.3.6 Data Types

The Cortex-M4F supports 32-bit words, 16-bit halfwords, and 8-bit bytes. The processor also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See “Memory Regions, Types and Attributes” on page 117 for more information.

## 2.4 Memory Model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4 GB of addressable memory.

The memory map for the TM4C129XKCZAD controller is provided in Table 2-4 on page 113. In this manual, register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see “Bit-Banding” on page 119).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see “Cortex-M4 Peripherals” on page 145).

**Note:** Within the memory map, attempts to read or write addresses in reserved spaces result in a bus fault. In addition, attempts to write addresses in the flash range also result in a bus fault.

**Table 2-4. Memory Map**

| Start | End | Description | For details, see page ... |
|---|---|---|---|
| **Memory** | | | |
| 0x0000.0000 | 0x0007.FFFF | On-chip Flash | 653 |
| 0x0008.0000 | 0x01FF.FFFF | Reserved | - |
| 0x0200.0000 | 0x02FF.FFFF | On-chip ROM (16 MB) | 634 |
| 0x0300.0000 | 0x1FFF.FFFF | Reserved | - |
| 0x2000.0000 | 0x2006.FFFF | Bit-banded on-chip SRAM | 634 |
| 0x2007.0000 | 0x21FF.FFFF | Reserved | - |
| 0x2200.0000 | 0x2234.FFFF | Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000 | 634 |
| 0x2235.0000 | 0x3FFF.FFFF | Reserved | - |
| **Peripherals** | | | |
| 0x4000.0000 | 0x4000.0FFF | Watchdog timer 0 | 1180 |
| 0x4000.1000 | 0x4000.1FFF | Watchdog timer 1 | 1180 |
| 0x4000.2000 | 0x4000.3FFF | Reserved | - |

**Table 2-4. Memory Map *(continued)***

| Start | End | Description | For details, see page ... |
|---|---|---|---|
| 0x4000.4000 | 0x4000.4FFF | GPIO Port A | 787 |
| 0x4000.5000 | 0x4000.5FFF | GPIO Port B | 787 |
| 0x4000.6000 | 0x4000.6FFF | GPIO Port C | 787 |
| 0x4000.7000 | 0x4000.7FFF | GPIO Port D | 787 |
| 0x4000.8000 | 0x4000.8FFF | SSI0 | 1395 |
| 0x4000.9000 | 0x4000.9FFF | SSI1 | 1395 |
| 0x4000.A000 | 0x4000.AFFF | SSI2 | 1395 |
| 0x4000.B000 | 0x4000.BFFF | SSI3 | 1395 |
| 0x4000.C000 | 0x4000.CFFF | UART0 | 1324 |
| 0x4000.D000 | 0x4000.DFFF | UART1 | 1324 |
| 0x4000.E000 | 0x4000.EFFF | UART2 | 1324 |
| 0x4000.F000 | 0x4000.FFFF | UART3 | 1324 |
| 0x4001.0000 | 0x4001.0FFF | UART4 | 1324 |
| 0x4001.1000 | 0x4001.1FFF | UART5 | 1324 |
| 0x4001.2000 | 0x4001.2FFF | UART6 | 1324 |
| 0x4001.3000 | 0x4001.3FFF | UART7 | 1324 |
| 0x4001.4000 | 0x4001.FFFF | Reserved | - |
| **Peripherals** | | | |
| 0x4002.0000 | 0x4002.0FFF | $I^2C$ 0 | 1451 |
| 0x4002.1000 | 0x4002.1FFF | $I^2C$ 1 | 1451 |
| 0x4002.2000 | 0x4002.2FFF | $I^2C$ 2 | 1451 |
| 0x4002.3000 | 0x4002.3FFF | $I^2C$ 3 | 1451 |
| 0x4002.4000 | 0x4002.4FFF | GPIO Port E | 787 |
| 0x4002.5000 | 0x4002.5FFF | GPIO Port F | 787 |
| 0x4002.6000 | 0x4002.6FFF | GPIO Port G | 787 |
| 0x4002.7000 | 0x4002.7FFF | GPIO Port H | 787 |
| 0x4002.8000 | 0x4002.8FFF | PWM 0 | 1935 |
| 0x4002.9000 | 0x4002.BFFF | Reserved | - |
| 0x4002.C000 | 0x4002.CFFF | QEI0 | 2009 |
| 0x4002.D000 | 0x4002.FFFF | Reserved | - |
| 0x4003.0000 | 0x4003.0FFF | 16/32-bit Timer 0 | 1124 |
| 0x4003.1000 | 0x4003.1FFF | 16/32-bit Timer 1 | 1124 |
| 0x4003.2000 | 0x4003.2FFF | 16/32-bit Timer 2 | 1124 |
| 0x4003.3000 | 0x4003.3FFF | 16/32-bit Timer 3 | 1124 |
| 0x4003.4000 | 0x4003.4FFF | 16/32-bit Timer 4 | 1124 |
| 0x4003.5000 | 0x4003.5FFF | 16/32-bit Timer 5 | 1124 |
| 0x4003.6000 | 0x4003.7FFF | Reserved | - |
| 0x4003.8000 | 0x4003.8FFF | ADC0 | 1224 |
| 0x4003.9000 | 0x4003.9FFF | ADC1 | 1224 |
| 0x4003.A000 | 0x4003.BFFF | Reserved | - |
| 0x4003.C000 | 0x4003.CFFF | Analog Comparators | 1915 |

**Table 2-4. Memory Map *(continued)***

| Start | End | Description | For details, see page ... |
|---|---|---|---|
| 0x4003.D000 | 0x4003.DFFF | GPIO Port J | 787 |
| 0x4003.E000 | 0x4003.FFFF | Reserved | - |
| 0x4004.0000 | 0x4004.0FFF | CAN0 Controller | 1556 |
| 0x4004.1000 | 0x4004.1FFF | CAN1 Controller | 1556 |
| 0x4004.2000 | 0x4004.FFFF | Reserved | - |
| 0x4005.0000 | 0x4005.0FFF | USB | 1833 |
| 0x4005.1000 | 0x4005.7FFF | Reserved | - |
| 0x4005.8000 | 0x4005.8FFF | GPIO Port A (AHB aperture) | 787 |
| 0x4005.9000 | 0x4005.9FFF | GPIO Port B (AHB aperture) | 787 |
| 0x4005.A000 | 0x4005.AFFF | GPIO Port C (AHB aperture) | 787 |
| 0x4005.B000 | 0x4005.BFFF | GPIO Port D (AHB aperture) | 787 |
| 0x4005.C000 | 0x4005.CFFF | GPIO Port E (AHB aperture) | 787 |
| 0x4005.D000 | 0x4005.DFFF | GPIO Port F (AHB aperture) | 787 |
| 0x4005.E000 | 0x4005.EFFF | GPIO Port G (AHB aperture) | 787 |
| 0x4005.F000 | 0x4005.FFFF | GPIO Port H (AHB aperture) | 787 |
| 0x4006.0000 | 0x4006.0FFF | GPIO Port J (AHB aperture) | 787 |
| 0x4006.1000 | 0x4006.1FFF | GPIO Port K (AHB aperture) | 787 |
| 0x4006.2000 | 0x4006.2FFF | GPIO Port L (AHB aperture) | 787 |
| 0x4006.3000 | 0x4006.3FFF | GPIO Port M (AHB aperture) | 787 |
| 0x4006.4000 | 0x4006.4FFF | GPIO Port N (AHB aperture) | 787 |
| 0x4006.5000 | 0x4006.5FFF | GPIO Port P (AHB aperture) | 787 |
| 0x4006.6000 | 0x4006.6FFF | GPIO Port Q (AHB aperture) | 787 |
| 0x4006.7000 | 0x4006.7FFF | GPIO Port R (AHB aperture) | 787 |
| 0x4006.8000 | 0x4006.8FFF | GPIO Port S (AHB aperture) | 787 |
| 0x4006.9000 | 0x4006.9FFF | GPIO Port T (AHB aperture) | 787 |
| 0x4006.A000 | 0x400A.EFFF | Reserved | - |
| 0x400A.F000 | 0x400A.FFFF | EEPROM and Key Locker | 653 |
| 0x400B.0000 | 0x400B.5FFF | Reserved | - |
| 0x400B.6000 | 0x400B.6FFF | 1-Wire Master | 1518 |
| 0x400B.7000 | 0x400B.7FFF | Reserved | - |
| 0x400B.8000 | 0x400B.8FFF | $I^2C$ 8 | 1451 |
| 0x400B.9000 | 0x400B.9FFF | $I^2C$ 9 | 1451 |
| 0x400B.A000 | 0x400B.FFFF | Reserved | - |
| 0x400C.0000 | 0x400C.0FFF | $I^2C$ 4 | 1451 |
| 0x400C.1000 | 0x400C.1FFF | $I^2C$ 5 | 1451 |
| 0x400C.2000 | 0x400C.2FFF | $I^2C$ 6 | 1451 |
| 0x400C.3000 | 0x400C.3FFF | $I^2C$ 7 | 1451 |
| 0x400C.4000 | 0x400C.FFFF | Reserved | - |
| 0x400D.0000 | 0x400D.0FFF | EPI 0 | 889 |
| 0x400D.1000 | 0x400D.FFFF | Reserved | - |
| 0x400E.0000 | 0x400E.0FFF | 16/32-bit Timer 6 | 1124 |

**Table 2-4. Memory Map *(continued)***

| Start | End | Description | For details, see page ... |
|---|---|---|---|
| 0x400E.1000 | 0x400E.1FFF | 16/32-bit Timer 7 | 1124 |
| 0x400E.2000 | 0x400E.BFFF | Reserved | - |
| 0x400E.C000 | 0x400E.CFFF | Ethernet Controller | 1653 |
| 0x400E.D000 | 0x400F.8FFF | Reserved | - |
| 0x400F.9000 | 0x400F.9FFF | System Exception Module | 555 |
| 0x400F.A000 | 0x400F.BFFF | Reserved | - |
| 0x400F.C000 | 0x400F.CFFF | Hibernation Module | 583 |
| 0x400F.D000 | 0x400F.DFFF | Flash memory control | 653 |
| 0x400F.E000 | 0x400F.EFFF | System control | 259 |
| 0x400F.F000 | 0x400F.FFFF | µDMA | 731 |
| 0x4010.0000 | 0x41FF.FFFF | Reserved | - |
| 0x4200.0000 | 0x43FF.FFFF | Bit-banded alias of 0x4000.0000 through 0x400F.FFFF | - |
| 0x4400.0000 | 0x4402.FFFF | Reserved | - |
| 0x4403.0000 | 0x4403.0FFF | CRC and Cryptographic Control Module | - |
| 0x4403.1000 | 0x4403.1FFF | Reserved [4 kB] | - |
| 0x4403.2000 | 0x4403.3FFF | Reserved [8 kB] | - |
| 0x4403.4000 | 0x4403.5FFF | SHA/MD5 | 1083 |
| 0x4403.6000 | 0x4403.7FFF | AES | 1009 |
| 0x4403.8000 | 0x4403.9FFF | DES | 1049 |
| 0x4403.A000 | 0x4403.EFFF | Reserved | - |
| 0x4403.F000 | 0x4403.FFFF | Reserved [4 kB] | - |
| 0x4404.0000 | 0x4404.FFFF | Reserved [64 kB] | - |
| 0x4405.0000 | 0x4405.0FFF | LCD | 1862 |
| 0x4405.1000 | 0x4405.3FFF | Reserved | - |
| 0x4405.4000 | 0x4405.4FFF | EPHY 0 | 1653 |
| 0x4405.5000 | 0x5FFF.FFFF | Reserved | - |
| 0x6000.0000 | 0xDFFF.FFFF | EPI0 mapped peripheral and RAM | - |
| **Private Peripheral Bus** | | | |
| 0xE000.0000 | 0xE000.0FFF | Instrumentation Trace Macrocell (ITM) | 92 |
| 0xE000.1000 | 0xE000.1FFF | Data Watchpoint and Trace (DWT) | 92 |
| 0xE000.2000 | 0xE000.2FFF | Flash Patch and Breakpoint (FPB) | 92 |
| 0xE000.3000 | 0xE000.DFFF | Reserved | - |
| 0xE000.E000 | 0xE000.EFFF | Cortex-M4F Peripherals (SysTick, NVIC, MPU, FPU and SCB) | 157 |
| 0xE000.F000 | 0xE003.FFFF | Reserved | - |
| 0xE004.0000 | 0xE004.0FFF | Trace Port Interface Unit (TPIU) | 93 |
| 0xE004.1000 | 0xE004.1FFF | Embedded Trace Macrocell (ETM) | 92 |
| 0xE004.2000 | 0xFFFF.FFFF | Reserved | - |

### 2.4.1 Memory Regions, Types and Attributes

The memory map and the programming of the MPU split the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

- Normal: The processor can re-order transactions for efficiency and perform speculative reads.
- Device: The processor preserves transaction order relative to other transactions to Device or Strongly Ordered memory.
- Strongly Ordered: The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly Ordered memory mean that the memory system can buffer a write to Device memory but must not buffer a write to Strongly Ordered memory.

An additional memory attribute is Execute Never (XN), which means the processor prevents instruction accesses. A fault exception is generated only on execution of an instruction executed from an XN region.

### 2.4.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing the order does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions (see “Software Ordering of Memory Accesses” on page 118).

However, the memory system does guarantee ordering of accesses to Device and Strongly Ordered memory. For two memory access instructions A1 and A2, if both A1 and A2 are accesses to either Device or Strongly Ordered memory, and if A1 occurs before A2 in program order, A1 is always observed before A2.

### 2.4.3 Behavior of Memory Accesses

Table 2-5 on page 117 shows the behavior of accesses to each region in the memory map. See “Memory Regions, Types and Attributes” on page 117 for more information on memory types and the XN attribute. Tiva™ C Series devices may have reserved memory areas within the address ranges shown below (refer to Table 2-4 on page 113 for more information).

**Table 2-5. Memory Access Behavior**

| Address Range | Memory Region | Memory Type | Execute Never (XN) | Description |
|---|---|---|---|---|
| 0x0000.0000 - 0x1FFF.FFFF | Code | Normal | - | This executable region is for program code. Data can also be stored here. |
| 0x2000.0000 - 0x3FFF.FFFF | SRAM | Normal | - | This executable region is for data. Code can also be stored here. This region includes bit band and bit band alias areas (see Table 2-6 on page 119). |
| 0x4000.0000 - 0x5FFF.FFFF | Peripheral | Device | XN | This region includes bit band and bit band alias areas (see Table 2-7 on page 120). |
| 0x6000.0000 - 0x9FFF.FFFF | External RAM | Normal | - | This executable region is for data. |

**Table 2-5. Memory Access Behavior *(continued)***

| Address Range | Memory Region | Memory Type | Execute Never (XN) | Description |
|---|---|---|---|---|
| 0xA000.0000 - 0xDFFF.FFFF | External device | Device | XN | This region is for external device memory. |
| 0xE000.0000- 0xE00F.FFFF | Private peripheral bus | Strongly Ordered | XN | This region includes the NVIC, system timer, and system control block. |
| 0xE010.0000- 0xFFFF.FFFF | Reserved | - | - | - |

The Code, SRAM, and external RAM regions can hold programs. However, it is recommended that programs always use the Code region because the Cortex-M4F has separate buses that can perform instruction fetches and data accesses simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see “Memory Protection Unit (MPU)” on page 148.

The Cortex-M4F prefetches instructions ahead of execution and speculatively prefetches from branch target addresses.

### 2.4.4 Software Ordering of Memory Accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions for the following reasons:

- The processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- The processor has multiple bus interfaces.
- Memory or devices in the memory map have different wait states.
- Some memory accesses are buffered or speculative.

“Memory System Ordering of Memory Accesses” on page 117 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The Cortex-M4F has the following memory barrier instructions:

- The Data Memory Barrier (`DMB`) instruction ensures that outstanding memory transactions complete before subsequent memory transactions.
- The Data Synchronization Barrier (`DSB`) instruction ensures that outstanding memory transactions complete before subsequent instructions execute.
- The Instruction Synchronization Barrier (`ISB`) instruction ensures that the effect of all completed memory transactions is recognizable by subsequent instructions.

Memory barrier instructions can be used in the following situations:

- MPU programming
  - If the MPU settings are changed and the change must be effective on the very next instruction, use a `DSB` instruction to ensure the effect of the MPU takes place immediately at the end of context switching.

– Use an ISB instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an ISB instruction is not required.

- Vector table

  If the program changes an entry in the vector table and then enables the corresponding exception, use a DMB instruction between the operations. The DMB instruction ensures that if the exception is taken immediately after being enabled, the processor uses the new exception vector.

- Self-modifying code

  If a program contains self-modifying code, use an ISB instruction immediately after the code modification in the program. The ISB instruction ensures subsequent instruction execution uses the updated program.

- Memory map switching

  If the system contains a memory map switching mechanism, use a DSB instruction after switching the memory map in the program. The DSB instruction ensures subsequent instruction execution uses the updated memory map.

- Dynamic exception priority change

  When an exception priority has to change when the exception is pending or active, use DSB instructions after the change. The change then takes effect on completion of the DSB instruction.

Memory accesses to Strongly Ordered memory, such as the System Control Block, do not require the use of DMB instructions.

For more information on the memory barrier instructions, see the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)*.

### 2.4.5 Bit-Banding

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1 MB of the SRAM and peripheral memory regions. Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region, as shown in Table 2-6 on page 119. Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region, as shown in Table 2-7 on page 120. For the specific address range of the bit-band regions, see Table 2-4 on page 113.

**Note:** A word access to the SRAM or the peripheral bit-band alias region maps to a single bit in the SRAM or peripheral bit-band region.

A word access to a bit band address results in a word access to the underlying memory, and similarly for halfword and byte accesses. This allows bit band accesses to match the access requirements of the underlying peripheral.

**Table 2-6. SRAM Memory Bit-Banding Regions**

| Address Range | | Memory Region | Instruction and Data Accesses |
|---|---|---|---|
| Start | End | | |
| 0x2000.0000 | 0x2006.FFFF | SRAM bit-band region | Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias. |

**Table 2-6. SRAM Memory Bit-Banding Regions *(continued)***

| Address Range | | Memory Region | Instruction and Data Accesses |
|---|---|---|---|
| Start | End | | |
| 0x2200.0000 | 0x2234.FFFF | SRAM bit-band alias | Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped. |

**Table 2-7. Peripheral Memory Bit-Banding Regions**

| Address Range | | Memory Region | Instruction and Data Accesses |
|---|---|---|---|
| Start | End | | |
| 0x4000.0000 | 0x400F.FFFF | Peripheral bit-band region | Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias. |
| 0x4200.0000 | 0x43FF.FFFF | Peripheral bit-band alias | Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted. |

The following formula shows how the alias region maps onto the bit-band region:

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)
```

```
bit_word_addr = bit_band_base + bit_word_offset
```

where:

bit_word_offset
The position of the target bit in the bit-band memory region.

bit_word_addr
The address of the word in the alias memory region that maps to the targeted bit.

bit_band_base
The starting address of the alias region.

byte_offset
The number of the byte in the bit-band region that contains the targeted bit.

bit_number
The bit position, 0-7, of the targeted bit.

Figure 2-4 on page 121 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FF.FFE0 maps to bit 0 of the bit-band byte at 0x200F.FFFF:

  ```
  0x23FF.FFE0 = 0x2200.0000 + (0x000F.FFFF*32) + (0*4)
  ```

- The alias word at 0x23FF.FFFC maps to bit 7 of the bit-band byte at 0x200F.FFFF:

  ```
  0x23FF.FFFC = 0x2200.0000 + (0x000F.FFFF*32) + (7*4)
  ```

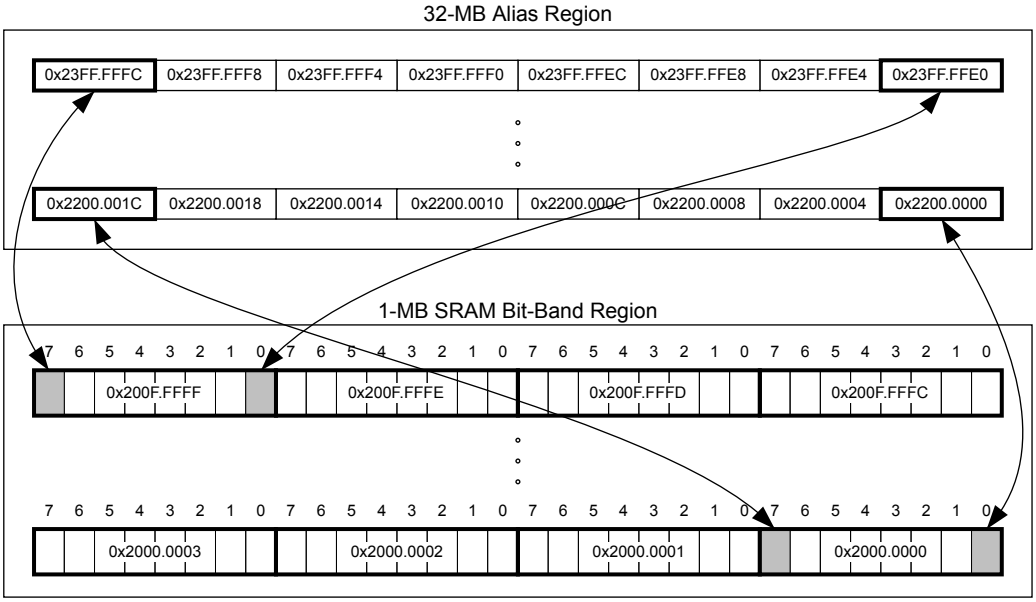
- The alias word at 0x2200.0000 maps to bit 0 of the bit-band byte at 0x2000.0000:

```
0x2200.0000 = 0x2200.0000 + (0*32) + (0*4)
```

- The alias word at 0x2200.001C maps to bit 7 of the bit-band byte at 0x2000.0000:

```
0x2200.001C = 0x2200.0000+ (0*32) + (7*4)
```

**Figure 2-4. Bit-Band Mapping**




### 2.4.5.1 Directly Accessing an Alias Region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit 0 of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit 0 set writes a 1 to the bit-band bit, and writing a value with bit 0 clear writes a 0 to the bit-band bit.

Bits 31:1 of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

When reading a word in the alias region, 0x0000.0000 indicates that the targeted bit in the bit-band region is clear and 0x0000.0001 indicates that the targeted bit in the bit-band region is set.

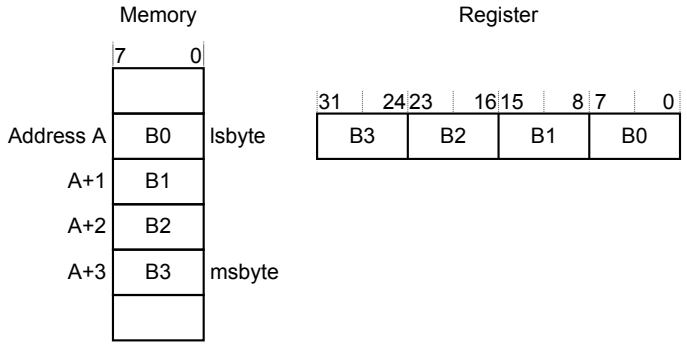
### 2.4.5.2 Directly Accessing a Bit-Band Region

"Behavior of Memory Accesses" on page 117 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

## 2.4.6 Data Storage

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Data is stored in little-endian format, with the least-significant byte (lsbyte) of a word stored at the lowest-numbered byte, and the most-significant byte (msbyte) stored at the highest-numbered byte. Figure 2-5 on page 122 illustrates how data is stored.

**Figure 2-5. Data Storage**




## 2.4.7 Synchronization Primitives

The Cortex-M4F instruction set includes pairs of synchronization primitives which provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use these primitives to perform a guaranteed read-modify-write memory update sequence or for a semaphore mechanism.

**Note:** The available pairs of synchronization primitives are only available for single processor use and should not be used with multi-processor systems.

A pair of synchronization primitives consists of:

- A Load-Exclusive instruction, which is used to read the value of a memory location and requests exclusive access to that location.
- A Store-Exclusive instruction, which is used to attempt to write to the same memory location and returns a status bit to a register. If this status bit is clear, it indicates that the thread or process gained exclusive access to the memory and the write succeeds; if this status bit is set, it indicates that the thread or process did not gain exclusive access to the memory and no write was performed.

The pairs of Load-Exclusive and Store-Exclusive instructions are:

- The word instructions LDREX and STREX
- The halfword instructions LDREXH and STREXH
- The byte instructions LDREXB and STREXB

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction.

To perform an exclusive read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Modify the value, as required.
3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location.
4. Test the returned status bit.

If the status bit is clear, the read-modify-write completed successfully. If the status bit is set, no write was performed, which indicates that the value returned at step 1 might be out of date. The software must retry the entire read-modify-write sequence.

Software can use the synchronization primitives to implement a semaphore as follows:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.
3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded, then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.

The Cortex-M4F includes an exclusive access monitor that tags the fact that the processor has executed a Load-Exclusive instruction. The processor removes its exclusive access tag if:

- It executes a CLREX instruction.
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs, which means the processor can resolve semaphore conflicts between different threads.

For more information about the synchronization primitive instructions, see the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)*.

## 2.5 Exception Model

The ARM Cortex-M4F processor and the Nested Vectored Interrupt Controller (NVIC) 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 vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration.

Table 2-8 on page 125 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 109 interrupts (listed in Table 2-9 on page 126).

Priorities on the system handlers are set with the NVIC **System Handler Priority n (SYSPRIn)** registers. Interrupts are enabled through the NVIC **Interrupt Set Enable n (ENn)** register and prioritized with the NVIC **Interrupt Priority n (PRIn)** registers. Priorities can be grouped by splitting priority levels into preemption priorities and subpriorities. All the interrupt registers are described in “Nested Vectored Interrupt Controller (NVIC)” on page 147.

Internally, the highest user-programmable priority (0) is treated as fourth priority, after a Reset, Non-Maskable Interrupt (NMI), and a Hard Fault, in that order. Note that 0 is the default priority for all the programmable priorities.

**Important:** After a write to clear an interrupt source, it may take several processor cycles for the NVIC to see the interrupt source deassert. 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 sees 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 or write after the write to clear the interrupt source (and flush the write buffer).

See “Nested Vectored Interrupt Controller (NVIC)” on page 147 for more information on exceptions and interrupts.

### 2.5.1 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 that is being serviced by the processor but has not completed.

  **Note:** 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.

### 2.5.2 Exception Types

The exception types are:

- **Reset.** Reset is invoked on power up or a warm 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 reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.
- **NMI.** A non-maskable Interrupt (NMI) can be signaled using the NMI signal or triggered by software using the **Interrupt Control and State (INTCTRL)** register. This exception has the highest priority other than reset. NMI is permanently enabled and has a fixed priority of -2. NMIs cannot be masked or prevented from activation by any other exception or preempted by any exception other than reset.
- **Hard Fault.** A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
- **Memory Management Fault.** A memory management fault is an exception that occurs because of a memory protection related fault, including access violation and no match. The MPU or the fixed memory protection constraints determine this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to Execute Never (XN) memory regions, even if the MPU is disabled.
- **Bus Fault.** A bus fault is an exception that occurs because of a memory-related fault for an instruction or data memory transaction such as a prefetch fault or a memory access fault. This fault can be enabled or disabled.

- **Usage Fault.** A usage fault is an exception that occurs because of a fault related to instruction execution, such as:
  - An undefined instruction
  - An illegal unaligned access
  - Invalid state on instruction execution
  - An error on exception return

  An unaligned address on a word or halfword memory access or division by zero can cause a usage fault when the core is properly configured.
- **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.
- **Debug Monitor.** This exception is caused by the debug monitor (when not halting). This exception is only active when enabled. This exception does not activate if it is a lower priority than the current activation.
- **PendSV.** PendSV is a pendable, interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. PendSV is triggered using the **Interrupt Control and State (INTCTRL)** register.
- **SysTick.** A SysTick exception is an exception that the system timer generates when it reaches zero when it is enabled to generate an interrupt. Software can also generate a SysTick exception using the **Interrupt Control and State (INTCTRL)** register. In an OS environment, the processor can use this exception as system tick.
- **Interrupt (IRQ).** An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized). All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor. Table 2-9 on page 126 lists the interrupts on the TM4C129XKCZAD controller.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that Table 2-8 on page 125 shows as having configurable priority (see the **SYSHNDCTRL** register on page 191 and the **DIS0** register on page 166).

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” on page 134.

**Table 2-8. Exception Types**

| Exception Type | Vector Number | Priority[a] | Vector Address or Offset[b] | Activation |
|---|---|---|---|---|
| - | 0 | - | 0x0000.0000 | Stack top is loaded from the first entry of the vector table on reset. |
| Reset | 1 | -3 (highest) | 0x0000.0004 | Asynchronous |
| Non-Maskable Interrupt (NMI) | 2 | -2 | 0x0000.0008 | Asynchronous |
| Hard Fault | 3 | -1 | 0x0000.000C | - |
| Memory Management | 4 | programmable[c] | 0x0000.0010 | Synchronous |

**Table 2-8. Exception Types *(continued)***

| Exception Type | Vector Number | Priority[a] | Vector Address or Offset[b] | Activation |
|---|---|---|---|---|
| Bus Fault | 5 | programmable[c] | 0x0000.0014 | Synchronous when precise and asynchronous when imprecise |
| Usage Fault | 6 | programmable[c] | 0x0000.0018 | Synchronous |
| - | 7-10 | - | - | Reserved |
| SVCall | 11 | programmable[c] | 0x0000.002C | Synchronous |
| Debug Monitor | 12 | programmable[c] | 0x0000.0030 | Synchronous |
| - | 13 | - | - | Reserved |
| PendSV | 14 | programmable[c] | 0x0000.0038 | Asynchronous |
| SysTick | 15 | programmable[c] | 0x0000.003C | Asynchronous |
| Interrupts | 16 and above | programmable[d] | 0x0000.0040 and above | Asynchronous |

a. 0 is the default priority for all the programmable priorities.

b. See “Vector Table” on page 129.

c. See **SYSPRI1** on page 188.

d. See **PRIn** registers on page 170.

**Table 2-9. Interrupts**

| Vector Number | Interrupt Number (Bit in Interrupt Registers) | Vector Address or Offset | Description |
|---|---|---|---|
| 0-15 | - | 0x0000.0000 - 0x0000.003C | Processor exceptions |
| 16 | 0 | 0x0000.0040 | GPIO Port A |
| 17 | 1 | 0x0000.0044 | GPIO Port B |
| 18 | 2 | 0x0000.0048 | GPIO Port C |
| 19 | 3 | 0x0000.004C | GPIO Port D |
| 20 | 4 | 0x0000.0050 | GPIO Port E |
| 21 | 5 | 0x0000.0054 | UART0 |
| 22 | 6 | 0x0000.0058 | UART1 |
| 23 | 7 | 0x0000.005C | SSI0 |
| 24 | 8 | 0x0000.0060 | $I^2C0$ |
| 25 | 9 | 0x0000.0064 | PWM Fault |
| 26 | 10 | 0x0000.0068 | PWM Generator 0 |
| 27 | 11 | 0x0000.006C | PWM Generator 1 |
| 28 | 12 | 0x0000.0070 | PWM Generator 2 |
| 29 | 13 | 0x0000.0074 | QEI0 |
| 30 | 14 | 0x0000.0078 | ADC0 Sequence 0 |
| 31 | 15 | 0x0000.007C | ADC0 Sequence 1 |
| 32 | 16 | 0x0000.0080 | ADC0 Sequence 2 |
| 33 | 17 | 0x0000.0084 | ADC0 Sequence 3 |
| 34 | 18 | 0x0000.0088 | Watchdog Timers 0 and 1 |
| 35 | 19 | 0x0000.008C | 16/32-Bit Timer 0A |
| 36 | 20 | 0x0000.0090 | 16/32-Bit Timer 0B |
| 37 | 21 | 0x0000.0094 | 16/32-Bit Timer 1A |

**Table 2-9. Interrupts *(continued)***

| Vector Number | Interrupt Number (Bit in Interrupt Registers) | Vector Address or Offset | Description |
|---|---|---|---|
| 38 | 22 | 0x0000.0098 | 16/32-Bit Timer 1B |
| 39 | 23 | 0x0000.009C | 16/32-Bit Timer 2A |
| 40 | 24 | 0x0000.00A0 | 16/32-Bit Timer 2B |
| 41 | 25 | 0x0000.00A4 | Analog Comparator 0 |
| 42 | 26 | 0x0000.00A8 | Analog Comparator 1 |
| 43 | 27 | 0x0000.00AC | Analog Comparator 2 |
| 44 | 28 | 0x0000.00B0 | System Control |
| 45 | 29 | 0x0000.00B4 | Flash Memory Control |
| 46 | 30 | 0x0000.00B8 | GPIO Port F |
| 47 | 31 | 0x0000.00BC | GPIO Port G |
| 48 | 32 | 0x0000.00C0 | GPIO Port H |
| 49 | 33 | 0x0000.00C4 | UART2 |
| 50 | 34 | 0x0000.00C8 | SSI1 |
| 51 | 35 | 0x0000.00CC | 16/32-Bit Timer 3A |
| 52 | 36 | 0x0000.00D0 | 16/32-Bit Timer 3B |
| 53 | 37 | 0x0000.00D4 | $I^2C1$ |
| 54 | 38 | 0x0000.00D8 | CAN 0 |
| 55 | 39 | 0x0000.00DC | CAN1 |
| 56 | 40 | 0x0000.00E0 | Ethernet MAC |
| 57 | 41 | 0x0000.00E4 | HIB |
| 58 | 42 | 0x0000.00E8 | USB MAC |
| 59 | 43 | 0x0000.00EC | PWM Generator 3 |
| 60 | 44 | 0x0000.00F0 | uDMA 0 Software |
| 61 | 45 | 0x0000.00F4 | uDMA 0 Error |
| 62 | 46 | 0x0000.00F8 | ADC1 Sequence 0 |
| 63 | 47 | 0x0000.00FC | ADC1 Sequence 1 |
| 64 | 48 | 0x0000.0100 | ADC1 Sequence 2 |
| 65 | 49 | 0x0000.0104 | ADC1 Sequence 3 |
| 66 | 50 | 0x0000.0108 | EPI 0 |
| 67 | 51 | 0x0000.010C | GPIO Port J |
| 68 | 52 | 0x0000.0110 | GPIO Port K |
| 69 | 53 | 0x0000.0114 | GPIO Port L |
| 70 | 54 | 0x0000.0118 | SSI 2 |
| 71 | 55 | 0x0000.011C | SSI 3 |
| 72 | 56 | 0x0000.0120 | UART 3 |
| 73 | 57 | 0x0000.0124 | UART 4 |
| 74 | 58 | 0x0000.0128 | UART 5 |
| 75 | 59 | 0x0000.012C | UART 6 |
| 76 | 60 | 0x0000.0130 | UART 7 |
| 77 | 61 | 0x0000.0134 | $I^2C$ 2 |
| 78 | 62 | 0x0000.0138 | $I^2C$ 3 |

**Table 2-9. Interrupts *(continued)***

| Vector Number | Interrupt Number (Bit in Interrupt Registers) | Vector Address or Offset | Description |
|---|---|---|---|
| 79 | 63 | 0x0000.013C | Timer 4A |
| 80 | 64 | 0x0000.0140 | Timer 4B |
| 81 | 65 | 0x0000.0144 | Timer 5A |
| 82 | 66 | 0x0000.0148 | Timer 5B |
| 83 | 67 | 0x0000.014C | Floating-Point Exception (imprecise) |
| 84-85 | 68-69 | - | Reserved |
| 86 | 70 | 0x0000.0158 | $I^2C$ 4 |
| 87 | 71 | 0x0000.015C | $I^2C$ 5 |
| 88 | 72 | 0x0000.0160 | GPIO Port M |
| 89 | 73 | 0x0000.0164 | GPIO Port N |
| 90 | 74 | - | Reserved |
| 91 | 75 | 0x0000.016C | Tamper |
| 92 | 76 | 0x0000.017 | GPIO Port P (Summary or P0) |
| 93 | 77 | 0x0000.0174 | GPIO Port P1 |
| 94 | 78 | 0x0000.0178 | GPIO Port P2 |
| 95 | 79 | 0x0000.017C | GPIO Port P3 |
| 96 | 80 | 0x0000.0180 | GPIO Port P4 |
| 97 | 81 | 0x0000.0184 | GPIO Port P5 |
| 98 | 82 | 0x0000.0188 | GPIO Port P6 |
| 99 | 83 | 0x0000.018C | GPIO Port P7 |
| 100 | 84 | 0x0000.0190 | GPIO Port Q (Summary or Q0) |
| 101 | 85 | 0x0000.0194 | GPIO Port Q1 |
| 102 | 86 | 0x0000.0198 | GPIO Port Q2 |
| 103 | 87 | 0x0000.019C | GPIO Port Q3 |
| 104 | 88 | 0x0000.01A0 | GPIO Port Q4 |
| 105 | 89 | 0x0000.01A4 | GPIO Port Q5 |
| 106 | 90 | 0x0000.01A8 | GPIO Port Q6 |
| 107 | 91 | 0x0000.01AC | GPIO Port Q7 |
| 108 | 92 | 0x0000.01B0 | GPIO Port R |
| 109 | 93 | 0x0000.01B4 | GPIO Port S |
| 110 | 94 | 0x0000.01B8 | SHA/MD5 |
| 111 | 95 | 0x0000.01BC | AES |
| 112 | 96 | 0x0000.01C0 | DES |
| 113 | 97 | 0x0000.01C4 | LCD |
| 114 | 98 | 0x0000.01C8 | 16/32-Bit Timer 6A |
| 115 | 99 | 0x0000.01CC | 16/32-Bit Timer 6B |
| 116 | 100 | 0x0000.01D0 | 16/32-Bit Timer 7A |
| 117 | 101 | 0x0000.01D4 | 16/32-Bit Timer 7B |
| 118 | 102 | 0x0000.01D8 | $I^2C$ 6 |
| 119 | 103 | 0x0000.01DC | $I^2C$ 7 |
| 120 | 104 | - | Reserved |

**Table 2-9. Interrupts *(continued)***

| Vector Number | Interrupt Number (Bit in Interrupt Registers) | Vector Address or Offset | Description |
|---|---|---|---|
| 121 | 105 | 0x0000.01E4 | 1-Wire |
| 122-124 | 106-108 | - | Reserved |
| 125 | 109 | 0x0000.01F4 | $I^2C$ 8 |
| 126 | 110 | 0x0000.01F8 | $I^2C$ 9 |
| 127 | 111 | 0x0000.01FC | GPIO T |
| 129 | 113 | - | Reserved |

### 2.5.3 Exception Handlers

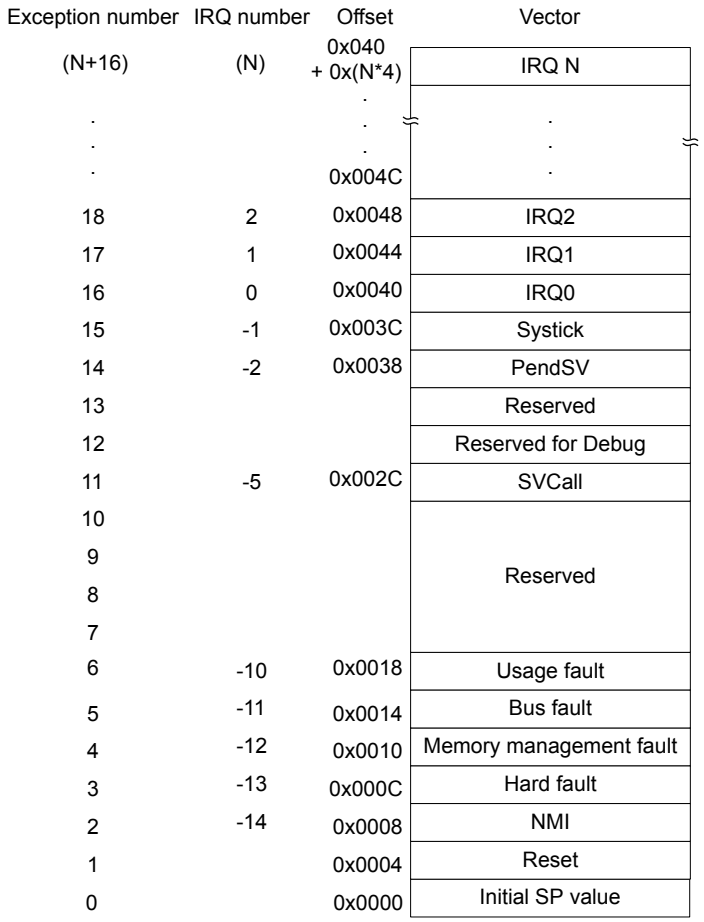
The processor handles exceptions using:

- **Interrupt Service Routines (ISRs).** Interrupts (IRQx) are the exceptions handled by ISRs.
- **Fault Handlers.** Hard fault, memory management fault, usage fault, and bus fault are fault exceptions handled by the fault handlers.
- **System Handlers.** NMI, PendSV, SVCall, SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

### 2.5.4 Vector Table

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. The vector table is constructed using the vector address or offset shown in Table 2-8 on page 125. Figure 2-6 on page 130 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code

**Figure 2-6. Vector Table**




On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the **Vector Table Offset (VTABLE)** register to relocate the vector table start address to a different memory location, in the range 0x0000.0400 to 0x3FFF.FC00 (see "Vector Table" on page 129). Note that when configuring the **VTABLE** register, the offset must be aligned on a 1024-byte boundary.

## 2.5.5 Exception Priorities

As Table 2-8 on page 125 shows, all exceptions have an associated priority, with a lower priority value indicating a higher priority and configurable priorities for all exceptions except Reset, Hard fault, and NMI. If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities, see page 188 and page 170.

**Note:** Configurable priority values for the Tiva™ C Series implementation are in the range 0-7. This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, 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.

### 2.5.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This grouping divides each interrupt priority register entry into two fields:

- An upper field that defines the group priority
- A lower field that defines a subpriority within the group

Only the group priority determines preemption 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 preempt 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.

For information about splitting the interrupt priority fields into group priority and subpriority, see page 182.

### 2.5.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

- **Preemption.** When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See “Interrupt Priority Grouping” on page 131 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” on page 132 more information.
- **Return.** Return occurs when the exception handler is completed, and there is no pending exception with sufficient priority to be serviced and the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See “Exception Return” on page 133 for more information.
- **Tail-Chaining.** This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.
- **Late-Arriving.** 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 is not affected by late arrival because the state saved is the same for both exceptions. Therefore, the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On

return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

#### 2.5.7.1 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.

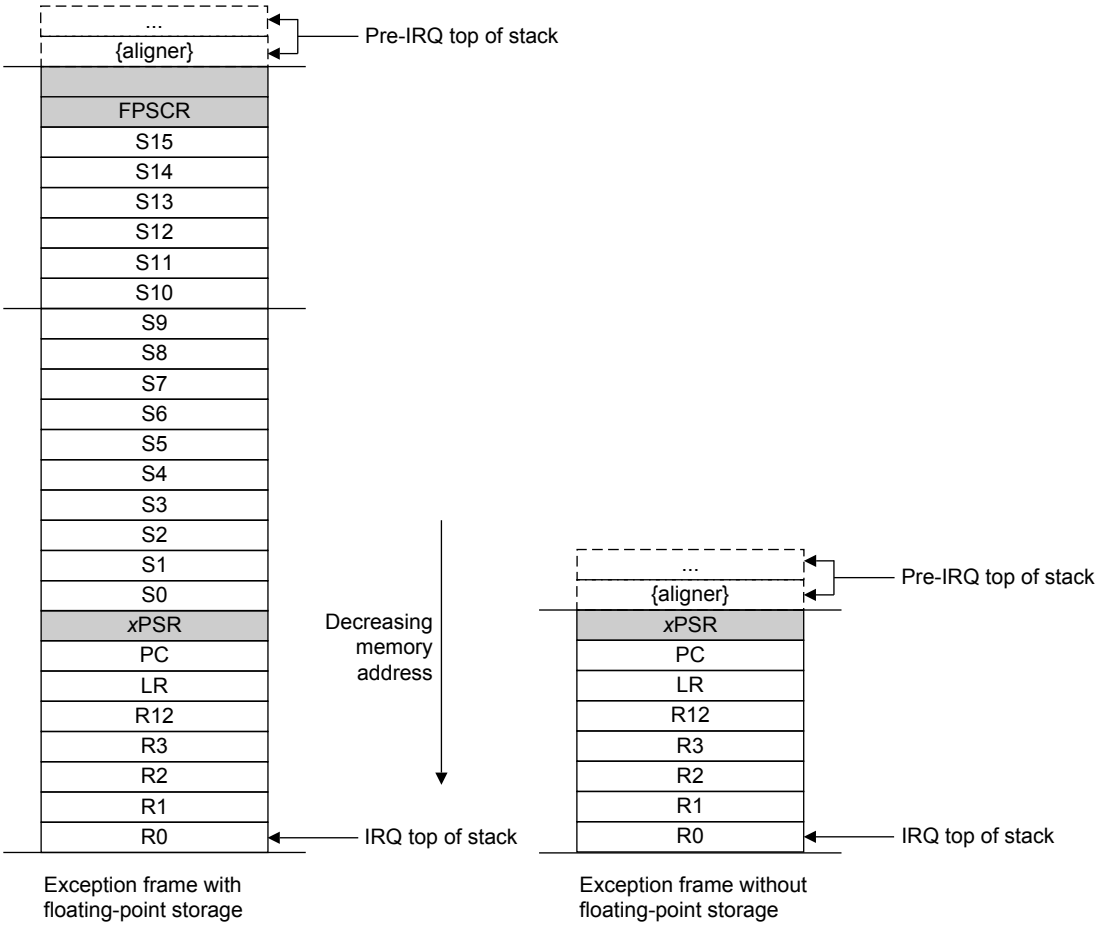
Sufficient priority means the exception has more priority than any limits set by the mask registers (see **PRIMASK** on page 106, **FAULTMASK** on page 107, and **BASEPRI** on page 108). An exception with less priority than this is pending but is not handled by the processor.

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 eight data words is referred to as *stack frame*.

When using floating-point routines, the Cortex-M4F processor automatically stacks the architected floating-point state on exception entry. Figure 2-7 on page 133 shows the Cortex-M4F stack frame layout when floating-point state is preserved on the stack as the result of an interrupt or an exception.

**Note:** Where stack space for floating-point state is not allocated, the stack frame is the same as that of ARMv7-M implementations without an FPU. Figure 2-7 on page 133 shows this stack frame also.

**Figure 2-7. Exception Stack Frame**




Immediately after stacking, the stack pointer indicates the lowest address in the stack frame.

The stack frame includes the return address, which is the address of the next instruction in the interrupted program. This value is restored to the **PC** at exception return so that the interrupted program resumes.

In parallel with the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the **LR**, indicating which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher-priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher-priority exception occurs during exception entry, known as late arrival, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception.

### 2.5.7.2 Exception Return

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC_RETURN value into the **PC**:

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

EXC_RETURN is the value loaded into the **LR** on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest five bits of this value provide information on the return stack and processor mode. Table 2-10 on page 134 shows the EXC_RETURN values with a description of the exception return behavior.

EXC_RETURN bits 31:5 are all set. When this value is loaded into the **PC**, it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

**Table 2-10. Exception Return Behavior**

| EXC_RETURN[31:0] | Description |
|---|---|
| 0xFFFF.FFE0 | Reserved |
| 0xFFFF.FFE1 | Return to Handler mode.<br>Exception return uses floating-point state from **MSP**.<br>Execution uses **MSP** after return. |
| 0xFFFF.FFE2 - 0xFFFF.FFE8 | Reserved |
| 0xFFFF.FFE9 | Return to Thread mode.<br>Exception return uses floating-point state from **MSP**.<br>Execution uses **MSP** after return. |
| 0xFFFF.FFEA - 0xFFFF.FFEC | Reserved |
| 0xFFFF.FFED | Return to Thread mode.<br>Exception return uses floating-point state from **PSP**.<br>Execution uses **PSP** after return. |
| 0xFFFF.FFEE - 0xFFFF.FFF0 | Reserved |
| 0xFFFF.FFF1 | Return to Handler mode.<br>Exception return uses non-floating-point state from **MSP**.<br>Execution uses **MSP** after return. |
| 0xFFFF.FFF2 - 0xFFFF.FFF8 | Reserved |
| 0xFFFF.FFF9 | Return to Thread mode.<br>Exception return uses non-floating-point state from **MSP**.<br>Execution uses **MSP** after return. |
| 0xFFFF.FFFA - 0xFFFF.FFFC | Reserved |
| 0xFFFF.FFFD | Return to Thread mode.<br>Exception return uses non-floating-point state from **PSP**.<br>Execution uses **PSP** after return. |
| 0xFFFF.FFFE - 0xFFFF.FFFF | Reserved |

## 2.6 Fault Handling

Faults are a subset of the exceptions (see “Exception Model” on page 123). The following conditions generate a fault:

- A bus error on an instruction fetch or vector table load or a data access.

- An internally detected error such as an undefined instruction or an attempt to change state with a BX instruction.
- Attempting to execute an instruction from a memory region marked as Non-Executable (XN).
- An MPU fault because of a privilege violation or an attempt to access an unmanaged region.

### 2.6.1 Fault Types

Table 2-11 on page 135 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates the fault has occurred. See page 195 for more information about the fault status registers.

**Table 2-11. Faults**

| Fault | Handler | Fault Status Register | Bit Name |
|---|---|---|---|
| Bus error on a vector read | Hard fault | **Hard Fault Status (HFAULTSTAT)** | VECT |
| Fault escalated to a hard fault | Hard fault | **Hard Fault Status (HFAULTSTAT)** | FORCED |
| MPU or default memory mismatch on instruction access | Memory management fault | **Memory Management Fault Status (MFAULTSTAT)** | IERR [a] |
| MPU or default memory mismatch on data access | Memory management fault | **Memory Management Fault Status (MFAULTSTAT)** | DERR |
| MPU or default memory mismatch on exception stacking | Memory management fault | **Memory Management Fault Status (MFAULTSTAT)** | MSTKE |
| MPU or default memory mismatch on exception unstacking | Memory management fault | **Memory Management Fault Status (MFAULTSTAT)** | MUSTKE |
| MPU or default memory mismatch during lazy floating-point state preservation | Memory management fault | **Memory Management Fault Status (MFAULTSTAT)** | MLSPERR |
| Bus error during exception stacking | Bus fault | **Bus Fault Status (BFAULTSTAT)** | BSTKE |
| Bus error during exception unstacking | Bus fault | **Bus Fault Status (BFAULTSTAT)** | BUSTKE |
| Bus error during instruction prefetch | Bus fault | **Bus Fault Status (BFAULTSTAT)** | IBUS |
| Bus error during lazy floating-point state preservation | Bus fault | **Bus Fault Status (BFAULTSTAT)** | BLSPE |
| Precise data bus error | Bus fault | **Bus Fault Status (BFAULTSTAT)** | PRECISE |
| Imprecise data bus error | Bus fault | **Bus Fault Status (BFAULTSTAT)** | IMPRE |
| Attempt to access a coprocessor | Usage fault | **Usage Fault Status (UFAULTSTAT)** | NOCP |
| Undefined instruction | Usage fault | **Usage Fault Status (UFAULTSTAT)** | UNDEF |
| Attempt to enter an invalid instruction set state [b] | Usage fault | **Usage Fault Status (UFAULTSTAT)** | INVSTAT |
| Invalid EXC_RETURN value | Usage fault | **Usage Fault Status (UFAULTSTAT)** | INVPC |
| Illegal unaligned load or store | Usage fault | **Usage Fault Status (UFAULTSTAT)** | UNALIGN |
| Divide by 0 | Usage fault | **Usage Fault Status (UFAULTSTAT)** | DIV0 |

a. Occurs on an access to an XN region even if the MPU is disabled.

b. Attempting to use an instruction set other than the Thumb instruction set, or returning to a non load-store-multiply instruction with ICI continuation.

### 2.6.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see **SYSPRI1** on page 188). Software can disable execution of the handlers for these faults (see **SYSHNDCTRL** on page 191).

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 as described in “Exception Model” on page 123.

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

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to hard fault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This situation happens 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 bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. Thus 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.

**Note:** Only Reset and NMI can preempt the fixed priority hard fault. A hard fault can preempt any exception other than Reset, NMI, or another hard fault.

### 2.6.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 2-12 on page 136.

**Table 2-12. Fault Status and Fault Address Registers**

| Handler | Status Register Name | Address Register Name | Register Description |
|---|---|---|---|
| Hard fault | **Hard Fault Status (HFAULTSTAT)** | - | page 201 |
| Memory management fault | **Memory Management Fault Status (MFAULTSTAT)** | **Memory Management Fault Address (MMADDR)** | page 195<br>page 202 |
| Bus fault | **Bus Fault Status (BFAULTSTAT)** | **Bus Fault Address (FAULTADDR)** | page 195<br>page 203 |
| Usage fault | **Usage Fault Status (UFAULTSTAT)** | - | page 195 |

### 2.6.4 Lockup

The processor enters a lockup state if a hard fault occurs when executing the NMI or hard fault handlers. When the processor is in the lockup state, it does not execute any instructions. The processor remains in lockup state until it is reset, an NMI occurs, or it is halted by a debugger.

**Note:** If the lockup state occurs from the NMI handler, a subsequent NMI does not cause the processor to leave the lockup state.

## 2.7 Power Management

The Cortex-M4F processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock.
- Deep-sleep mode stops the system clock and switches off the PLL and Flash memory.

The SLEEPDEEP bit of the **System Control (SYSCTRL)** register selects which sleep mode is used (see page 184). For more information about the behavior of the sleep modes, see "System Control" on page 251.

This section describes the mechanisms for entering sleep mode and the conditions for waking up from sleep mode, both of which apply to Sleep mode and Deep-sleep mode.

### 2.7.1 Entering Sleep Modes

This section describes the mechanisms software can use to put the processor into one of the sleep modes.

The system can generate spurious wake-up events, for example a debug operation wakes up the processor. Therefore, software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

#### 2.7.1.1 Wait for Interrupt

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode unless the wake-up condition is true (see "Wake Up from WFI or Sleep-on-Exit" on page 138). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information.

#### 2.7.1.2 Wait for Event

The wait for event instruction, WFE, causes entry to sleep mode conditional on the value of a one-bit event register. When the processor executes a WFE instruction, it checks the event register. If the register is 0, the processor stops executing instructions and enters sleep mode. If the register is 1, the processor clears the register and continues executing instructions without entering sleep mode.

If the event register is 1, the processor must not enter sleep mode on execution of a WFE instruction. Typically, this situation occurs if an SEV instruction has been executed. Software cannot access this register directly.

See the Cortex™-M4 instruction set chapter in the *ARM® Cortex™-M4 Devices Generic User Guide (literature number ARM DUI 0553A)* for more information.

#### 2.7.1.3 Sleep-on-Exit

If the SLEEPEXIT bit of the **SYSCTRL** register is set, when the processor completes the execution of all exception handlers, it returns to Thread mode and immediately enters sleep mode. This mechanism can be used in applications that only require the processor to run when an exception occurs.

### 2.7.2 Wake Up from Sleep Mode

The conditions for the processor to wake up depend on the mechanism that caused it to enter sleep mode.

#### 2.7.2.1 Wake Up from WFI or Sleep-on-Exit

Normally, the processor wakes up only when the NVIC detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the PRIMASK bit and clearing the FAULTMASK bit. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears PRIMASK. For more information about **PRIMASK** and **FAULTMASK**, see page 106 and page 107.

#### 2.7.2.2 Wake Up from WFE

The processor wakes up if it detects an exception with sufficient priority to cause exception entry.

In addition, if the SEVONPEND bit in the **SYSCTRL** register is set, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about **SYSCTRL**, see page 184.

## 2.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 2-13 on page 138 lists the supported instructions.

**Note:** In Table 2-13 on page 138:

- Angle brackets, <>, enclose alternative forms of the operand
- Braces, {}, enclose optional operands
- The Operands column is not exhaustive
- Op2 is a flexible second operand that can be either a register or a constant
- Most instructions can use an optional condition code suffix

For more information on the instructions and operands, see the instruction descriptions in the *ARM® Cortex™-M4 Technical Reference Manual*.

**Table 2-13. Cortex-M4F Instruction Summary**

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| ADC, ADCS | {Rd,} Rn, Op2 | Add with carry | N,Z,C,V |
| ADD, ADDS | {Rd,} Rn, Op2 | Add | N,Z,C,V |
| ADD, ADDW | {Rd,} Rn , #imm12 | Add | - |
| ADR | Rd, label | Load PC-relative address | - |
| AND, ANDS | {Rd,} Rn, Op2 | Logical AND | N,Z,C |
| ASR, ASRS | Rd, Rm, <Rs\|#n> | Arithmetic shift right | N,Z,C |
| B | label | Branch | - |
| BFC | Rd, #lsb, #width | Bit field clear | - |
| BFI | Rd, Rn, #lsb, #width | Bit field insert | - |
| BIC, BICS | {Rd,} Rn, Op2 | Bit clear | N,Z,C |
| BKPT | #imm | Breakpoint | - |
| BL | label | Branch with link | - |
| BLX | Rm | Branch indirect with link | - |
| BX | Rm | Branch indirect | - |
| CBNZ | Rn, label | Compare and branch if non-zero | - |

**Table 2-13. Cortex-M4F Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| CBZ | Rn, label | Compare and branch if zero | - |
| CLREX | - | Clear exclusive | - |
| CLZ | Rd, Rm | Count leading zeros | - |
| CMN | Rn, Op2 | Compare negative | N,Z,C,V |
| CMP | Rn, Op2 | Compare | N,Z,C,V |
| CPSID | i | Change processor state, disable interrupts | - |
| CPSIE | i | Change processor state, enable interrupts | - |
| DMB | - | Data memory barrier | - |
| DSB | - | Data synchronization barrier | - |
| EOR, EORS | {Rd,} Rn, Op2 | Exclusive OR | N,Z,C |
| ISB | - | Instruction synchronization barrier | - |
| IT | - | If-Then condition block | - |
| LDM | Rn{!}, reglist | Load multiple registers, increment after | - |
| LDMDB, LDMEA | Rn{!}, reglist | Load multiple registers, decrement before | - |
| LDMFD, LDMIA | Rn{!}, reglist | Load multiple registers, increment after | - |
| LDR | Rt, [Rn, #offset] | Load register with word | - |
| LDRB, LDRBT | Rt, [Rn, #offset] | Load register with byte | - |
| LDRD | Rt, Rt2, [Rn, #offset] | Load register with two bytes | - |
| LDREX | Rt, [Rn, #offset] | Load register exclusive | - |
| LDREXB | Rt, [Rn] | Load register exclusive with byte | - |
| LDREXH | Rt, [Rn] | Load register exclusive with halfword | - |
| LDRH, LDRHT | Rt, [Rn, #offset] | Load register with halfword | - |
| LDRSB, LDRSBT | Rt, [Rn, #offset] | Load register with signed byte | - |
| LDRSH, LDRSHT | Rt, [Rn, #offset] | Load register with signed halfword | - |
| LDRT | Rt, [Rn, #offset] | Load register with word | - |
| LSL, LSLS | Rd, Rm, <Rs\|#n> | Logical shift left | N,Z,C |
| LSR, LSRS | Rd, Rm, <Rs\|#n> | Logical shift right | N,Z,C |
| MLA | Rd, Rn, Rm, Ra | Multiply with accumulate, 32-bit result | - |
| MLS | Rd, Rn, Rm, Ra | Multiply and subtract, 32-bit result | - |
| MOV, MOVS | Rd, Op2 | Move | N,Z,C |
| MOV, MOVW | Rd, #imm16 | Move 16-bit constant | N,Z,C |
| MOVT | Rd, #imm16 | Move top | - |
| MRS | Rd, spec_reg | Move from special register to general register | - |
| MSR | spec_reg, Rm | Move from general register to special register | N,Z,C,V |
| MUL, MULS | {Rd,} Rn, Rm | Multiply, 32-bit result | N,Z |
| MVN, MVNS | Rd, Op2 | Move NOT | N,Z,C |
| NOP | - | No operation | - |
| ORN, ORNS | {Rd,} Rn, Op2 | Logical OR NOT | N,Z,C |

**Table 2-13. Cortex-M4F Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| ORR, ORRS | {Rd,} Rn, Op2 | Logical OR | N,Z,C |
| PKHTB, PKHBT | {Rd,} Rn, Rm, Op2 | Pack halfword | - |
| POP | reglist | Pop registers from stack | - |
| PUSH | reglist | Push registers onto stack | - |
| QADD | {Rd,} Rn, Rm | Saturating add | Q |
| QADD16 | {Rd,} Rn, Rm | Saturating add 16 | - |
| QADD8 | {Rd,} Rn, Rm | Saturating add 8 | - |
| QASX | {Rd,} Rn, Rm | Saturating add and subtract with exchange | - |
| QDADD | {Rd,} Rn, Rm | Saturating double and add | Q |
| QDSUB | {Rd,} Rn, Rm | Saturating double and subtract | Q |
| QSAX | {Rd,} Rn, Rm | Saturating subtract and add with exchange | - |
| QSUB | {Rd,} Rn, Rm | Saturating subtract | Q |
| QSUB16 | {Rd,} Rn, Rm | Saturating subtract 16 | - |
| QSUB8 | {Rd,} Rn, Rm | Saturating subtract 8 | - |
| RBIT | Rd, Rn | Reverse bits | - |
| REV | Rd, Rn | Reverse byte order in a word | - |
| REV16 | Rd, Rn | Reverse byte order in each halfword | - |
| REVSH | Rd, Rn | Reverse byte order in bottom halfword and sign extend | - |
| ROR, RORS | Rd, Rm, <Rs\|#n> | Rotate right | N,Z,C |
| RRX, RRXS | Rd, Rm | Rotate right with extend | N,Z,C |
| RSB, RSBS | {Rd,} Rn, Op2 | Reverse subtract | N,Z,C,V |
| SADD16 | {Rd,} Rn, Rm | Signed add 16 | GE |
| SADD8 | {Rd,} Rn, Rm | Signed add 8 | GE |
| SASX | {Rd,} Rn, Rm | Signed add and subtract with exchange | GE |
| SBC, SBCS | {Rd,} Rn, Op2 | Subtract with carry | N,Z,C,V |
| SBFX | Rd, Rn, #lsb, #width | Signed bit field extract | - |
| SDIV | {Rd,} Rn, Rm | Signed divide | - |
| SEL | {Rd,} Rn, Rm | Select bytes | - |
| SEV | - | Send event | - |
| SHADD16 | {Rd,} Rn, Rm | Signed halving add 16 | - |
| SHADD8 | {Rd,} Rn, Rm | Signed halving add 8 | - |
| SHASX | {Rd,} Rn, Rm | Signed halving add and subtract with exchange | - |
| SHSAX | {Rd,} Rn, Rm | Signed halving add and subtract with exchange | - |
| SHSUB16 | {Rd,} Rn, Rm | Signed halving subtract 16 | - |
| SHSUB8 | {Rd,} Rn, Rm | Signed halving subtract 8 | - |

**Table 2-13. Cortex-M4F Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| SMLABB, SMLABT, SMLATB, SMLATT | Rd, Rn, Rm, Ra | Signed multiply accumulate long (halfwords) | Q |
| SMLAD, SMLADX | Rd, Rn, Rm, Ra | Signed multiply accumulate dual | Q |
| SMLAL | RdLo, RdHi, Rn, Rm | Signed multiply with accumulate (32x32+64), 64-bit result | - |
| SMLALBB, SMLALBT, SMLALTB, SMLALTT | RdLo, RdHi, Rn, Rm | Signed multiply accumulate long (halfwords) | - |
| SMLALD, SMLALDX | RdLo, RdHi, Rn, Rm | Signed multiply accumulate long dual | - |
| SMLAWB,SMLAWT | Rd, Rn, Rm, Ra | Signed multiply accumulate, word by halfword | Q |
| SMLSD SMLSDX | Rd, Rn, Rm, Ra | Signed multiply subtract dual | Q |
| SMLSLD SMLSLDX | RdLo, RdHi, Rn, Rm | Signed multiply subtract long dual | |
| SMMLA | Rd, Rn, Rm, Ra | Signed most significant word multiply accumulate | - |
| SMMLS, SMMLR | Rd, Rn, Rm, Ra | Signed most significant word multiply subtract | - |
| SMMUL, SMMULR | {Rd,} Rn, Rm | Signed most significant word multiply | - |
| SMUAD SMUADX | {Rd,} Rn, Rm | Signed dual multiply add | Q |
| SMULBB, SMULBT, SMULTB, SMULTT | {Rd,} Rn, Rm | Signed multiply halfwords | - |
| SMULL | RdLo, RdHi, Rn, Rm | Signed multiply (32x32), 64-bit result | - |
| SMULWB, SMULWT | {Rd,} Rn, Rm | Signed multiply by halfword | - |
| SMUSD, SMUSDX | {Rd,} Rn, Rm | Signed dual multiply subtract | - |
| SSAT | Rd, #n, Rm {,shift #s} | Signed saturate | Q |
| SSAT16 | Rd, #n, Rm | Signed saturate 16 | Q |
| SSAX | {Rd,} Rn, Rm | Saturating subtract and add with exchange | GE |
| SSUB16 | {Rd,} Rn, Rm | Signed subtract 16 | - |
| SSUB8 | {Rd,} Rn, Rm | Signed subtract 8 | - |
| STM | Rn{!}, reglist | Store multiple registers, increment after | - |

**Table 2-13. Cortex-M4F Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| STMDB, STMEA | Rn{!}, reglist | Store multiple registers, decrement before | - |
| STMFD, STMIA | Rn{!}, reglist | Store multiple registers, increment after | - |
| STR | Rt, [Rn {, #offset}] | Store register word | - |
| STRB, STRBT | Rt, [Rn {, #offset}] | Store register byte | - |
| STRD | Rt, Rt2, [Rn {, #offset}] | Store register two words | - |
| STREX | Rt, Rt, [Rn {, #offset}] | Store register exclusive | - |
| STREXB | Rd, Rt, [Rn] | Store register exclusive byte | - |
| STREXH | Rd, Rt, [Rn] | Store register exclusive halfword | - |
| STRH, STRHT | Rt, [Rn {, #offset}] | Store register halfword | - |
| STRSB, STRSBT | Rt, [Rn {, #offset}] | Store register signed byte | - |
| STRSH, STRSHT | Rt, [Rn {, #offset}] | Store register signed halfword | - |
| STRT | Rt, [Rn {, #offset}] | Store register word | - |
| SUB, SUBS | {Rd,} Rn, Op2 | Subtract | N,Z,C,V |
| SUB, SUBW | {Rd,} Rn, #imm12 | Subtract 12-bit constant | N,Z,C,V |
| SVC | #imm | Supervisor call | - |
| SXTAB | {Rd,} Rn, Rm, {,ROR #} | Extend 8 bits to 32 and add | - |
| SXTAB16 | {Rd,} Rn, Rm,{,ROR #} | Dual extend 8 bits to 16 and add | - |
| SXTAH | {Rd,} Rn, Rm,{,ROR #} | Extend 16 bits to 32 and add | - |
| SXTB16 | {Rd,} Rm {,ROR #n} | Signed extend byte 16 | - |
| SXTB | {Rd,} Rm {,ROR #n} | Sign extend a byte | - |
| SXTH | {Rd,} Rm {,ROR #n} | Sign extend a halfword | - |
| TBB | [Rn, Rm] | Table branch byte | - |
| TBH | [Rn, Rm, LSL #1] | Table branch halfword | - |
| TEQ | Rn, Op2 | Test equivalence | N,Z,C |
| TST | Rn, Op2 | Test | N,Z,C |
| UADD16 | {Rd,} Rn, Rm | Unsigned add 16 | GE |
| UADD8 | {Rd,} Rn, Rm | Unsigned add 8 | GE |
| UASX | {Rd,} Rn, Rm | Unsigned add and subtract with exchange | GE |
| UHADD16 | {Rd,} Rn, Rm | Unsigned halving add 16 | - |
| UHADD8 | {Rd,} Rn, Rm | Unsigned halving add 8 | - |
| UHASX | {Rd,} Rn, Rm | Unsigned halving add and subtract with exchange | - |
| UHSAX | {Rd,} Rn, Rm | Unsigned halving subtract and add with exchange | - |
| UHSUB16 | {Rd,} Rn, Rm | Unsigned halving subtract 16 | - |
| UHSUB8 | {Rd,} Rn, Rm | Unsigned halving subtract 8 | - |
| UBFX | Rd, Rn, #lsb, #width | Unsigned bit field extract | - |
| UDIV | {Rd,} Rn, Rm | Unsigned divide | - |
| UMAAL | RdLo, RdHi, Rn, Rm | Unsigned multiply accumulate accumulate long (32x32+64), 64-bit result | - |

**Table 2-13. Cortex-M4F Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| UMLAL | RdLo, RdHi, Rn, Rm | Unsigned multiply with accumulate (32x32+32+32), 64-bit result | - |
| UMULL | RdLo, RdHi, Rn, Rm | Unsigned multiply (32x 2), 64-bit result | - |
| UQADD16 | {Rd,} Rn, Rm | Unsigned Saturating Add 16 | - |
| UQADD8 | {Rd,} Rn, Rm | Unsigned Saturating Add 8 | - |
| UQASX | {Rd,} Rn, Rm | Unsigned Saturating Add and Subtract with Exchange | - |
| UQSAX | {Rd,} Rn, Rm | Unsigned Saturating Subtract and Add with Exchange | - |
| UQSUB16 | {Rd,} Rn, Rm | Unsigned Saturating Subtract 16 | - |
| UQSUB8 | {Rd,} Rn, Rm | Unsigned Saturating Subtract 8 | - |
| USAD8 | {Rd,} Rn, Rm | Unsigned Sum of Absolute Differences | - |
| USADA8 | {Rd,} Rn, Rm, Ra | Unsigned Sum of Absolute Differences and Accumulate | - |
| USAT | Rd, #n, Rm {,shift #s} | Unsigned Saturate | Q |
| USAT16 | Rd, #n, Rm | Unsigned Saturate 16 | Q |
| USAX | {Rd,} Rn, Rm | Unsigned Subtract and add with Exchange | GE |
| USUB16 | {Rd,} Rn, Rm | Unsigned Subtract 16 | GE |
| USUB8 | {Rd,} Rn, Rm | Unsigned Subtract 8 | GE |
| UXTAB | {Rd,} Rn, Rm, {,ROR #} | Rotate, extend 8 bits to 32 and Add | - |
| UXTAB16 | {Rd,} Rn, Rm, {,ROR #} | Rotate, dual extend 8 bits to 16 and Add | - |
| UXTAH | {Rd,} Rn, Rm, {,ROR #} | Rotate, unsigned extend and Add Halfword | - |
| UXTB | {Rd,} Rm, {,ROR #n} | Zero extend a Byte | - |
| UXTB16 | {Rd,} Rm, {,ROR #n} | Unsigned Extend Byte 16 | - |
| UXTH | {Rd,} Rm, {,ROR #n} | Zero extend a Halfword | - |
| VABS.F32 | Sd, Sm | Floating-point Absolute | - |
| VADD.F32 | {Sd,} Sn, Sm | Floating-point Add | - |
| VCMP.F32 | Sd, <Sm \| #0.0> | Compare two floating-point registers, or one floating-point register and zero | FPSCR |
| VCMPE.F32 | Sd, <Sm \| #0.0> | Compare two floating-point registers, or one floating-point register and zero with Invalid Operation check | FPSCR |
| VCVT.S32.F32 | Sd, Sm | Convert between floating-point and integer | - |
| VCVT.S16.F32 | Sd, Sd, #fbits | Convert between floating-point and fixed point | - |
| VCVTR.S32.F32 | Sd, Sm | Convert between floating-point and integer with rounding | - |
| VCVT<B\|H>.F32.F16 | Sd, Sm | Converts half-precision value to single-precision | - |
| VCVTT<B\|T>.F32.F16 | Sd, Sm | Converts single-precision register to half-precision | - |
| VDIV.F32 | {Sd,} Sn, Sm | Floating-point Divide | - |
| VFMA.F32 | {Sd,} Sn, Sm | Floating-point Fused Multiply Accumulate | - |

**Table 2-13. Cortex-M4F Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| VFNMA.F32 | {Sd,} Sn, Sm | Floating-point Fused Negate Multiply Accumulate | - |
| VFMS.F32 | {Sd,} Sn, Sm | Floating-point Fused Multiply Subtract | - |
| VFNMS.F32 | {Sd,} Sn, Sm | Floating-point Fused Negate Multiply Subtract | - |
| VLDM.F<32\|64> | Rn{!}, list | Load Multiple extension registers | - |
| VLDR.F<32\|64> | <Dd\|Sd>, [Rn] | Load an extension register from memory | - |
| VLMA.F32 | {Sd,} Sn, Sm | Floating-point Multiply Accumulate | - |
| VLMS.F32 | {Sd,} Sn, Sm | Floating-point Multiply Subtract | - |
| VMOV.F32 | Sd, #imm | Floating-point Move immediate | - |
| VMOV | Sd, Sm | Floating-point Move register | - |
| VMOV | Sn, Rt | Copy ARM core register to single precision | - |
| VMOV | Sm, Sm1, Rt, Rt2 | Copy 2 ARM core registers to 2 single precision | - |
| VMOV | Dd[x], Rt | Copy ARM core register to scalar | - |
| VMOV | Rt, Dn[x] | Copy scalar to ARM core register | - |
| VMRS | Rt, FPSCR | Move FPSCR to ARM core register or APSR | N,Z,C,V |
| VMSR | FPSCR, Rt | Move to FPSCR from ARM Core register | FPSCR |
| VMUL.F32 | {Sd,} Sn, Sm | Floating-point Multiply | - |
| VNEG.F32 | Sd, Sm | Floating-point Negate | - |
| VNMLA.F32 | {Sd,} Sn, Sm | Floating-point Multiply and Add | - |
| VNMLS.F32 | {Sd,} Sn, Sm | Floating-point Multiply and Subtract | - |
| VNMUL | {Sd,} Sn, Sm | Floating-point Multiply | - |
| VPOP | list | Pop extension registers | - |
| VPUSH | list | Push extension registers | - |
| VSQRT.F32 | Sd, Sm | Calculates floating-point Square Root | - |
| VSTM | Rn{!}, list | Floating-point register Store Multiple | - |
| VSTR.F3<32\|64> | Sd, [Rn] | Stores an extension register to memory | - |
| VSUB.F<32\|64> | {Sd,} Sn, Sm | Floating-point Subtract | - |
| WFE | - | Wait for event | - |
| WFI | - | Wait for interrupt | - |

# 3 Cortex-M4 Peripherals

This chapter provides information on the Tiva™ C Series implementation of the Cortex-M4 processor peripherals, including:

- SysTick (see page 146)

  Provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.

- Nested Vectored Interrupt Controller (NVIC) (see page 147)
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers

- System Control Block (SCB) (see page 148)

  Provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

- Memory Protection Unit (MPU) (see page 148)

  Supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

- Floating-Point Unit (FPU) (see page 153)

  Fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions.

Table 3-1 on page 145 shows the address map of the Private Peripheral Bus (PPB). Some peripheral register regions are split into two address regions, as indicated by two addresses listed.

**Table 3-1. Core Peripheral Register Regions**

| Address | Core Peripheral | Description (see page ...) |
|---|---|---|
| 0xE000.E010-0xE000.E01F | System Timer | 146 |
| 0xE000.E100-0xE000.E4EF<br>0xE000.EF00-0xE000.EF03 | Nested Vectored Interrupt Controller | 147 |
| 0xE000.E008-0xE000.E00F<br>0xE000.ED00-0xE000.ED3F | System Control Block | 148 |
| 0xE000.ED90-0xE000.EDB8 | Memory Protection Unit | 148 |
| 0xE000.EF30-0xE000.EF44 | Floating Point Unit | 153 |

## 3.1 Functional Description

This chapter provides information on the Tiva™ C Series implementation of the Cortex-M4 processor peripherals: SysTick, NVIC, SCB, MPU, FPU.

### 3.1.1 System Timer (SysTick)

Cortex-M4 includes an integrated system timer, SysTick, which 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 as:

- 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 the system clock.
- A variable rate alarm or signal timer—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/meeting durations. The COUNT bit in the **STCTRL** control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

The timer consists of three registers:

- **SysTick Control and Status (STCTRL)**: A control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status.
- **SysTick Reload Value (STRELOAD)**: The reload value for the counter, used to provide the counter's wrap value.
- **SysTick Current Value (STCURRENT)**: The current value of the counter.

When enabled, the timer counts down on each clock from the reload value to zero, reloads (wraps) to the value in the **STRELOAD** register on the next clock edge, then decrements on subsequent clocks. Clearing the **STRELOAD** register disables the counter on the next wrap. When the counter reaches zero, the COUNT status bit is set. The COUNT bit clears on reads.

Writing to the **STCURRENT** register clears the register and the COUNT status bit. The write does not trigger the SysTick exception logic. On a read, the current value is the value of the register at the time the register is accessed.

The SysTick counter runs on the system clock. If this clock signal is stopped for low power mode, the SysTick counter stops. Ensure software uses aligned word accesses to access the SysTick registers.

The SysTick counter reload and current value are undefined at reset; the correct initialization sequence for the SysTick counter is:

1. Program the value in the **STRELOAD** register.
2. Clear the **STCURRENT** register by writing to it with any value.
3. Configure the **STCTRL** register for the required operation.

**Note:** When the processor is halted for debugging, the counter does not decrement.

### 3.1.2 Nested Vectored Interrupt Controller (NVIC)

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. The NVIC supports:

- 109 interrupts.
- A programmable priority level of 0-7 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Low-latency exception and interrupt handling.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non-maskable interrupt (NMI).

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead, providing low latency exception handling.

#### 3.1.2.1 Level-Sensitive and Pulse Interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt (see “Hardware and Software Control of Interrupts” on page 147 for more information). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. As a result, the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

#### 3.1.2.2 Hardware and Software Control of Interrupts

The Cortex-M4 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- The NVIC detects that the interrupt signal is High and the interrupt is not active.
- The NVIC detects a rising edge on the interrupt signal.
- Software writes to the corresponding interrupt set-pending register bit, or to the **Software Trigger Interrupt (SWTRIG)** register to make a Software-Generated Interrupt pending. See the `INT` bit in the **PEND0** register on page 167 or **SWTRIG** on page 174.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt, changing the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.

    If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit
  - For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the state of the interrupt changes to inactive, if the state was pending or to active, if the state was active and pending.

### 3.1.3 System Control Block (SCB)

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

### 3.1.4 Memory Protection Unit (MPU)

This section describes the Memory protection unit (MPU). The MPU divides the memory map into a number of regions and defines the location, size, access permissions, and memory attributes of each region. The MPU supports independent attribute settings for each region, overlapping regions, and export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M4 MPU defines eight separate memory regions, 0-7, and a background region.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M4 MPU memory map is unified, meaning that instruction accesses and data accesses have the same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a memory management fault, causing a fault exception and possibly causing termination of the process in an OS environment. In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types (see “Memory Regions, Types and Attributes” on page 117 for more information).

Table 3-2 on page 149 shows the possible MPU region attributes. See the section called “MPU Configuration for a Tiva™ C Series Microcontroller” on page 153 for guidelines for programming a microcontroller implementation.

**Table 3-2. Memory Attributes Summary**

| Memory Type | Description |
|---|---|
| Strongly Ordered | All accesses to Strongly Ordered memory occur in program order. |
| Device | Memory-mapped peripherals |
| Normal | Normal memory |

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- Except for the **MPU Region Attribute and Size (MPUATTR)** register, all MPU registers must be accessed with aligned word accesses.
- The **MPUATTR** register can be accessed with byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

#### 3.1.4.1 Updating an MPU Region

To update the attributes for an MPU region, the **MPU Region Number (MPUNUMBER)**, **MPU Region Base Address (MPUBASE)** and **MPUATTR** registers must be updated. Each register can be programmed separately or with a multiple-word write to program all of these registers. You can use the **MPUBASEx** and **MPUATTRx** aliases to program up to four regions simultaneously using an STM instruction.

***Updating an MPU Region Using Separate Words***

This example simple code configures one region:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER        ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]       ; Region Number
STR R4, [R0, #0x4]       ; Region Base Address
STRH R2, [R0, #0x8]      ; Region Size and Enable
STRH R3, [R0, #0xA]      ; Region Attribute
```

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER         ; 0xE000ED98, MPU region number register
```

```
STR R1, [R0, #0x0]        ; Region Number
BIC R2, R2, #1            ; Disable
STRH R2, [R0, #0x8]       ; Region Size and Enable
STR R4, [R0, #0x4]        ; Region Base Address
STRH R3, [R0, #0xA]       ; Region Attribute
ORR R2, #1                ; Enable
STRH R2, [R0, #0x8]       ; Region Size and Enable
```

Software must use memory barrier instructions:

- Before MPU setup, if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings.
- After MPU setup, if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not need any memory barrier instructions during MPU setup, because it accesses the MPU through the Private Peripheral Bus (PPB), which is a Strongly Ordered memory region.

For example, if all of the memory access behavior is intended to take effect immediately after the programming sequence, then a DSB instruction and an ISB instruction should be used. A DSB is required after changing MPU settings, such as at the end of context switch. An ISB is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then an ISB is not required.

### *Updating an MPU Region Using Multi-Word Writes*

The MPU can be programmed directly using multi-word writes, depending how the information is divided. Consider the following reprogramming:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER  ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]  ; Region Number
STR R2, [R0, #0x4]  ; Region Base Address
STR R3, [R0, #0x8]  ; Region Attribute, Size and Enable
```

An STM instruction can be used to optimize this:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER  ; 0xE000ED98, MPU region number register
STM R0, {R1-R3}     ; Region number, address, attribute, size and enable
```

This operation can be done in two words for prepacked information, meaning that the **MPU Region Base Address (MPUBASE)** register (see page 208) contains the required region number and has the VALID bit set. This method can be used when the data is statically packed, for example in a boot loader:

```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, =MPUBASE    ; 0xE000ED9C, MPU Region Base register
STR R1, [R0, #0x0]  ; Region base address and region number combined
                    ; with VALID (bit 4) set
STR R2, [R0, #0x4]  ; Region Attribute, Size and Enable
```

***Subregions***

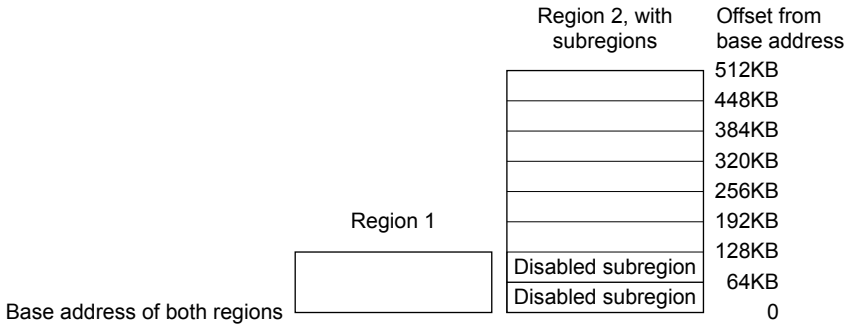
Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the **MPU Region Attribute and Size (MPUATTR)** register (see page 210) to disable a subregion. The least-significant bit of the SRD field controls the first subregion, and the most-significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions. With regions of these sizes, the SRD field must be configured to 0x00, otherwise the MPU behavior is unpredictable.

***Example of SRD Use***

Two regions with the same base address overlap. Region one is 128 KB, and region two is 512 KB. To ensure the attributes from region one apply to the first 128 KB region, configure the SRD field for region two to 0x03 to disable the first two subregions, as Figure 3-1 on page 151 shows.

**Figure 3-1. SRD Use Example**




### 3.1.4.2 MPU Access Permission Attributes

The access permission bits, TEX, S, C, B, AP, and XN of the **MPUATTR** register, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

Table 3-3 on page 151 shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M4 does not support the concept of cacheability or shareability. Refer to the section called “MPU Configuration for a Tiva™ C Series Microcontroller” on page 153 for information on programming the MPU for TM4C129XKCZAD implementations.

**Table 3-3. TEX, S, C, and B Bit Field Encoding**

| TEX | S | C | B | Memory Type | Shareability | Other Attributes |
|---|---|---|---|---|---|---|
| 000b | x[a] | 0 | 0 | Strongly Ordered | Shareable | - |
| 000 | x[a] | 0 | 1 | Device | Shareable | - |

**Table 3-3. TEX, S, C, and B Bit Field Encoding *(continued)***

| TEX | S | C | B | Memory Type | Shareability | Other Attributes |
|---|---|---|---|---|---|---|
| 000 | 0 | 1 | 0 | Normal | Not shareable | Outer and inner write-through. No write allocate. |
| 000 | 1 | 1 | 0 | Normal | Shareable | |
| 000 | 0 | 1 | 1 | Normal | Not shareable | |
| 000 | 1 | 1 | 1 | Normal | Shareable | |
| 001 | 0 | 0 | 0 | Normal | Not shareable | Outer and inner non-cacheable. |
| 001 | 1 | 0 | 0 | Normal | Shareable | |
| 001 | x[a] | 0 | 1 | Reserved encoding | - | - |
| 001 | x[a] | 1 | 0 | Reserved encoding | - | - |
| 001 | 0 | 1 | 1 | Normal | Not shareable | Outer and inner write-back. Write and read allocate. |
| 001 | 1 | 1 | 1 | Normal | Shareable | |
| 010 | x[a] | 0 | 0 | Device | Not shareable | Nonshared Device. |
| 010 | x[a] | 0 | 1 | Reserved encoding | - | - |
| 010 | x[a] | 1 | x[a] | Reserved encoding | - | - |
| 1BB | 0 | A | A | Normal | Not shareable | Cached memory (BB = outer policy, AA = inner policy).<br>See Table 3-4 for the encoding of the AA and BB bits. |
| 1BB | 1 | A | A | Normal | Shareable | |

a. The MPU ignores the value of this bit.

Table 3-4 on page 152 shows the cache policy for memory attribute encodings with a TEX value in the range of 0x4-0x7.

**Table 3-4. Cache Policy for Memory Attribute Encoding**

| Encoding, AA or BB | Corresponding Cache Policy |
|---|---|
| 00 | Non-cacheable |
| 01 | Write back, write and read allocate |
| 10 | Write through, no write allocate |
| 11 | Write back, no write allocate |

Table 3-5 on page 152 shows the AP encodings in the **MPUATTR** register that define the access permissions for privileged and unprivileged software.

**Table 3-5. AP Bit Field Encoding**

| AP Bit Field | Privileged Permissions | Unprivileged Permissions | Description |
|---|---|---|---|
| 000 | No access | No access | All accesses generate a permission fault. |
| 001 | RW | No access | Access from privileged software only. |
| 010 | RW | RO | Writes by unprivileged software generate a permission fault. |
| 011 | RW | RW | Full access. |
| 100 | Unpredictable | Unpredictable | Reserved. |
| 101 | RO | No access | Reads by privileged software only. |

**Table 3-5. AP Bit Field Encoding *(continued)***

| AP Bit Field | Privileged Permissions | Unprivileged Permissions | Description |
|---|---|---|---|
| 110 | RO | RO | Read-only, by privileged or unprivileged software. |
| 111 | RO | RO | Read-only, by privileged or unprivileged software. |

***MPU Configuration for a Tiva™ C Series Microcontroller***

Tiva™ C Series microcontrollers have only a single processor and no caches. As a result, the MPU should be programmed as shown in Table 3-6 on page 153.

**Table 3-6. Memory Region Attributes for Tiva™ C Series Microcontrollers**

| Memory Region | TEX | S | C | B | Memory Type and Attributes |
|---|---|---|---|---|---|
| Flash memory | 000b | 0 | 1 | 0 | Normal memory, non-shareable, write-through |
| Internal SRAM | 000b | 1 | 1 | 0 | Normal memory, shareable, write-through |
| External SRAM | 000b | 1 | 1 | 1 | Normal memory, shareable, write-back, write-allocate |
| Peripherals | 000b | 1 | 0 | 1 | Device memory, shareable |

In current Tiva™ C Series microcontroller implementations, the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable. The values given are for typical situations.

#### 3.1.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault (see “Exceptions and Interrupts” on page 113 for more information). The **MFAULTSTAT** register indicates the cause of the fault. See page 195 for more information.

### 3.1.5 Floating-Point Unit (FPU)

This section describes the Floating-Point Unit (FPU) and the registers it uses. The FPU provides:

- 32-bit instructions for single-precision (C float) data-processing operations
- Combined multiply and accumulate instructions for increased precision (Fused MAC)
- Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root
- Hardware support for denormals and all IEEE rounding modes
- 32 dedicated 32-bit single-precision registers, also addressable as 16 double-word registers
- Decoupled three stage pipeline

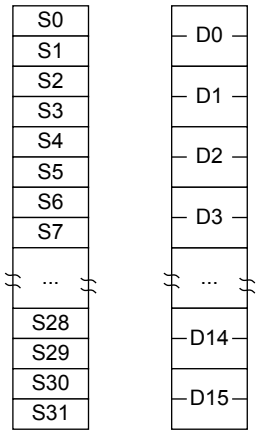
The Cortex-M4F FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed-point and floating-point data formats, and floating-point constant instructions. The FPU provides floating-point computation 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's single-precision extension registers can also be accessed as 16 doubleword registers for load, store, and move operations.

### 3.1.5.1 FPU Views of the Register Bank

The FPU provides an extension register file containing 32 single-precision registers. These can be viewed as:

- Sixteen 64-bit doubleword registers, D0-D15
- Thirty-two 32-bit single-word registers, S0-S31
- A combination of registers from the above views

**Figure 3-2. FPU Register Bank**




The mapping between the registers is as follows:

- S<2n> maps to the least significant half of D<n>
- S<2n+1> maps to the most significant half of D<n>

For example, you can access the least significant half of the value in D6 by accessing S12, and the most significant half of the elements by accessing S13.

### 3.1.5.2 Modes of Operation

The FPU provides three modes of operation to accommodate a variety of applications.

**Full-Compliance mode.** In Full-Compliance mode, the FPU processes all operations according to the IEEE 754 standard in hardware.

**Flush-to-Zero mode.** Setting the FZ bit of the **Floating-Point Status and Control (FPSC)** register enables Flush-to-Zero mode. In this mode, the FPU treats all subnormal input operands of arithmetic CDP operations as zeros in the operation. Exceptions that result from a zero operand are signalled appropriately. VABS, VNEG, and VMOV are not considered arithmetic CDP operations and are not affected by Flush-to-Zero mode. A result that is tiny, as described in the IEEE 754 standard, where the destination precision is smaller in magnitude than the minimum normal value before rounding, is replaced with a zero. The IDC bit in **FPSC** indicates when an input flush occurs. The UFC bit in **FPSC** indicates when a result flush occurs.

**Default NaN mode.** Setting the DN bit in the **FPSC** register enables default NaN mode. In this mode, the result of any arithmetic data processing operation that involves an input NaN, or that generates a NaN result, returns the default NaN. Propagation of the fraction bits is maintained only by VABS,

VNEG, and VMOV operations. All other CDP operations ignore any information in the fraction bits of an input NaN.

### 3.1.5.3 Compliance with the IEEE 754 standard

When Default NaN (DN) and Flush-to-Zero (FZ) modes are disabled, FPv4 functionality is compliant with the IEEE 754 standard in hardware. No support code is required to achieve this compliance.

### 3.1.5.4 Complete Implementation of the IEEE 754 standard

The Cortex-M4F floating point instruction set does not support all operations defined in the IEEE 754-2008 standard. Unsupported operations include, but are not limited to the following:

- Remainder
- Round floating-point number to integer-valued floating-point number
- Binary-to-decimal conversions
- Decimal-to-binary conversions
- Direct comparison of single-precision and double-precision values

The Cortex-M4 FPU supports fused MAC operations as described in the IEEE standard. For complete implementation of the IEEE 754-2008 standard, floating-point functionality must be augmented with library functions.

### 3.1.5.5 IEEE 754 standard implementation choices

#### *NaN handling*

All single-precision values with the maximum exponent field value and a nonzero fraction field are valid NaNs. A most-significant fraction bit of zero indicates a Signaling NaN (SNaN). A one indicates a Quiet NaN (QNaN). Two NaN values are treated as different NaNs if they differ in any bit. The below table shows the default NaN values.

| Sign | Fraction | Fraction |
|---|---|---|
| 0 | 0xFF | bit [22] = 1, bits [21:0] are all zeros |

Processing of input NaNs for ARM floating-point functionality and libraries is defined as follows:

- In full-compliance mode, NaNs are handled as described in the ARM Architecture Reference Manual. The hardware processes the NaNs directly for arithmetic CDP instructions. For data transfer operations, NaNs are transferred without raising the Invalid Operation exception. For the non-arithmetic CDP instructions, VABS, VNEG, and VMOV, NaNs are copied, with a change of sign if specified in the instructions, without causing the Invalid Operation exception.
- In default NaN mode, arithmetic CDP instructions involving NaN operands return the default NaN regardless of the fractions of any NaN operands. SNaNs in an arithmetic CDP operation set the IOC flag, FPSCR[0]. NaN handling by data transfer and non-arithmetic CDP instructions is the same as in full-compliance mode.

**Table 3-7. QNaN and SNaN Handling**

| Instruction Type | Default NaN Mode | With QNaN Operand | With SNaN Operand |
|---|---|---|---|
| Arithmetic CDP | Off | The QNaN or one of the QNaN operands, if there is more than one, is returned according to the rules given in the ARM Architecture Reference Manual. | IOC[a] set. The SNaN is quieted and the result NaN is determined by the rules given in the ARM Architecture Reference Manual. |
| | On | Default NaN returns. | IOC[a] set. Default NaN returns. |
| Non-arithmetic CDP | Off/On | NaN passes to destination with sign changed as appropriate. | |
| FCMP(Z) | - | Unordered compare. | IOC set. Unordered compare. |
| FCMPE(Z) | - | IOC set. Unordered compare. | IOC set. Unordered compare. |
| Load/store | Off/On | All NaNs transferred. | |

a. IOC is the Invalid Operation exception flag, FPSCR[0].

#### *Comparisons*

Comparison results modify the flags in the FPSCR. You can use the MVRS APSR_nzcv instruction (formerly FMSTAT) to transfer the current flags from the FPSCR to the APSR. See the ARM Architecture Reference Manual for mapping of IEEE 754-2008 standard predicates to ARM conditions. The flags used are chosen so that subsequent conditional execution of ARM instructions can test the predicates defined in the IEEE standard.

#### *Underflow*

The Cortex-M4F FPU uses the before rounding form of tininess and the inexact result form of loss of accuracy as described in the IEEE 754-2008 standard to generate Underflow exceptions.

In flush-to-zero mode, results that are tiny before rounding, as described in the IEEE standard, are flushed to a zero, and the UFC flag, FPSCR[3], is set. See the ARM Architecture Reference Manual for information on flush-to-zero mode.

When the FPU is not in flush-to-zero mode, operations are performed on subnormal operands. If the operation does not produce a tiny result, it returns the computed result, and the UFC flag, FPSCR[3], is not set. The IXC flag, FPSCR[4], is set if the operation is inexact. If the operation produces a tiny result, the result is a subnormal or zero value, and the UFC flag, FPSCR[3], is set if the result was also inexact.

### 3.1.5.6 Exceptions

The FPU sets the cumulative exception status flag in the FPSCR register as required for each instruction, in accordance with the FPv4 architecture. The FPU does not support user-mode traps. The exception enable bits in the FPSCR read-as-zero, and writes are ignored. The processor also has six output pins, FPIXC, FPUFC, FPOFC, FPDZC, FPIDC, and FPIOC, that each reflect the status of one of the cumulative exception flags. For a description of these outputs, see the *ARM Cortex-M4 Integration and Implementation Manual* (ARM DII 0239, available from ARM).

The processor can reduce the exception latency by using lazy stacking. See Auxiliary Control Register, ACTLR on page 4-5. This means that the processor reserves space on the stack for the FP state, but does not save that state information to the stack. See the ARMv7-M Architecture Reference Manual (available from ARM) for more information.

### 3.1.5.7 Enabling the FPU

The FPU is disabled from reset. You must enable it before you can use any floating-point instructions. The processor must be in privileged mode to read from and write to the **Coprocessor Access**

**Control (CPAC)** register. The below example code sequence enables the FPU in both privileged and user modes.

```
; CPACR is located at address 0xE000ED88
LDR.W R0, =0xE000ED88
; Read CPACR
LDR R1, [R0]
; Set bits 20-23 to enable CP10 and CP11 coprocessors
ORR R1, R1, #(0xF << 20)
; Write back the modified value to the CPACR
STR R1, [R0]; wait for store to complete
DSB
;reset pipeline now the FPU is enabled
ISB
```

## 3.2 Register Map

Table 3-8 on page 157 lists the Cortex-M4 Peripheral SysTick, NVIC, MPU, FPU and SCB registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000.E000.

**Note:** Register spaces that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

**Table 3-8. Peripherals Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **System Timer (SysTick) Registers** | | | | | |
| 0x010 | STCTRL | RW | 0x0000.0000 | SysTick Control and Status Register | 161 |
| 0x014 | STRELOAD | RW | - | SysTick Reload Value Register | 163 |
| 0x018 | STCURRENT | RWC | - | SysTick Current Value Register | 164 |
| **Nested Vectored Interrupt Controller (NVIC) Registers** | | | | | |
| 0x100 | EN0 | RW | 0x0000.0000 | Interrupt 0-31 Set Enable | 165 |
| 0x104 | EN1 | RW | 0x0000.0000 | Interrupt 32-63 Set Enable | 165 |
| 0x108 | EN2 | RW | 0x0000.0000 | Interrupt 64-95 Set Enable | 165 |
| 0x10C | EN3 | RW | 0x0000.0000 | Interrupt 96-113 Set Enable | 165 |
| 0x180 | DIS0 | RW | 0x0000.0000 | Interrupt 0-31 Clear Enable | 166 |
| 0x184 | DIS1 | RW | 0x0000.0000 | Interrupt 32-63 Clear Enable | 166 |
| 0x188 | DIS2 | RW | 0x0000.0000 | Interrupt 64-95 Clear Enable | 166 |
| 0x18C | DIS3 | RW | 0x0000.0000 | Interrupt 96-113 Clear Enable | 166 |
| 0x200 | PEND0 | RW | 0x0000.0000 | Interrupt 0-31 Set Pending | 167 |
| 0x204 | PEND1 | RW | 0x0000.0000 | Interrupt 32-63 Set Pending | 167 |
| 0x208 | PEND2 | RW | 0x0000.0000 | Interrupt 64-95 Set Pending | 167 |
| 0x20C | PEND3 | RW | 0x0000.0000 | Interrupt 96-113 Set Pending | 167 |

**Table 3-8. Peripherals Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x280 | UNPEND0 | RW | 0x0000.0000 | Interrupt 0-31 Clear Pending | 168 |
| 0x284 | UNPEND1 | RW | 0x0000.0000 | Interrupt 32-63 Clear Pending | 168 |
| 0x288 | UNPEND2 | RW | 0x0000.0000 | Interrupt 64-95 Clear Pending | 168 |
| 0x28C | UNPEND3 | RW | 0x0000.0000 | Interrupt 96-113 Clear Pending | 168 |
| 0x300 | ACTIVE0 | RO | 0x0000.0000 | Interrupt 0-31 Active Bit | 169 |
| 0x304 | ACTIVE1 | RO | 0x0000.0000 | Interrupt 32-63 Active Bit | 169 |
| 0x308 | ACTIVE2 | RO | 0x0000.0000 | Interrupt 64-95 Active Bit | 169 |
| 0x30C | ACTIVE3 | RO | 0x0000.0000 | Interrupt 96-127 Active Bit | 169 |
| 0x400 | PRI0 | RW | 0x0000.0000 | Interrupt 0-3 Priority | 170 |
| 0x404 | PRI1 | RW | 0x0000.0000 | Interrupt 4-7 Priority | 170 |
| 0x408 | PRI2 | RW | 0x0000.0000 | Interrupt 8-11 Priority | 170 |
| 0x40C | PRI3 | RW | 0x0000.0000 | Interrupt 12-15 Priority | 170 |
| 0x410 | PRI4 | RW | 0x0000.0000 | Interrupt 16-19 Priority | 170 |
| 0x414 | PRI5 | RW | 0x0000.0000 | Interrupt 20-23 Priority | 170 |
| 0x418 | PRI6 | RW | 0x0000.0000 | Interrupt 24-27 Priority | 170 |
| 0x41C | PRI7 | RW | 0x0000.0000 | Interrupt 28-31 Priority | 170 |
| 0x420 | PRI8 | RW | 0x0000.0000 | Interrupt 32-35 Priority | 170 |
| 0x424 | PRI9 | RW | 0x0000.0000 | Interrupt 36-39 Priority | 170 |
| 0x428 | PRI10 | RW | 0x0000.0000 | Interrupt 40-43 Priority | 170 |
| 0x42C | PRI11 | RW | 0x0000.0000 | Interrupt 44-47 Priority | 170 |
| 0x430 | PRI12 | RW | 0x0000.0000 | Interrupt 48-51 Priority | 170 |
| 0x434 | PRI13 | RW | 0x0000.0000 | Interrupt 52-55 Priority | 170 |
| 0x438 | PRI14 | RW | 0x0000.0000 | Interrupt 56-59 Priority | 170 |
| 0x43C | PRI15 | RW | 0x0000.0000 | Interrupt 60-63 Priority | 170 |
| 0x440 | PRI16 | RW | 0x0000.0000 | Interrupt 64-67 Priority | 172 |
| 0x444 | PRI17 | RW | 0x0000.0000 | Interrupt 68-71 Priority | 172 |
| 0x448 | PRI18 | RW | 0x0000.0000 | Interrupt 72-75 Priority | 172 |
| 0x44C | PRI19 | RW | 0x0000.0000 | Interrupt 76-79 Priority | 172 |
| 0x450 | PRI20 | RW | 0x0000.0000 | Interrupt 80-83 Priority | 172 |
| 0x454 | PRI21 | RW | 0x0000.0000 | Interrupt 84-87 Priority | 172 |
| 0x458 | PRI22 | RW | 0x0000.0000 | Interrupt 88-91 Priority | 172 |
| 0x45C | PRI23 | RW | 0x0000.0000 | Interrupt 92-95 Priority | 172 |

**Table 3-8. Peripherals Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x460 | PRI24 | RW | 0x0000.0000 | Interrupt 96-99 Priority | 172 |
| 0x464 | PRI25 | RW | 0x0000.0000 | Interrupt 100-103 Priority | 172 |
| 0x468 | PRI26 | RW | 0x0000.0000 | Interrupt 104-107 Priority | 172 |
| 0x46C | PRI27 | RW | 0x0000.0000 | Interrupt 108-111 Priority | 172 |
| 0x470 | PRI28 | RW | 0x0000.0000 | Interrupt 112-113 Priority | 172 |
| 0xF00 | SWTRIG | WO | 0x0000.0000 | Software Trigger Interrupt | 174 |
| **System Control Block (SCB) Registers** | | | | | |
| 0x008 | ACTLR | RW | 0x0000.0000 | Auxiliary Control | 175 |
| 0xD00 | CPUID | RO | 0x410F.C241 | CPU ID Base | 177 |
| 0xD04 | INTCTRL | RW | 0x0000.0000 | Interrupt Control and State | 178 |
| 0xD08 | VTABLE | RW | 0x0000.0000 | Vector Table Offset | 181 |
| 0xD0C | APINT | RW | 0xFA05.0000 | Application Interrupt and Reset Control | 182 |
| 0xD10 | SYSCTRL | RW | 0x0000.0000 | System Control | 184 |
| 0xD14 | CFGCTRL | RW | 0x0000.0200 | Configuration and Control | 186 |
| 0xD18 | SYSPRI1 | RW | 0x0000.0000 | System Handler Priority 1 | 188 |
| 0xD1C | SYSPRI2 | RW | 0x0000.0000 | System Handler Priority 2 | 189 |
| 0xD20 | SYSPRI3 | RW | 0x0000.0000 | System Handler Priority 3 | 190 |
| 0xD24 | SYSHNDCTRL | RW | 0x0000.0000 | System Handler Control and State | 191 |
| 0xD28 | FAULTSTAT | RW1C | 0x0000.0000 | Configurable Fault Status | 195 |
| 0xD2C | HFAULTSTAT | RW1C | 0x0000.0000 | Hard Fault Status | 201 |
| 0xD34 | MMADDR | RW | - | Memory Management Fault Address | 202 |
| 0xD38 | FAULTADDR | RW | - | Bus Fault Address | 203 |
| **Memory Protection Unit (MPU) Registers** | | | | | |
| 0xD90 | MPUTYPE | RO | 0x0000.0800 | MPU Type | 204 |
| 0xD94 | MPUCTRL | RW | 0x0000.0000 | MPU Control | 205 |
| 0xD98 | MPUNUMBER | RW | 0x0000.0000 | MPU Region Number | 207 |
| 0xD9C | MPUBASE | RW | 0x0000.0000 | MPU Region Base Address | 208 |
| 0xDA0 | MPUATTR | RW | 0x0000.0000 | MPU Region Attribute and Size | 210 |
| 0xDA4 | MPUBASE1 | RW | 0x0000.0000 | MPU Region Base Address Alias 1 | 208 |
| 0xDA8 | MPUATTR1 | RW | 0x0000.0000 | MPU Region Attribute and Size Alias 1 | 210 |
| 0xDAC | MPUBASE2 | RW | 0x0000.0000 | MPU Region Base Address Alias 2 | 208 |
| 0xDB0 | MPUATTR2 | RW | 0x0000.0000 | MPU Region Attribute and Size Alias 2 | 210 |

**Table 3-8. Peripherals Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xDB4 | MPUBASE3 | RW | 0x0000.0000 | MPU Region Base Address Alias 3 | 208 |
| 0xDB8 | MPUATTR3 | RW | 0x0000.0000 | MPU Region Attribute and Size Alias 3 | 210 |
| **Floating-Point Unit (FPU) Registers** | | | | | |
| 0xD88 | CPAC | RW | 0x0000.0000 | Coprocessor Access Control | 213 |
| 0xF34 | FPCC | RW | 0xC000.0000 | Floating-Point Context Control | 214 |
| 0xF38 | FPCA | RW | - | Floating-Point Context Address | 216 |
| 0xF3C | FPDSC | RW | 0x0000.0000 | Floating-Point Default Status Control | 217 |

## 3.3 System Timer (SysTick) Register Descriptions

This section lists and describes the System Timer registers, in numerical order by address offset.

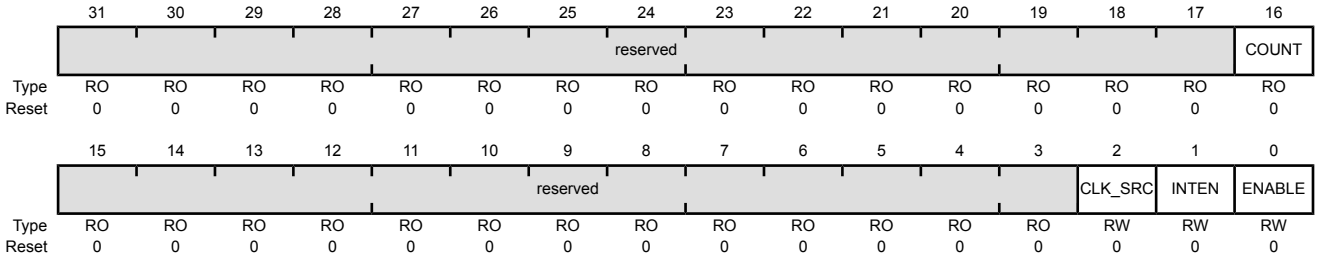
## Register 1: SysTick Control and Status Register (STCTRL), offset 0x010

**Note:** This register can only be accessed from privileged mode.

The SysTick **STCTRL** register enables the SysTick features.

SysTick Control and Status Register (STCTRL)

Base 0xE000.E000
Offset 0x010
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0x000 | 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. |
| 16 | COUNT | RO | 0 | Count Flag<br><br>Value Description<br>0 The SysTick timer has not counted to 0 since the last time this bit was read.<br>1 The SysTick timer has counted to 0 since the last time this bit was read.<br><br>This bit is cleared by a read of the register or if the **STCURRENT** register is written with any value.<br><br>If read by the debugger using the DAP, this bit is cleared only if the `MasterType` bit in the **AHB-AP Control Register** is clear. Otherwise, the `COUNT` bit is not changed by the debugger read. See the *ARM® Debug Interface V5 Architecture Specification* for more information on `MasterType`. |
| 15:3 | reserved | RO | 0x000 | 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. |
| 2 | CLK_SRC | RW | 0 | Clock Source<br><br>Value Description<br>0 Precision internal oscillator (PIOSC) divided by 4<br>1 System clock |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INTEN | RW | 0 | Interrupt Enable |

| Value | Description |
|---|---|
| 0 | Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0. |
| 1 | An interrupt is generated to the NVIC when SysTick counts to 0. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ENABLE | RW | 0 | Enable |

| Value | Description |
|---|---|
| 0 | The counter is disabled. |
| 1 | Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting. |

## Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014

**Note:** This register can only be accessed from privileged mode.

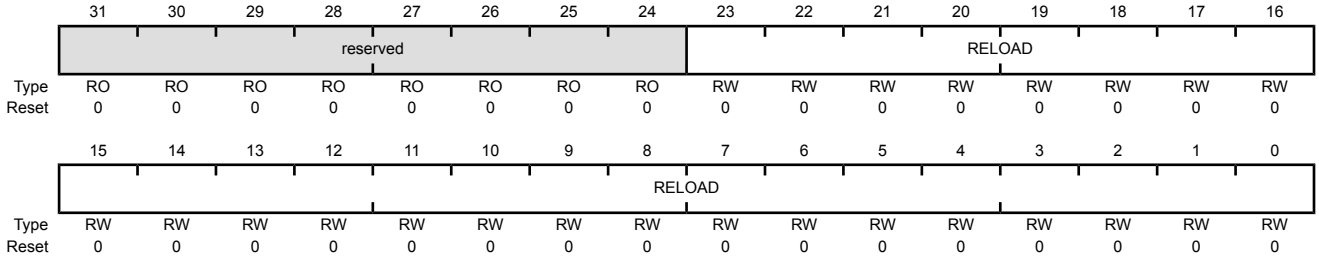
The **STRELOAD** register specifies the start value to load into the **SysTick Current Value (STCURRENT)** register when the counter reaches 0. The start value can be between 0x1 and 0x00FF.FFFF. A start value of 0 is possible but has no effect because the SysTick interrupt and the COUNT bit are activated when counting from 1 to 0.

SysTick can be configured as a multi-shot timer, repeated over and over, firing every N+1 clock pulses, where N is any value from 1 to 0x00FF.FFFF. For example, if a tick interrupt is required every 100 clock pulses, 99 must be written into the RELOAD field.

Note that in order to access this register correctly, the system clock must be faster than 8 MHz.

SysTick Reload Value Register (STRELOAD)

Base 0xE000.E000
Offset 0x014
Type RW, reset -



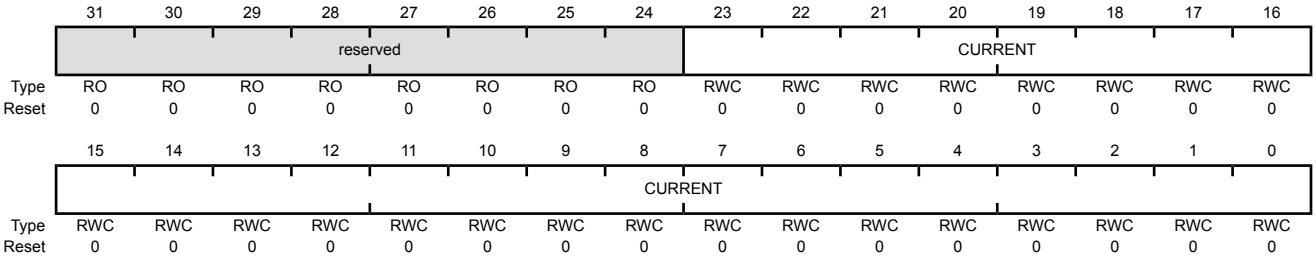

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 23:0 | RELOAD | RW | 0x00.0000 | Reload Value<br>Value to load into the **SysTick Current Value (STCURRENT)** register when the counter reaches 0. |

### Register 3: SysTick Current Value Register (STCURRENT), offset 0x018

**Note:** This register can only be accessed from privileged mode.

The **STCURRENT** register contains the current value of the SysTick counter.

SysTick Current Value Register (STCURRENT)
Base 0xE000.E000
Offset 0x018
Type RWC, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x00 | 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. |
| 23:0 | CURRENT | RWC | 0x00.0000 | Current Value<br>This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care.<br>This register is write-clear. Writing to it with any value clears the register. Clearing this register also clears the COUNT bit of the **STCTRL** register. |

## 3.4 NVIC Register Descriptions

This section lists and describes the NVIC registers, in numerical order by address offset.

The NVIC registers can only be fully accessed from privileged mode, but interrupts can be pended while in unprivileged mode by enabling the **Configuration and Control (CFGCTRL)** register. Any other unprivileged mode access causes a bus fault.

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers.

An interrupt can enter the pending state even if it is disabled.

Before programming the **VTABLE** register to relocate the vector table, ensure the vector table entries of the new vector table are set up for fault handlers, NMI, and all enabled exceptions such as interrupts. For more information, see page 181.

## Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100

## Register 5: Interrupt 32-63 Set Enable (EN1), offset 0x104

## Register 6: Interrupt 64-95 Set Enable (EN2), offset 0x108

## Register 7: Interrupt 96-113 Set Enable (EN3), offset 0x10C

**Note:** This register can only be accessed from privileged mode.

The **ENn** registers enable interrupts and show which interrupts are enabled. Bit 0 of **EN0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **EN1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **EN2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **EN3** corresponds to Interrupt 96; bit 17 corresponds to Interrupt 113.

See Table 2-9 on page 126 for interrupt assignments.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

Interrupt 0-31 Set Enable (EN0)

Base 0xE000.E000
Offset 0x100
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INT | | | | | | | | | | | | | | | |
| Type RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INT | | | | | | | | | | | | | | | |
| Type RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | RW | 0x0000.0000 | Interrupt Enable |

| Value | Description |
|---|---|
| 0 | On a read, indicates the interrupt is disabled.<br>On a write, no effect. |
| 1 | On a read, indicates the interrupt is enabled.<br>On a write, enables the interrupt. |

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **DISn** register.

### Register 8: Interrupt 0-31 Clear Enable (DIS0), offset 0x180

### Register 9: Interrupt 32-63 Clear Enable (DIS1), offset 0x184

### Register 10: Interrupt 64-95 Clear Enable (DIS2), offset 0x188

### Register 11: Interrupt 96-113 Clear Enable (DIS3), offset 0x18C

**Note:** This register can only be accessed from privileged mode.

The **DISn** registers disable interrupts. Bit 0 of **DIS0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **DIS1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **DIS2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **DIS3** corresponds to Interrupt 96; .

See Table 2-9 on page 126 for interrupt assignments.

Interrupt 0-31 Clear Enable (DIS0)

Base 0xE000.E000
Offset 0x180
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | RW | 0x0000.0000 | Interrupt Disable |

| Value | Description |
|---|---|
| 0 | On a read, indicates the interrupt is disabled.<br>On a write, no effect. |
| 1 | On a read, indicates the interrupt is enabled.<br>On a write, clears the corresponding `INT[n]` bit in the **EN0** register, disabling interrupt [n]. |

## Register 12: Interrupt 0-31 Set Pending (PEND0), offset 0x200

## Register 13: Interrupt 32-63 Set Pending (PEND1), offset 0x204

## Register 14: Interrupt 64-95 Set Pending (PEND2), offset 0x208

## Register 15: Interrupt 96-113 Set Pending (PEND3), offset 0x20C

**Note:** This register can only be accessed from privileged mode.

The **PENDn** registers force interrupts into the pending state and show which interrupts are pending. Bit 0 of **PEND0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **PEND1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **PEND2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **PEND3** corresponds to Interrupt 96; bit 17 corresponds to interrupt 113.

See Table 2-9 on page 126 for interrupt assignments.

Interrupt 0-31 Set Pending (PEND0)

Base 0xE000.E000
Offset 0x200
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INT | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | RW | 0x0000.0000 | Interrupt Set Pending |

| Value | Description |
|---|---|
| 0 | On a read, indicates that the interrupt is not pending.<br>On a write, no effect. |
| 1 | On a read, indicates that the interrupt is pending.<br>On a write, the corresponding interrupt is set to pending even if it is disabled. |

If the corresponding interrupt is already pending, setting a bit has no effect.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **UNPEND0** register.

### Register 16: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280

### Register 17: Interrupt 32-63 Clear Pending (UNPEND1), offset 0x284

### Register 18: Interrupt 64-95 Clear Pending (UNPEND2), offset 0x288

### Register 19: Interrupt 96-113 Clear Pending (UNPEND3), offset 0x28C

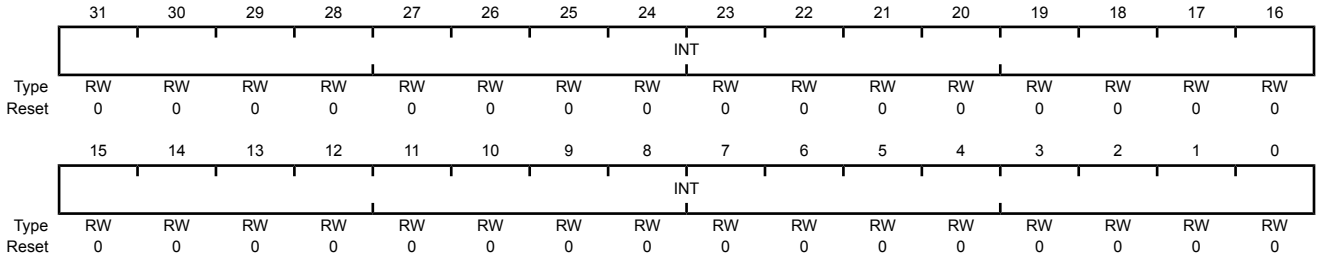
**Note:** This register can only be accessed from privileged mode.

The **UNPENDn** registers show which interrupts are pending and remove the pending state from interrupts. Bit 0 of **UNPEND0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **UNPEND1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **UNPEND2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **UNPEND3** corresponds to Interrupt 96; bit 31 corresponds to Interrupt 113.

See Table 2-9 on page 126 for interrupt assignments.

Interrupt 0-31 Clear Pending (UNPEND0)

Base 0xE000.E000
Offset 0x280
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | RW | 0x0000.0000 | Interrupt Clear Pending |

| Value | Description |
|---|---|
| 0 | On a read, indicates that the interrupt is not pending.<br>On a write, no effect. |
| 1 | On a read, indicates that the interrupt is pending.<br>On a write, clears the corresponding `INT[n]` bit in the **PEND0** register, so that interrupt [n] is no longer pending.<br>Setting a bit does not affect the active state of the corresponding interrupt. |

## Register 20: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300

## Register 21: Interrupt 32-63 Active Bit (ACTIVE1), offset 0x304

## Register 22: Interrupt 64-95 Active Bit (ACTIVE2), offset 0x308

## Register 23: Interrupt 96-127 Active Bit (ACTIVE3), offset 0x30C

**Note:** This register can only be accessed from privileged mode.

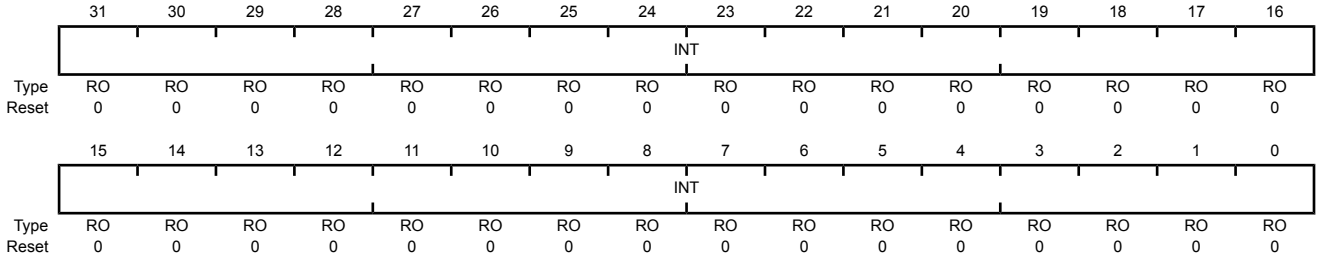
The **UNPENDn** registers indicate which interrupts are active. Bit 0 of **ACTIVE0** corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31. Bit 0 of **ACTIVE1** corresponds to Interrupt 32; bit 31 corresponds to Interrupt 63. Bit 0 of **ACTIVE2** corresponds to Interrupt 64; bit 31 corresponds to Interrupt 95. Bit 0 of **ACTIVE3** corresponds to Interrupt 96; bit 17 corresponds to Interrupt 113.

See Table 2-9 on page 126 for interrupt assignments.

**Caution – Do not manually set or clear the bits in this register.**

Interrupt 0-31 Active Bit (ACTIVE0)

Base 0xE000.E000
Offset 0x300
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | RO | 0x0000.0000 | Interrupt Active |

| Value | Description |
|---|---|
| 0 | The corresponding interrupt is not active. |
| 1 | The corresponding interrupt is active, or active and pending. |

### Register 24: Interrupt 0-3 Priority (PRI0), offset 0x400
### Register 25: Interrupt 4-7 Priority (PRI1), offset 0x404
### Register 26: Interrupt 8-11 Priority (PRI2), offset 0x408
### Register 27: Interrupt 12-15 Priority (PRI3), offset 0x40C
### Register 28: Interrupt 16-19 Priority (PRI4), offset 0x410
### Register 29: Interrupt 20-23 Priority (PRI5), offset 0x414
### Register 30: Interrupt 24-27 Priority (PRI6), offset 0x418
### Register 31: Interrupt 28-31 Priority (PRI7), offset 0x41C
### Register 32: Interrupt 32-35 Priority (PRI8), offset 0x420
### Register 33: Interrupt 36-39 Priority (PRI9), offset 0x424
### Register 34: Interrupt 40-43 Priority (PRI10), offset 0x428
### Register 35: Interrupt 44-47 Priority (PRI11), offset 0x42C
### Register 36: Interrupt 48-51 Priority (PRI12), offset 0x430
### Register 37: Interrupt 52-55 Priority (PRI13), offset 0x434
### Register 38: Interrupt 56-59 Priority (PRI14), offset 0x438
### Register 39: Interrupt 60-63 Priority (PRI15), offset 0x43C

**Note:** This register can only be accessed from privileged mode.

The **PRIn** registers (see also page 172) provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

| PRIn Register Bit Field | Interrupt |
|---|---|
| Bits 31:29 | Interrupt [4n+3] |
| Bits 23:21 | Interrupt [4n+2] |
| Bits 15:13 | Interrupt [4n+1] |
| Bits 7:5 | Interrupt [4n] |

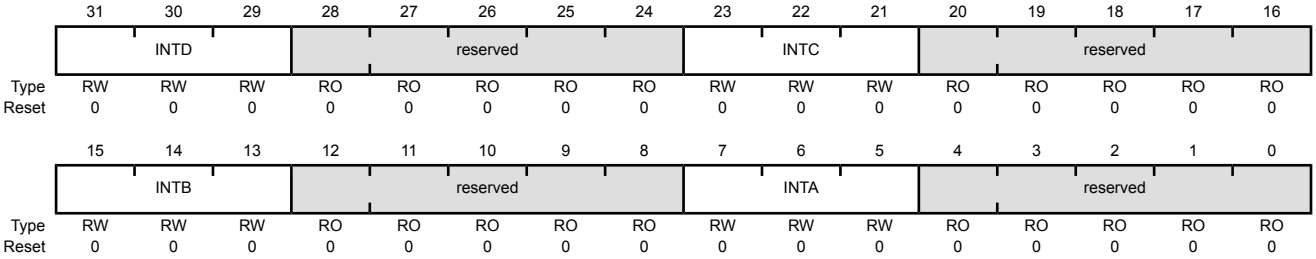
See Table 2-9 on page 126 for interrupt assignments.

Each priority level can be split into separate group priority and subpriority fields. The PRIGROUP field in the **Application Interrupt and Reset Control (APINT)** register (see page 182) indicates the position of the binary point that splits the priority and subpriority fields.

These registers can only be accessed from privileged mode.

### Interrupt 0-3 Priority (PRI0)

Base 0xE000.E000
Offset 0x400
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | INTD | RW | 0x0 | Interrupt Priority for Interrupt [4n+3]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 28:24 | reserved | RO | 0x0 | 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. |
| 23:21 | INTC | RW | 0x0 | Interrupt Priority for Interrupt [4n+2]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 20:16 | reserved | RO | 0x0 | 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. |
| 15:13 | INTB | RW | 0x0 | Interrupt Priority for Interrupt [4n+1]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 12:8 | reserved | RO | 0x0 | 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. |
| 7:5 | INTA | RW | 0x0 | Interrupt Priority for Interrupt [4n]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 4:0 | reserved | RO | 0x0 | 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. |

### Register 40: Interrupt 64-67 Priority (PRI16), offset 0x440

### Register 41: Interrupt 68-71 Priority (PRI17), offset 0x444

### Register 42: Interrupt 72-75 Priority (PRI18), offset 0x448

### Register 43: Interrupt 76-79 Priority (PRI19), offset 0x44C

### Register 44: Interrupt 80-83 Priority (PRI20), offset 0x450

### Register 45: Interrupt 84-87 Priority (PRI21), offset 0x454

### Register 46: Interrupt 88-91 Priority (PRI22), offset 0x458

### Register 47: Interrupt 92-95 Priority (PRI23), offset 0x45C

### Register 48: Interrupt 96-99 Priority (PRI24), offset 0x460

### Register 49: Interrupt 100-103 Priority (PRI25), offset 0x464

### Register 50: Interrupt 104-107 Priority (PRI26), offset 0x468

### Register 51: Interrupt 108-111 Priority (PRI27), offset 0x46C

### Register 52: Interrupt 112-113 Priority (PRI28), offset 0x470

**Note:** This register can only be accessed from privileged mode.

The **PRIn** registers (see also page 170) provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

| PRIn Register Bit Field | Interrupt |
|---|---|
| Bits 31:29 | Interrupt [4n+3] |
| Bits 23:21 | Interrupt [4n+2] |
| Bits 15:13 | Interrupt [4n+1] |
| Bits 7:5 | Interrupt [4n] |

See Table 2-9 on page 126 for interrupt assignments.

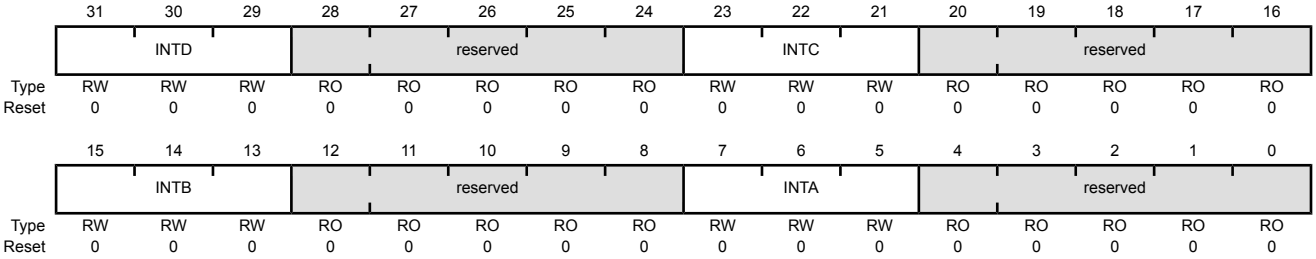
Each priority level can be split into separate group priority and subpriority fields. The PRIGROUP field in the **Application Interrupt and Reset Control (APINT)** register (see page 182) indicates the position of the binary point that splits the priority and subpriority fields .

These registers can only be accessed from privileged mode.

**Note:** Because the last interrupt vector is number 113, bits [31:16] of the **PRI28** register are reserved.

Interrupt 64-67 Priority (PRI16)

Base 0xE000.E000
Offset 0x440
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | INTD | RW | 0x0 | Interrupt Priority for Interrupt [4n+3]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 28:24 | reserved | RO | 0x0 | 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. |
| 23:21 | INTC | RW | 0x0 | Interrupt Priority for Interrupt [4n+2]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 20:16 | reserved | RO | 0x0 | 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. |
| 15:13 | INTB | RW | 0x0 | Interrupt Priority for Interrupt [4n+1]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 12:8 | reserved | RO | 0x0 | 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. |
| 7:5 | INTA | RW | 0x0 | Interrupt Priority for Interrupt [4n]<br>This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the **Interrupt Priority** register (n=0 for **PRI0**, and so on). The lower the value, the greater the priority of the corresponding interrupt. |
| 4:0 | reserved | RO | 0x0 | 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. |

### Register 53: Software Trigger Interrupt (SWTRIG), offset 0xF00

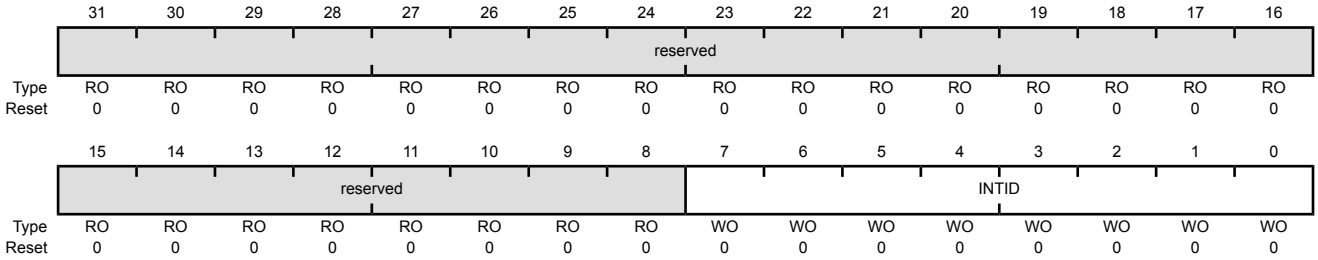
**Note:** Only privileged software can enable unprivileged access to the **SWTRIG** register.

Writing an interrupt number to the **SWTRIG** register generates a Software Generated Interrupt (SGI). See Table 2-9 on page 126 for interrupt assignments.

When the MAINPEND bit in the **Configuration and Control (CFGCTRL)** register (see page 186) is set, unprivileged software can access the **SWTRIG** register.

Software Trigger Interrupt (SWTRIG)

Base 0xE000.E000
Offset 0xF00
Type WO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | INTID | WO | 0x00 | Interrupt ID<br>This field holds the interrupt ID of the required SGI. For example, a value of 0x3 generates an interrupt on IRQ3. |

## 3.5 System Control Block (SCB) Register Descriptions

This section lists and describes the System Control Block (SCB) registers, in numerical order by address offset. The SCB registers can only be accessed from privileged mode.

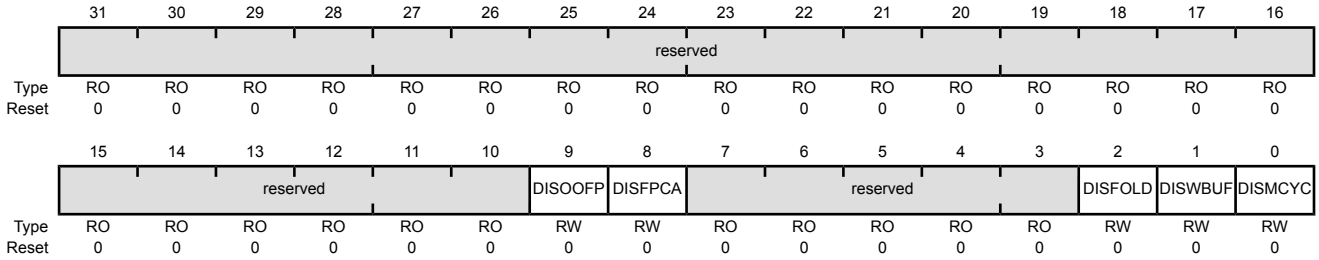
All registers must be accessed with aligned word accesses except for the **FAULTSTAT** and **SYSPRI1**-**SYSPRI3** registers, which can be accessed with byte or aligned halfword or word accesses. The processor does not support unaligned accesses to system control block registers.

## Register 54: Auxiliary Control (ACTLR), offset 0x008

**Note:** This register can only be accessed from privileged mode.

The **ACTLR** register provides disable bits for IT folding, write buffer use for accesses to the default memory map, and interruption of multi-cycle instructions. By default, this register is set to provide optimum performance from the Cortex-M4 processor and does not normally require modification.

Auxiliary Control (ACTLR)
Base 0xE000.E000
Offset 0x008
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x00 | 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. |
| 9 | DISOOFP | RW | 0 | Disable Out-Of-Order Floating Point<br>Disables floating-point instructions completing out of order with respect to integer instructions. |
| 8 | DISFPCA | RW | 0 | Disable CONTROL.FPCA<br>Disable automatic update of the FPCA bit in the **CONTROL** register.<br>**Important:** Two bits control when FPCA can be enabled: the ASPEN bit in the **Floating-Point Context Control (FPCC)** register and the DISFPCA bit in the **Auxiliary Control (ACTLR)** register. |
| 7:3 | reserved | RO | 0x00 | 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. |
| 2 | DISFOLD | RW | 0 | Disable IT Folding<br>Value Description<br>0 No effect.<br>1 Disables IT folding.<br>In some situations, the processor can start executing the first instruction in an IT block while it is still executing the IT instruction. This behavior is called *IT folding*, and improves performance, However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit before executing the task, to disable IT folding. |

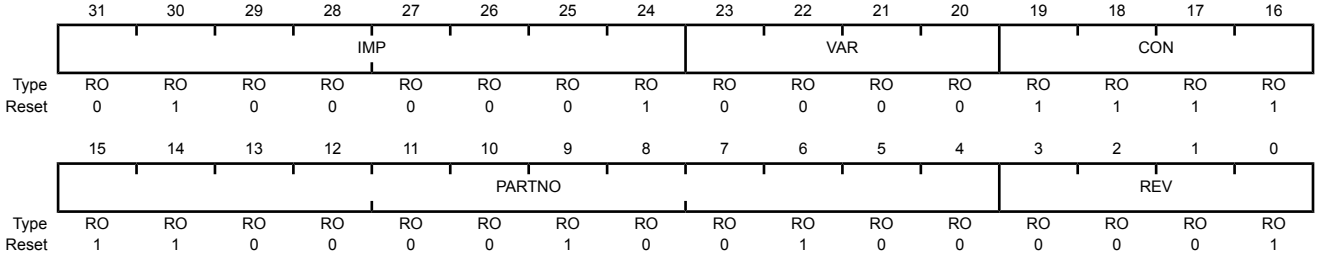
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | DISWBUF | RW | 0 | Disable Write Buffer<br><br>Value Description<br>0 No effect.<br>1 Disables write buffer use during default memory map accesses. In this situation, all bus faults are precise bus faults but performance is decreased because any store to memory must complete before the processor can execute the next instruction.<br><br>**Note:** This bit only affects write buffers implemented in the Cortex-M4 processor. |
| 0 | DISMCYC | RW | 0 | Disable Interrupts of Multiple Cycle Instructions<br><br>Value Description<br>0 No effect.<br>1 Disables interruption of load multiple and store multiple instructions. In this situation, the interrupt latency of the processor is increased because any `LDM` or `STM` must complete before the processor can stack the current state and enter the interrupt handler. |

## Register 55: CPU ID Base (CPUID), offset 0xD00

**Note:** This register can only be accessed from privileged mode.

The **CPUID** register contains the ARM® Cortex™-M4 processor part number, version, and implementation information.

CPU ID Base (CPUID)
Base 0xE000.E000
Offset 0xD00
Type RO, reset 0x410F.C241




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | IMP | RO | 0x41 | Implementer Code<br><br>Value Description<br>0x41 ARM |
| 23:20 | VAR | RO | 0x0 | Variant Number<br><br>Value Description<br>0x0 The rn value in the rnpn product revision identifier, for example, the 0 in r0p0. |
| 19:16 | CON | RO | 0xF | Constant<br><br>Value Description<br>0xF Always reads as 0xF. |
| 15:4 | PARTNO | RO | 0xC24 | Part Number<br><br>Value Description<br>0xC24 Cortex-M4 processor. |
| 3:0 | REV | RO | 0x1 | Revision Number<br><br>Value Description<br>0x1 The pn value in the rnpn product revision identifier, for example, the 1 in r0p1. |

## Register 56: Interrupt Control and State (INTCTRL), offset 0xD04

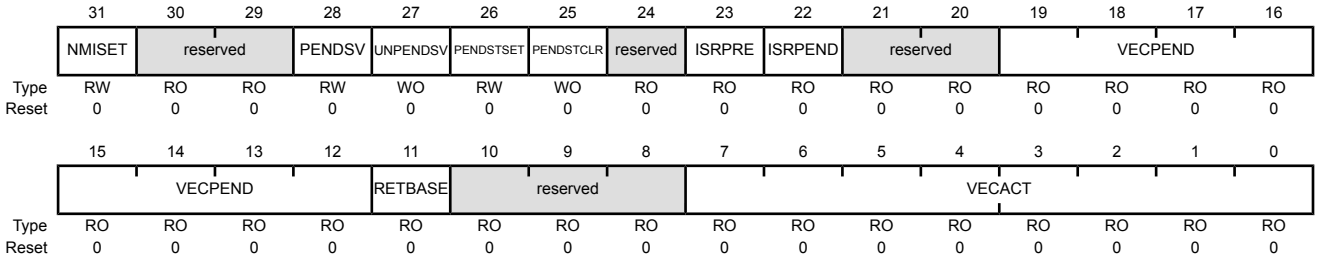
**Note:** This register can only be accessed from privileged mode.

The **INCTRL** register provides a set-pending bit for the NMI exception, and set-pending and clear-pending bits for the PendSV and SysTick exceptions. In addition, bits in this register indicate the exception number of the exception being processed, whether there are preempted active exceptions, the exception number of the highest priority pending exception, and whether any interrupts are pending.

When writing to **INCTRL**, the effect is unpredictable when writing a 1 to both the PENDSV and UNPENDSV bits, or writing a 1 to both the PENDSTSET and PENDSTCLR bits.

Interrupt Control and State (INTCTRL)

Base 0xE000.E000
Offset 0xD04
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NMISET | RW | 0 | NMI Set Pending<br><br>Value Description<br>0 On a read, indicates an NMI exception is not pending.<br>On a write, no effect.<br>1 On a read, indicates an NMI exception is pending.<br>On a write, changes the NMI exception state to pending.<br><br>Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it registers the setting of this bit, and clears this bit on entering the interrupt handler. A read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler. |
| 30:29 | reserved | RO | 0x0 | 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. |
| 28 | PENDSV | RW | 0 | PendSV Set Pending<br><br>Value Description<br>0 On a read, indicates a PendSV exception is not pending.<br>On a write, no effect.<br>1 On a read, indicates a PendSV exception is pending.<br>On a write, changes the PendSV exception state to pending.<br><br>Setting this bit is the only way to set the PendSV exception state to pending. This bit is cleared by writing a 1 to the UNPENDSV bit. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 27 | UNPENDSV | WO | 0 | PendSV Clear Pending<br><br>Value Description<br>0 On a write, no effect.<br>1 On a write, removes the pending state from the PendSV exception.<br><br>This bit is write only; on a register read, its value is unknown. |
| 26 | PENDSTSET | RW | 0 | SysTick Set Pending<br><br>Value Description<br>0 On a read, indicates a SysTick exception is not pending.<br>On a write, no effect.<br>1 On a read, indicates a SysTick exception is pending.<br>On a write, changes the SysTick exception state to pending.<br><br>This bit is cleared by writing a 1 to the PENDSTCLR bit. |
| 25 | PENDSTCLR | WO | 0 | SysTick Clear Pending<br><br>Value Description<br>0 On a write, no effect.<br>1 On a write, removes the pending state from the SysTick exception.<br><br>This bit is write only; on a register read, its value is unknown. |
| 24 | reserved | RO | 0 | 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. |
| 23 | ISRPRE | RO | 0 | Debug Interrupt Handling<br><br>Value Description<br>0 The release from halt does not take an interrupt.<br>1 The release from halt takes an interrupt.<br><br>This bit is only meaningful in Debug mode and reads as zero when the processor is not in Debug mode. |
| 22 | ISRPEND | RO | 0 | Interrupt Pending<br><br>Value Description<br>0 No interrupt is pending.<br>1 An interrupt is pending.<br><br>This bit provides status for all interrupts excluding NMI and Faults. |
| 21:20 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 19:12 | VECPEND | RO | 0x00 | Interrupt Pending Vector Number<br>This field contains the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the **BASEPRI** and **FAULTMASK** registers, but not any effect of the **PRIMASK** register.<br>Value / Description<br>0x00 No exceptions are pending<br>0x01 Reserved<br>0x02 NMI<br>0x03 Hard fault<br>0x04 Memory management fault<br>0x05 Bus fault<br>0x06 Usage fault<br>0x07-0x0A Reserved<br>0x0B SVCall<br>0x0C Reserved for Debug<br>0x0D Reserved<br>0x0E PendSV<br>0x0F SysTick<br>0x10 Interrupt Vector 0<br>0x11 Interrupt Vector 1<br>... ...<br>0xD9 Interrupt Vector 199 |
| 11 | RETBASE | RO | 0 | Return to Base<br>Value / Description<br>0 There are preempted active exceptions to execute.<br>1 There are no active exceptions, or the currently executing exception is the only active exception.<br>This bit provides status for all interrupts excluding NMI and Faults. This bit only has meaning if the processor is currently executing an ISR (the **Interrupt Program Status (IPSR)** register is non-zero). |
| 10:8 | reserved | RO | 0x0 | 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. |
| 7:0 | VECACT | RO | 0x00 | Interrupt Pending Vector Number<br>This field contains the active exception number. The exception numbers can be found in the description for the VECPEND field. If this field is clear, the processor is in Thread mode. This field contains the same value as the ISRNUM field in the **IPSR** register.<br>Subtract 16 from this value to obtain the IRQ number required to index into the **Interrupt Set Enable (ENn)**, **Interrupt Clear Enable (DISn**), **Interrupt Set Pending (PENDn)**, **Interrupt Clear Pending (UNPENDn)**, and **Interrupt Priority (PRIn)** registers (see page 102). |

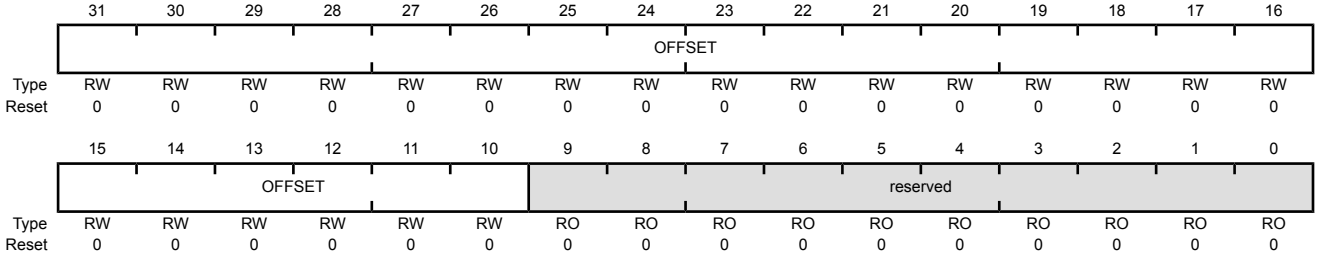
## Register 57: Vector Table Offset (VTABLE), offset 0xD08

**Note:** This register can only be accessed from privileged mode.

The **VTABLE** register indicates the offset of the vector table base address from memory address 0x0000.0000.

Vector Table Offset (VTABLE)

Base 0xE000.E000
Offset 0xD08
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | OFFSET | RW | 0x000.00 | Vector Table Offset<br>When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 112 interrupts, the offset must be aligned on a 1024-byte boundary. |
| 9:0 | reserved | RO | 0x00 | 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. |

## Register 58: Application Interrupt and Reset Control (APINT), offset 0xD0C

**Note:** This register can only be accessed from privileged mode.

The **APINT** register provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. To write to this register, 0x05FA must be written to the VECTKEY field, otherwise the write is ignored.

The PRIGROUP field indicates the position of the binary point that splits the INTx fields in the **Interrupt Priority (PRIx)** registers into separate group priority and subpriority fields. Table 3-9 on page 182 shows how the PRIGROUP value controls this split. The bit numbers in the Group Priority Field and Subpriority Field columns in the table refer to the bits in the INTA field. For the INTB field, the corresponding bits are 15:13; for INTC, 23:21; and for INTD, 31:29.

**Note:** Determining preemption of an exception uses only the group priority field.

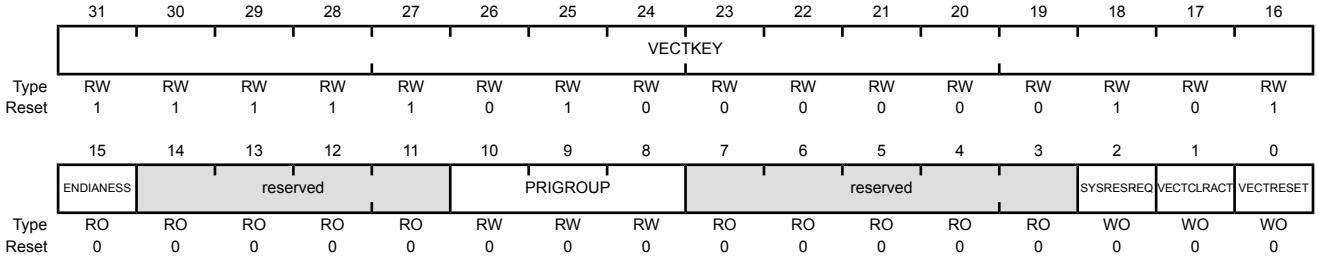
**Table 3-9. Interrupt Priority Levels**

| PRIGROUP Bit Field | Binary Point[a] | Group Priority Field | Subpriority Field | Group Priorities | Subpriorities |
|---|---|---|---|---|---|
| 0x0 - 0x4 | bxxx. | [7:5] | None | 8 | 1 |
| 0x5 | bxx.y | [7:6] | [5] | 4 | 2 |
| 0x6 | bx.yy | [7] | [6:5] | 2 | 4 |
| 0x7 | b.yyy | None | [7:5] | 1 | 8 |

a. INTx field showing the binary point. An x denotes a group priority field bit, and a y denotes a subpriority field bit.

Application Interrupt and Reset Control (APINT)

Base 0xE000.E000
Offset 0xD0C
Type RW, reset 0xFA05.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | VECTKEY | RW | 0xFA05 | Register Key<br>This field is used to guard against accidental writes to this register. 0x05FA must be written to this field in order to change the bits in this register. On a read, 0xFA05 is returned. |
| 15 | ENDIANESS | RO | 0 | Data Endianess<br>The Tiva™ C Series implementation uses only little-endian mode so this is cleared to 0. |
| 14:11 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10:8 | PRIGROUP | RW | 0x0 | Interrupt Priority Grouping<br>This field determines the split of group priority from subpriority (see Table 3-9 on page 182 for more information). |
| 7:3 | reserved | RO | 0x0 | 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. |
| 2 | SYSRESREQ | WO | 0 | System Reset Request<br>Value Description<br>0 No effect.<br>1 Resets the core and all on-chip peripherals except the Debug interface.<br>This bit is automatically cleared during the reset of the core and reads as 0. |
| 1 | VECTCLRACT | WO | 0 | Clear Active NMI / Fault<br>This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable. |
| 0 | VECTRESET | WO | 0 | System Reset<br>This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable. |

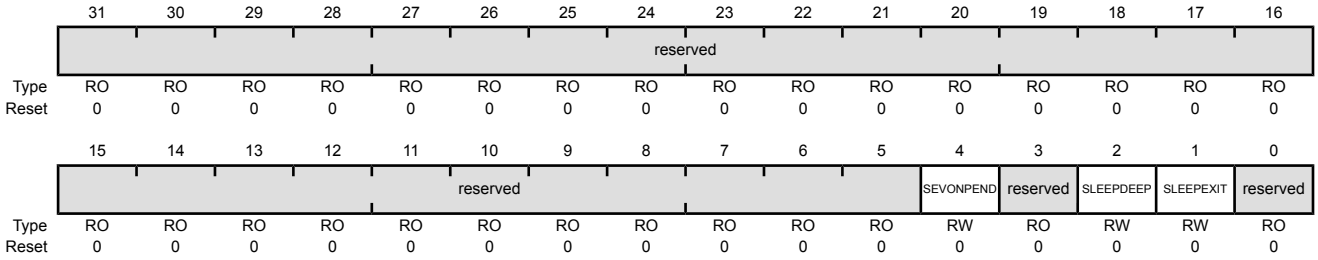
## Register 59: System Control (SYSCTRL), offset 0xD10

**Note:** This register can only be accessed from privileged mode.

The **SYSCTRL** register controls features of entry to and exit from low-power state.

System Control (SYSCTRL)

Base 0xE000.E000
Offset 0xD10
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x0000.00 | 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. |
| 4 | SEVONPEND | RW | 0 | Wake Up on Pending<br><br>Value Description<br>0 Only enabled interrupts or events can wake up the processor; disabled interrupts are excluded.<br>1 Enabled events and all interrupts, including disabled interrupts, can wake up the processor.<br><br>When an event or interrupt enters the 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.<br>The processor also wakes up on execution of a SEV instruction or an external event. |
| 3 | reserved | RO | 0 | 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. |
| 2 | SLEEPDEEP | RW | 0 | Deep Sleep Enable<br><br>Value Description<br>0 Use Sleep mode as the low power mode.<br>1 Use Deep-sleep mode as the low power mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SLEEPEXIT | RW | 0 | Sleep on ISR Exit<br><br>Value Description<br>0 When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode.<br>1 When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR.<br><br>Setting this bit enables an interrupt-driven application to avoid returning to an empty main application. |
| 0 | reserved | RO | 0 | 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. |

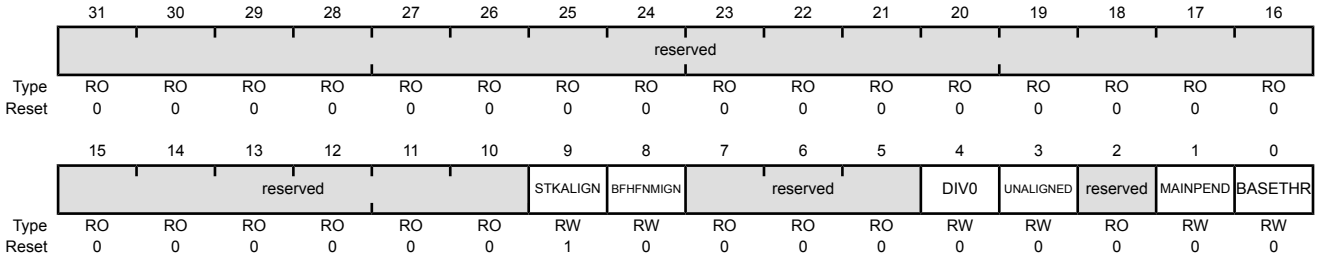
## Register 60: Configuration and Control (CFGCTRL), offset 0xD14

**Note:** This register can only be accessed from privileged mode.

The **CFGCTRL** register controls entry to Thread mode and enables: the handlers for NMI, hard fault and faults escalated by the **FAULTMASK** register to ignore bus faults; trapping of divide by zero and unaligned accesses; and access to the **SWTRIG** register by unprivileged software (see page 174).

Configuration and Control (CFGCTRL)

Base 0xE000.E000
Offset 0xD14
Type RW, reset 0x0000.0200




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9 | STKALIGN | RW | 1 | Stack Alignment on Exception Entry<br><br>Value Description<br>0 The stack is 4-byte aligned.<br>1 The stack is 8-byte aligned.<br><br>On exception entry, the processor uses bit 9 of the stacked **PSR** to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment. |
| 8 | BFHFNMIGN | RW | 0 | Ignore Bus Fault in NMI and Fault<br><br>This bit enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. The setting of this bit applies to the hard fault, NMI, and **FAULTMASK** escalated handlers.<br><br>Value Description<br>0 Data bus faults caused by load and store instructions cause a lock-up.<br>1 Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions.<br><br>Set this bit 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 control path problems and fix them. |
| 7:5 | reserved | RO | 0x0 | 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. |

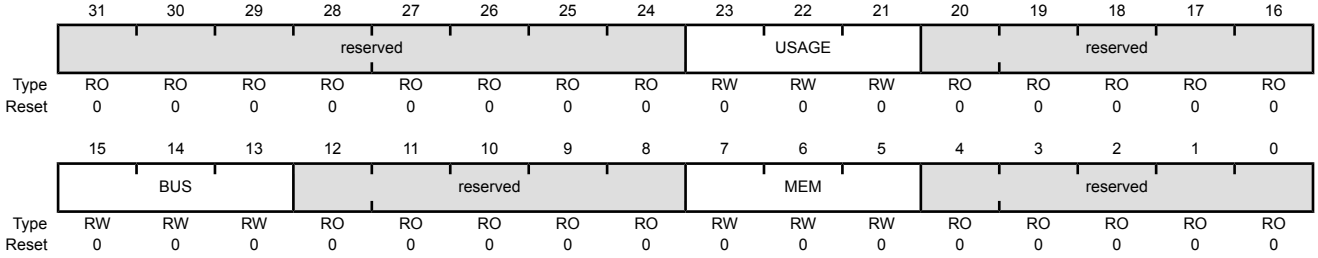
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | DIV0 | RW | 0 | Trap on Divide by 0<br>This bit enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0.<br><br>Value Description<br>0 Do not trap on divide by 0. A divide by zero returns a quotient of 0.<br>1 Trap on divide by 0. |
| 3 | UNALIGNED | RW | 0 | Trap on Unaligned Access<br><br>Value Description<br>0 Do not trap on unaligned halfword and word accesses.<br>1 Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault.<br><br>Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of whether UNALIGNED is set. |
| 2 | reserved | RO | 0 | 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. |
| 1 | MAINPEND | RW | 0 | Allow Main Interrupt Trigger<br><br>Value Description<br>0 Disables unprivileged software access to the **SWTRIG** register.<br>1 Enables unprivileged software access to the **SWTRIG** register (see page 174). |
| 0 | BASETHR | RW | 0 | Thread State Control<br><br>Value Description<br>0 The processor can enter Thread mode only when no exception is active.<br>1 The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see “Exception Return” on page 133 for more information). |

## Register 61: System Handler Priority 1 (SYSPRI1), offset 0xD18

**Note:** This register can only be accessed from privileged mode.

The **SYSPRI1** register configures the priority level, 0 to 7 of the usage fault, bus fault, and memory management fault exception handlers. This register is byte-accessible.

System Handler Priority 1 (SYSPRI1)
Base 0xE000.E000
Offset 0xD18
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x00 | 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. |
| 23:21 | USAGE | RW | 0x0 | Usage Fault Priority<br>This field configures the priority level of the usage fault. Configurable priority values are in the range 0-7, with lower values having higher priority. |
| 20:16 | reserved | RO | 0x0 | 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. |
| 15:13 | BUS | RW | 0x0 | Bus Fault Priority<br>This field configures the priority level of the bus fault. Configurable priority values are in the range 0-7, with lower values having higher priority. |
| 12:8 | reserved | RO | 0x0 | 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. |
| 7:5 | MEM | RW | 0x0 | Memory Management Fault Priority<br>This field configures the priority level of the memory management fault. Configurable priority values are in the range 0-7, with lower values having higher priority. |
| 4:0 | reserved | RO | 0x0 | 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. |

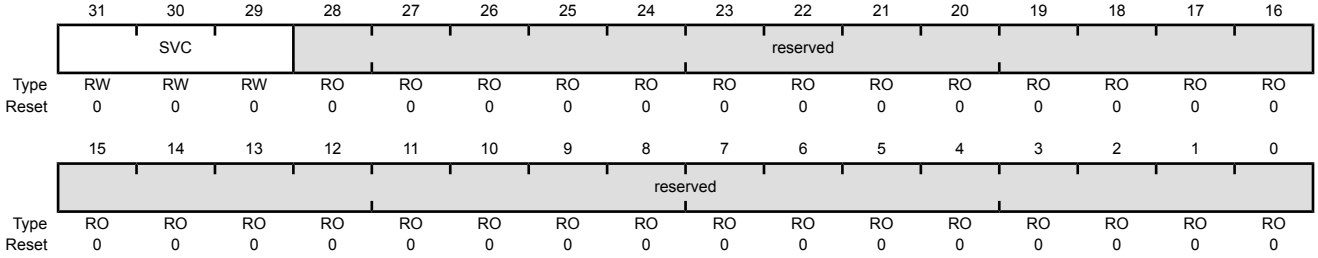
## Register 62: System Handler Priority 2 (SYSPRI2), offset 0xD1C

**Note:** This register can only be accessed from privileged mode.

The **SYSPRI2** register configures the priority level, 0 to 7 of the SVCall handler. This register is byte-accessible.

System Handler Priority 2 (SYSPRI2)

Base 0xE000.E000
Offset 0xD1C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | SVC | RW | 0x0 | SVCall Priority<br>This field configures the priority level of SVCall. Configurable priority values are in the range 0-7, with lower values having higher priority. |
| 28:0 | reserved | RO | 0x000.0000 | 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. |

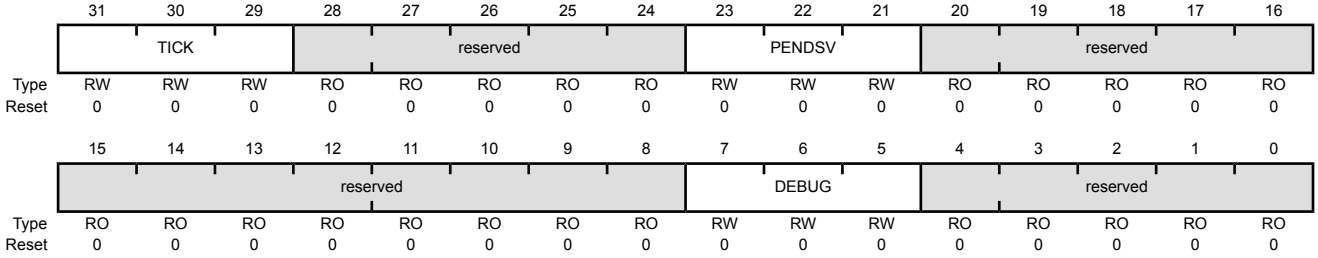
## Register 63: System Handler Priority 3 (SYSPRI3), offset 0xD20

**Note:** This register can only be accessed from privileged mode.

The **SYSPRI3** register configures the priority level, 0 to 7 of the SysTick exception and PendSV handlers. This register is byte-accessible.

System Handler Priority 3 (SYSPRI3)

Base 0xE000.E000
Offset 0xD20
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | TICK | RW | 0x0 | SysTick Exception Priority<br>This field configures the priority level of the SysTick exception. Configurable priority values are in the range 0-7, with lower values having higher priority. |
| 28:24 | reserved | RO | 0x0 | 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. |
| 23:21 | PENDSV | RW | 0x0 | PendSV Priority<br>This field configures the priority level of PendSV. Configurable priority values are in the range 0-7, with lower values having higher priority. |
| 20:8 | reserved | RO | 0x000 | 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. |
| 7:5 | DEBUG | RW | 0x0 | Debug Priority<br>This field configures the priority level of Debug. Configurable priority values are in the range 0-7, with lower values having higher priority. |
| 4:0 | reserved | RO | 0x0.0000 | 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. |

## Register 64: System Handler Control and State (SYSHNDCTRL), offset 0xD24

**Note:** This register can only be accessed from privileged mode.

The **SYSHNDCTRL** register enables the system handlers, and indicates the pending status of the usage fault, bus fault, memory management fault, and SVC exceptions as well as the active status of the system handlers.

If a system handler is disabled and the corresponding fault occurs, the processor treats the fault as a hard fault.

This register can be modified to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

**Caution – Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.**

**If the value of a bit in this register must be modified after enabling the system handlers, a read-modify-write procedure must be used to ensure that only the required bit is modified.**

System Handler Control and State (SYSHNDCTRL)

Base 0xE000.E000
Offset 0xD24
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | USAGE | BUS | MEM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SVC | BUSP | MEMP | USAGEP | TICK | PNDSV | reserved | MON | SVCA | reserved | | | USGA | reserved | BUSA | MEMA |
| Type | RW | RW | RW | RW | RW | RW | RO | RW | RW | RO | RO | RO | RW | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x000 | 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. |
| 18 | USAGE | RW | 0 | Usage Fault Enable<br>Value Description<br>0 Disables the usage fault exception.<br>1 Enables the usage fault exception. |
| 17 | BUS | RW | 0 | Bus Fault Enable<br>Value Description<br>0 Disables the bus fault exception.<br>1 Enables the bus fault exception. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | MEM | RW | 0 | Memory Management Fault Enable<br><br>Value Description<br>0 Disables the memory management fault exception.<br>1 Enables the memory management fault exception. |
| 15 | SVC | RW | 0 | SVC Call Pending<br><br>Value Description<br>0 An SVC call exception is not pending.<br>1 An SVC call exception is pending.<br><br>This bit can be modified to change the pending status of the SVC call exception. |
| 14 | BUSP | RW | 0 | Bus Fault Pending<br><br>Value Description<br>0 A bus fault exception is not pending.<br>1 A bus fault exception is pending.<br><br>This bit can be modified to change the pending status of the bus fault exception. |
| 13 | MEMP | RW | 0 | Memory Management Fault Pending<br><br>Value Description<br>0 A memory management fault exception is not pending.<br>1 A memory management fault exception is pending.<br><br>This bit can be modified to change the pending status of the memory management fault exception. |
| 12 | USAGEP | RW | 0 | Usage Fault Pending<br><br>Value Description<br>0 A usage fault exception is not pending.<br>1 A usage fault exception is pending.<br><br>This bit can be modified to change the pending status of the usage fault exception. |
| 11 | TICK | RW | 0 | SysTick Exception Active<br><br>Value Description<br>0 A SysTick exception is not active.<br>1 A SysTick exception is active.<br><br>This bit can be modified to change the active status of the SysTick exception, however, see the Caution above before setting this bit. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | PNDSV | RW | 0 | PendSV Exception Active<br><br>Value Description<br>0 A PendSV exception is not active.<br>1 A PendSV exception is active.<br><br>This bit can be modified to change the active status of the PendSV exception, however, see the Caution above before setting this bit. |
| 9 | reserved | RO | 0 | 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. |
| 8 | MON | RW | 0 | Debug Monitor Active<br><br>Value Description<br>0 The Debug monitor is not active.<br>1 The Debug monitor is active. |
| 7 | SVCA | RW | 0 | SVC Call Active<br><br>Value Description<br>0 SVC call is not active.<br>1 SVC call is active.<br><br>This bit can be modified to change the active status of the SVC call exception, however, see the Caution above before setting this bit. |
| 6:4 | reserved | RO | 0x0 | 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. |
| 3 | USGA | RW | 0 | Usage Fault Active<br><br>Value Description<br>0 Usage fault is not active.<br>1 Usage fault is active.<br><br>This bit can be modified to change the active status of the usage fault exception, however, see the Caution above before setting this bit. |
| 2 | reserved | RO | 0 | 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. |
| 1 | BUSA | RW | 0 | Bus Fault Active<br><br>Value Description<br>0 Bus fault is not active.<br>1 Bus fault is active.<br><br>This bit can be modified to change the active status of the bus fault exception, however, see the Caution above before setting this bit. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | MEMA | RW | 0 | Memory Management Fault Active |

| Value | Description |
|---|---|
| 0 | Memory management fault is not active. |
| 1 | Memory management fault is active. |

This bit can be modified to change the active status of the memory management fault exception, however, see the Caution above before setting this bit.

## Register 65: Configurable Fault Status (FAULTSTAT), offset 0xD28

**Note:** This register can only be accessed from privileged mode.

The **FAULTSTAT** register indicates the cause of a memory management fault, bus fault, or usage fault. Each of these functions is assigned to a subregister as follows:

- **Usage Fault Status (UFAULTSTAT)**, bits 31:16
- **Bus Fault Status (BFAULTSTAT)**, bits 15:8
- **Memory Management Fault Status (MFAULTSTAT)**, bits 7:0

**FAULTSTAT** is byte accessible. **FAULTSTAT** or its subregisters can be accessed as follows:

- The complete **FAULTSTAT** register, with a word access to offset 0xD28
- The **MFAULTSTAT**, with a byte access to offset 0xD28
- The **MFAULTSTAT** and **BFAULTSTAT**, with a halfword access to offset 0xD28
- The **BFAULTSTAT**, with a byte access to offset 0xD29
- The **UFAULTSTAT**, with a halfword access to offset 0xD2A

Bits are cleared by writing a 1 to them.

In a fault handler, the true faulting address can be determined by:

1. Read and save the **Memory Management Fault Address (MMADDR)** or **Bus Fault Address (FAULTADDR)** value.
2. Read the MMARV bit in **MFAULTSTAT**, or the BFARV bit in **BFAULTSTAT** to determine if the **MMADDR** or **FAULTADDR** contents are valid.

Software must follow this sequence because another higher priority exception might change the **MMADDR** or **FAULTADDR** value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the **MMADDR** or **FAULTADDR** value.

Configurable Fault Status (FAULTSTAT)
Base 0xE000.E000
Offset 0xD28
Type RW1C, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | DIV0 | UNALIGN | reserved | | | | NOCP | INVPC | INVSTAT | UNDEF |
| RO | RO | RO | RO | RO | RO | RW1C | RW1C | RO | RO | RO | RO | RW1C | RW1C | RW1C | RW1C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BFARV | reserved | BLSPERR | BSTKE | BUSTKE | IMPRE | PRECISE | IBUS | MMARV | reserved | MLSPERR | MSTKE | MUSTKE | reserved | DERR | IERR |
| RW1C | RO | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RO | RW1C | RW1C | RW1C | RO | RW1C | RW1C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25 | DIV0 | RW1C | 0 | Divide-by-Zero Usage Fault<br><br>Value Description<br>0 No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled.<br>1 The processor has executed an SDIV or UDIV instruction with a divisor of 0.<br><br>When this bit is set, the **PC** value stacked for the exception return points to the instruction that performed the divide by zero.<br>Trapping on divide-by-zero is enabled by setting the DIV0 bit in the **Configuration and Control (CFGCTRL)** register (see page 186).<br>This bit is cleared by writing a 1 to it. |
| 24 | UNALIGN | RW1C | 0 | Unaligned Access Usage Fault<br><br>Value Description<br>0 No unaligned access fault has occurred, or unaligned access trapping is not enabled.<br>1 The processor has made an unaligned memory access.<br><br>Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of the configuration of this bit.<br>Trapping on unaligned access is enabled by setting the UNALIGNED bit in the **CFGCTRL** register (see page 186).<br>This bit is cleared by writing a 1 to it. |
| 23:20 | reserved | RO | 0x00 | 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. |
| 19 | NOCP | RW1C | 0 | No Coprocessor Usage Fault<br><br>Value Description<br>0 A usage fault has not been caused by attempting to access a coprocessor.<br>1 The processor has attempted to access a coprocessor.<br><br>This bit is cleared by writing a 1 to it. |
| 18 | INVPC | RW1C | 0 | Invalid PC Load Usage Fault<br><br>Value Description<br>0 A usage fault has not been caused by attempting to load an invalid **PC** value.<br>1 The processor has attempted an illegal load of EXC_RETURN to the **PC** as a result of an invalid context or an invalid EXC_RETURN value.<br><br>When this bit is set, the **PC** value stacked for the exception return points to the instruction that tried to perform the illegal load of the **PC**.<br>This bit is cleared by writing a 1 to it. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | INVSTAT | RW1C | 0 | Invalid State Usage Fault<br><br>Value Description<br>0 A usage fault has not been caused by an invalid state.<br>1 The processor has attempted to execute an instruction that makes illegal use of the **EPSR** register.<br><br>When this bit is set, the **PC** value stacked for the exception return points to the instruction that attempted the illegal use of the **Execution Program Status Register (EPSR)** register.<br>This bit is not set if an undefined instruction uses the **EPSR** register.<br>This bit is cleared by writing a 1 to it. |
| 16 | UNDEF | RW1C | 0 | Undefined Instruction Usage Fault<br><br>Value Description<br>0 A usage fault has not been caused by an undefined instruction.<br>1 The processor has attempted to execute an undefined instruction.<br><br>When this bit is set, the **PC** value stacked for the exception return points to the undefined instruction.<br>An undefined instruction is an instruction that the processor cannot decode.<br>This bit is cleared by writing a 1 to it. |
| 15 | BFARV | RW1C | 0 | Bus Fault Address Register Valid<br><br>Value Description<br>0 The value in the **Bus Fault Address (FAULTADDR)** register is not a valid fault address.<br>1 The **FAULTADDR** register is holding a valid fault address.<br><br>This bit is set after a bus fault, where the address is known. Other faults can clear this bit, such as a memory management fault occurring later.<br>If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active bus fault handler whose **FAULTADDR** register value has been overwritten.<br>This bit is cleared by writing a 1 to it. |
| 14 | reserved | RO | 0 | 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. |
| 13 | BLSPERR | RW1C | 0 | Bus Fault on Floating-Point Lazy State Preservation<br><br>Value Description<br>0 No bus fault has occurred during floating-point lazy state preservation.<br>1 A bus fault has occurred during floating-point lazy state preservation.<br><br>This bit is cleared by writing a 1 to it. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | BSTKE | RW1C | 0 | Stack Bus Fault<br><br>Value Description<br>0 No bus fault has occurred on stacking for exception entry.<br>1 Stacking for an exception entry has caused one or more bus faults.<br><br>When this bit is set, the **SP** is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the **FAULTADDR** register.<br>This bit is cleared by writing a 1 to it. |
| 11 | BUSTKE | RW1C | 0 | Unstack Bus Fault<br><br>Value Description<br>0 No bus fault has occurred on unstacking for a return from exception.<br>1 Unstacking for a return from exception has caused one or more bus faults.<br><br>This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The **SP** is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the **FAULTADDR** register.<br>This bit is cleared by writing a 1 to it. |
| 10 | IMPRE | RW1C | 0 | Imprecise Data Bus Error<br><br>Value Description<br>0 An imprecise data bus error has not occurred.<br>1 A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error.<br><br>When this bit is set, a fault address is not written to the **FAULTADDR** register.<br>This fault is asynchronous. Therefore, if the fault is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher-priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects that both the IMPRE bit is set and one of the precise fault status bits is set.<br>This bit is cleared by writing a 1 to it. |
| 9 | PRECISE | RW1C | 0 | Precise Data Bus Error<br><br>Value Description<br>0 A precise data bus error has not occurred.<br>1 A data bus error has occurred, and the **PC** value stacked for the exception return points to the instruction that caused the fault.<br><br>When this bit is set, the fault address is written to the **FAULTADDR** register.<br>This bit is cleared by writing a 1 to it. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | IBUS | RW1C | 0 | Instruction Bus Error<br><br>Value Description<br>0 An instruction bus error has not occurred.<br>1 An instruction bus error has occurred.<br><br>The processor detects the instruction bus error on prefetching an instruction, but sets this bit only if it attempts to issue the faulting instruction.<br>When this bit is set, a fault address is not written to the **FAULTADDR** register.<br>This bit is cleared by writing a 1 to it. |
| 7 | MMARV | RW1C | 0 | Memory Management Fault Address Register Valid<br><br>Value Description<br>0 The value in the **Memory Management Fault Address (MMADDR)** register is not a valid fault address.<br>1 The **MMADDR** register is holding a valid fault address.<br><br>If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active memory management fault handler whose **MMADDR** register value has been overwritten.<br>This bit is cleared by writing a 1 to it. |
| 6 | reserved | RO | 0 | 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 | MLSPERR | RW1C | 0 | Memory Management Fault on Floating-Point Lazy State Preservation<br><br>Value Description<br>0 No memory management fault has occurred during floating-point lazy state preservation.<br>1 No memory management fault has occurred during floating-point lazy state preservation.<br><br>This bit is cleared by writing a 1 to it. |
| 4 | MSTKE | RW1C | 0 | Stack Access Violation<br><br>Value Description<br>0 No memory management fault has occurred on stacking for exception entry.<br>1 Stacking for an exception entry has caused one or more access violations.<br><br>When this bit is set, the **SP** is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the **MMADDR** register.<br>This bit is cleared by writing a 1 to it. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MUSTKE | RW1C | 0 | Unstack Access Violation<br><br>Value Description<br>0 No memory management fault has occurred on unstacking for a return from exception.<br>1 Unstacking for a return from exception has caused one or more access violations.<br><br>This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The **SP** is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the **MMADDR** register.<br>This bit is cleared by writing a 1 to it. |
| 2 | reserved | RO | 0 | 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. |
| 1 | DERR | RW1C | 0 | Data Access Violation<br><br>Value Description<br>0 A data access violation has not occurred.<br>1 The processor attempted a load or store at a location that does not permit the operation.<br><br>When this bit is set, the **PC** value stacked for the exception return points to the faulting instruction and the address of the attempted access is written to the **MMADDR** register.<br>This bit is cleared by writing a 1 to it. |
| 0 | IERR | RW1C | 0 | Instruction Access Violation<br><br>Value Description<br>0 An instruction access violation has not occurred.<br>1 The processor attempted an instruction fetch from a location that does not permit execution.<br><br>This fault occurs on any access to an XN region, even when the MPU is disabled or not present.<br>When this bit is set, the **PC** value stacked for the exception return points to the faulting instruction and the address of the attempted access is not written to the **MMADDR** register.<br>This bit is cleared by writing a 1 to it. |

## Register 66: Hard Fault Status (HFAULTSTAT), offset 0xD2C

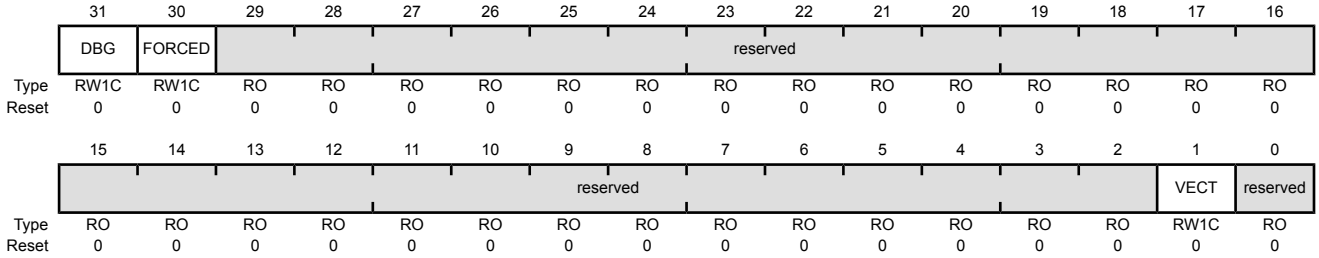
**Note:** This register can only be accessed from privileged mode.

The **HFAULTSTAT** register gives information about events that activate the hard fault handler.

Bits are cleared by writing a 1 to them.

Hard Fault Status (HFAULTSTAT)

Base 0xE000.E000
Offset 0xD2C
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | DBG | RW1C | 0 | Debug Event<br>This bit is reserved for Debug use. This bit must be written as a 0, otherwise behavior is unpredictable. |
| 30 | FORCED | RW1C | 0 | Forced Hard Fault<br>Value Description<br>0 No forced hard fault has occurred.<br>1 A forced hard fault has been generated by escalation of a fault with configurable priority that cannot be handled, either because of priority or because it is disabled.<br>When this bit is set, the hard fault handler must read the other fault status registers to find the cause of the fault.<br>This bit is cleared by writing a 1 to it. |
| 29:2 | reserved | RO | 0x00 | 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. |
| 1 | VECT | RW1C | 0 | Vector Table Read Fault<br>Value Description<br>0 No bus fault has occurred on a vector table read.<br>1 A bus fault occurred on a vector table read.<br>This error is always handled by the hard fault handler.<br>When this bit is set, the **PC** value stacked for the exception return points to the instruction that was preempted by the exception.<br>This bit is cleared by writing a 1 to it. |
| 0 | reserved | RO | 0 | 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. |

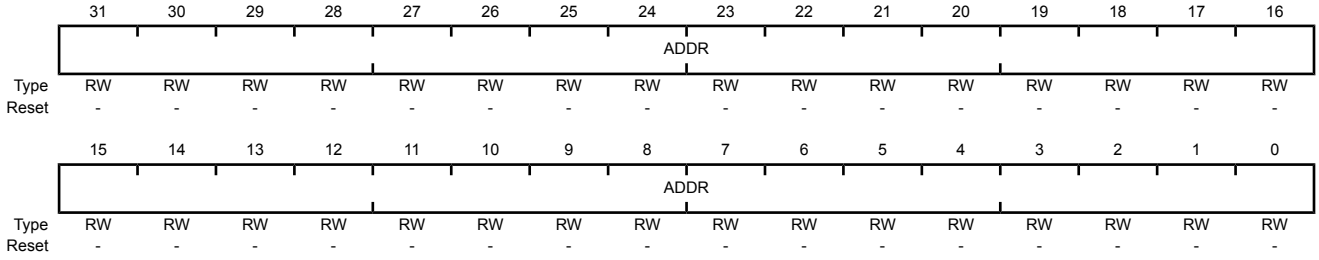
## Register 67: Memory Management Fault Address (MMADDR), offset 0xD34

**Note:** This register can only be accessed from privileged mode.

The **MMADDR** register contains the address of the location that generated a memory management fault. When an unaligned access faults, the address in the **MMADDR** register is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size. Bits in the **Memory Management Fault Status (MFAULTSTAT)** register indicate the cause of the fault and whether the value in the **MMADDR** register is valid (see page 195).

Memory Management Fault Address (MMADDR)

Base 0xE000.E000
Offset 0xD34
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | RW | - | Fault Address<br>When the MMARV bit of **MFAULTSTAT** is set, this field holds the address of the location that generated the memory management fault. |

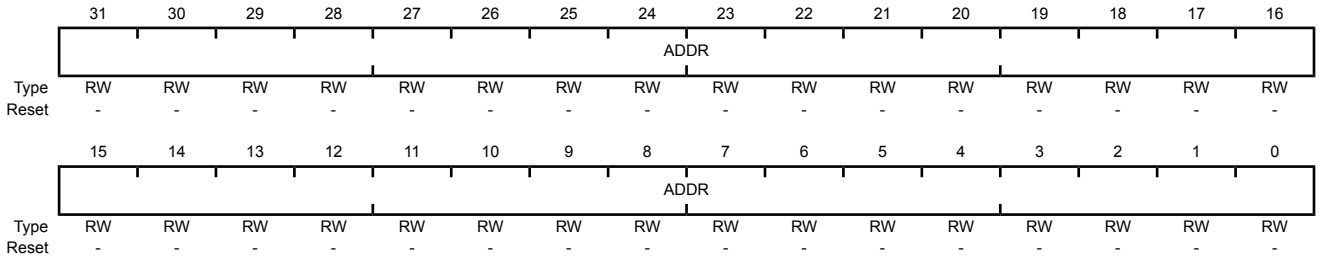
### Register 68: Bus Fault Address (FAULTADDR), offset 0xD38

**Note:** This register can only be accessed from privileged mode.

The **FAULTADDR** register contains the address of the location that generated a bus fault. When an unaligned access faults, the address in the **FAULTADDR** register is the one requested by the instruction, even if it is not the address of the fault. Bits in the **Bus Fault Status (BFAULTSTAT)** register indicate the cause of the fault and whether the value in the **FAULTADDR** register is valid (see page 195).

Bus Fault Address (FAULTADDR)

Base 0xE000.E000
Offset 0xD38
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | RW | - | Fault Address<br>When the FAULTADDRV bit of **BFAULTSTAT** is set, this field holds the address of the location that generated the bus fault. |

## 3.6 Memory Protection Unit (MPU) Register Descriptions

This section lists and describes the Memory Protection Unit (MPU) registers, in numerical order by address offset.

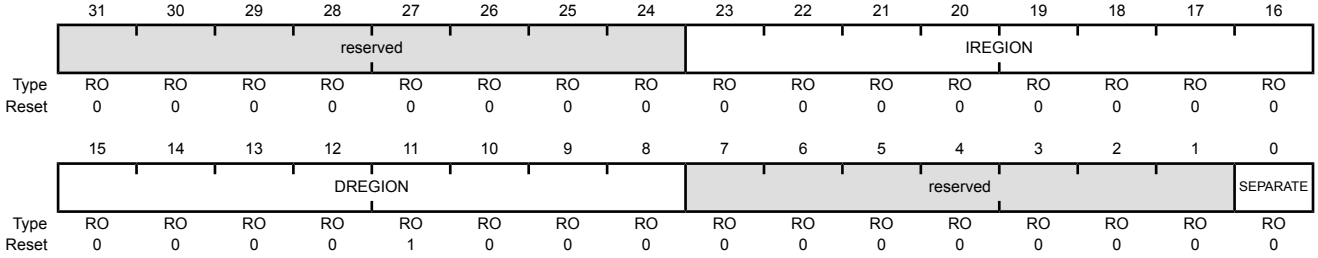
The MPU registers can only be accessed from privileged mode.

## Register 69: MPU Type (MPUTYPE), offset 0xD90

**Note:** This register can only be accessed from privileged mode.

The **MPUTYPE** register indicates whether the MPU is present, and if so, how many regions it supports.

MPU Type (MPUTYPE)
Base 0xE000.E000
Offset 0xD90
Type RO, reset 0x0000.0800




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x00 | 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. |
| 23:16 | IREGION | RO | 0x00 | Number of I Regions<br>This field indicates the number of supported MPU instruction regions. This field always contains 0x00. The MPU memory map is unified and is described by the `DREGION` field. |
| 15:8 | DREGION | RO | 0x08 | Number of D Regions<br>Value Description<br>0x08 Indicates there are eight supported MPU data regions. |
| 7:1 | reserved | RO | 0x00 | 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. |
| 0 | SEPARATE | RO | 0 | Separate or Unified MPU<br>Value Description<br>0 Indicates the MPU is unified. |

## Register 70: MPU Control (MPUCTRL), offset 0xD94

**Note:** This register can only be accessed from privileged mode.

The **MPUCTRL** register enables the MPU, enables the default memory map background region, and enables use of the MPU when in the hard fault, Non-maskable Interrupt (NMI), and **Fault Mask Register (FAULTMASK)** escalated handlers.

When the ENABLE and PRIVDEFEN bits are both set:

- For privileged accesses, the default memory map is as described in "Memory Model" on page 113. Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
- Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

Execute Never (XN) and Strongly Ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the ENABLE bit is set, at least one region of the memory map must be enabled for the system to function unless the PRIVDEFEN bit is set. If the PRIVDEFEN bit is set and no regions are enabled, then only privileged software can operate.

When the ENABLE bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see Table 2-5 on page 117 for more information). The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether PRIVDEFEN is set.

Unless HFNMIENA is set, the MPU is not enabled when the processor is executing the handler for an exception with priority –1 or –2. These priorities are only possible when handling a hard fault or NMI exception or when **FAULTMASK** is enabled. Setting the HFNMIENA bit enables the MPU when operating with these two priorities.

MPU Control (MPUCTRL)

Base 0xE000.E000
Offset 0xD94
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| Type RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | PRIVDEFEN | HFNMIENA | ENABLE |
| Type RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | PRIVDEFEN | RW | 0 | MPU Default Region<br>This bit enables privileged software access to the default memory map.<br><br>Value Description<br>0 If the MPU is enabled, this bit disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault.<br>1 If the MPU is enabled, this bit enables use of the default memory map as a background region for privileged software accesses.<br><br>When this bit is set, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map.<br>If the MPU is disabled, the processor ignores this bit. |
| 1 | HFNMIENA | RW | 0 | MPU Enabled During Faults<br>This bit controls the operation of the MPU during hard fault, NMI, and **FAULTMASK** handlers.<br><br>Value Description<br>0 The MPU is disabled during hard fault, NMI, and **FAULTMASK** handlers, regardless of the value of the ENABLE bit.<br>1 The MPU is enabled during hard fault, NMI, and **FAULTMASK** handlers.<br><br>When the MPU is disabled and this bit is set, the resulting behavior is unpredictable. |
| 0 | ENABLE | RW | 0 | MPU Enable<br><br>Value Description<br>0 The MPU is disabled.<br>1 The MPU is enabled.<br><br>When the MPU is disabled and the HFNMIENA bit is set, the resulting behavior is unpredictable. |

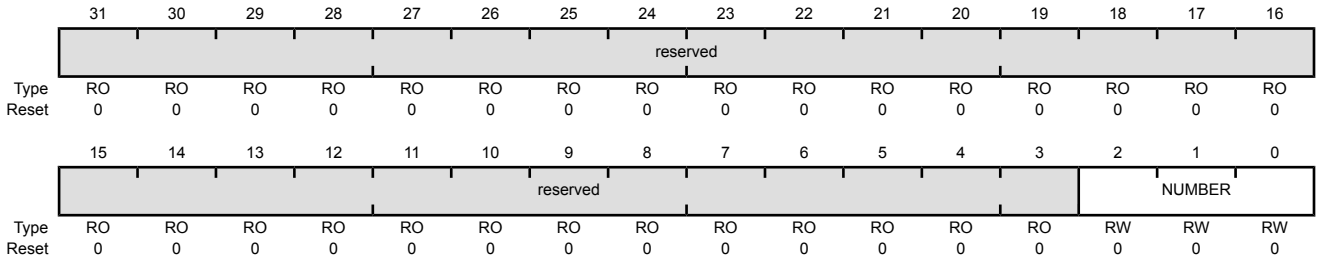
## Register 71: MPU Region Number (MPUNUMBER), offset 0xD98

**Note:** This register can only be accessed from privileged mode.

The **MPUNUMBER** register selects which memory region is referenced by the **MPU Region Base Address (MPUBASE)** and **MPU Region Attribute and Size (MPUATTR)** registers. Normally, the required region number should be written to this register before accessing the **MPUBASE** or the **MPUATTR** register. However, the region number can be changed by writing to the **MPUBASE** register with the VALID bit set (see page 208). This write updates the value of the REGION field.

MPU Region Number (MPUNUMBER)

Base 0xE000.E000
Offset 0xD98
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | 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. |
| 2:0 | NUMBER | RW | 0x0 | MPU Region to Access<br>This field indicates the MPU region referenced by the **MPUBASE** and **MPUATTR** registers. The MPU supports eight memory regions. |

## Register 72: MPU Region Base Address (MPUBASE), offset 0xD9C

## Register 73: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4

## Register 74: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC

## Register 75: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4

**Note:** This register can only be accessed from privileged mode.

The **MPUBASE** register defines the base address of the MPU region selected by the **MPU Region Number (MPUNUMBER)** register and can update the value of the **MPUNUMBER** register. To change the current region number and update the **MPUNUMBER** register, write the **MPUBASE** register with the VALID bit set.

The ADDR field is bits 31:*N* of the **MPUBASE** register. Bits (*N*-1):5 are reserved. The region size, as specified by the SIZE field in the **MPU Region Attribute and Size (MPUATTR)** register, defines the value of *N* where:

$N = Log_2(\text{Region size in bytes})$

If the region size is configured to 4 GB in the **MPUATTR** register, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x0000.0000.

The base address is aligned to the size of the region. For example, a 64-KB region must be aligned on a multiple of 64 KB, for example, at 0x0001.0000 or 0x0002.0000.

MPU Region Base Address (MPUBASE)

Base 0xE000.E000
Offset 0xD9C
Type RW, reset 0x0000.0000

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

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR | | | | | | | | | | | VALID | reserved | REGION | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | WO | RO | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | ADDR | RW | 0x0000.000 | Base Address Mask<br>Bits 31:*N* in this field contain the region base address. The value of *N* depends on the region size, as shown above. The remaining bits (*N*-1):5 are reserved.<br>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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | VALID | WO | 0 | Region Number Valid<br><br>Value Description<br>0 The **MPUNUMBER** register is not changed and the processor updates the base address for the region specified in the **MPUNUMBER** register and ignores the value of the REGION field.<br>1 The **MPUNUMBER** register is updated with the value of the REGION field and the base address is updated for the region specified in the REGION field.<br><br>This bit is always read as 0. |
| 3 | reserved | RO | 0 | 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. |
| 2:0 | REGION | RW | 0x0 | Region Number<br><br>On a write, contains the value to be written to the **MPUNUMBER** register. On a read, returns the current region number in the **MPUNUMBER** register. |

## Register 76: MPU Region Attribute and Size (MPUATTR), offset 0xDA0

## Register 77: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8

## Register 78: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0

## Register 79: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8

**Note:** This register can only be accessed from privileged mode.

The **MPUATTR** register defines the region size and memory attributes of the MPU region specified by the **MPU Region Number (MPUNUMBER)** register and enables that region and any subregions.

The **MPUATTR** register is accessible using word or halfword accesses with the most-significant halfword holding the region attributes and the least-significant halfword holds the region size and the region and subregion enable bits.

The MPU access permission attribute bits, XN, AP, TEX, S, C, and B, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register as follows:

(Region size in bytes) = $2^{(SIZE+1)}$

The smallest permitted region size is 32 bytes, corresponding to a SIZE value of 4. Table 3-10 on page 210 gives example SIZE values with the corresponding region size and value of N in the **MPU Region Base Address (MPUBASE)** register.

**Table 3-10. Example SIZE Field Values**

| SIZE Encoding | Region Size | Value of N[a] | Note |
|---|---|---|---|
| 00100b (0x4) | 32 B | 5 | Minimum permitted size |
| 01001b (0x9) | 1 KB | 10 | - |
| 10011b (0x13) | 1 MB | 20 | - |
| 11101b (0x1D) | 1 GB | 30 | - |
| 11111b (0x1F) | 4 GB | No valid ADDR field in **MPUBASE**; the region occupies the complete memory map. | Maximum possible size |

a. Refers to the N parameter in the **MPUBASE** register (see page 208).

MPU Region Attribute and Size (MPUATTR)

Base 0xE000.E000
Offset 0xDA0
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | XN | reserved | AP | | | reserved | | TEX | | | S | C | B |
| Type | RO | RO | RO | RW | RO | RW | RW | RW | RO | RO | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRD | | | | | | | | reserved | | SIZE | | | | | ENABLE |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RO | RO | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0x00 | 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. |
| 28 | XN | RW | 0 | Instruction Access Disable<br>Value Description<br>0 Instruction fetches are enabled.<br>1 Instruction fetches are disabled. |
| 27 | reserved | RO | 0 | 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. |
| 26:24 | AP | RW | 0 | Access Privilege<br>For information on using this bit field, see Table 3-5 on page 152. |
| 23:22 | reserved | RO | 0x0 | 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. |
| 21:19 | TEX | RW | 0x0 | Type Extension Mask<br>For information on using this bit field, see Table 3-3 on page 151. |
| 18 | S | RW | 0 | Shareable<br>For information on using this bit, see Table 3-3 on page 151. |
| 17 | C | RW | 0 | Cacheable<br>For information on using this bit, see Table 3-3 on page 151. |
| 16 | B | RW | 0 | Bufferable<br>For information on using this bit, see Table 3-3 on page 151. |
| 15:8 | SRD | RW | 0x00 | Subregion Disable Bits<br>Value Description<br>0 The corresponding subregion is enabled.<br>1 The corresponding subregion is disabled.<br>Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, configure the SRD field as 0x00. See the section called “Subregions” on page 151 for more information. |
| 7:6 | reserved | RO | 0x0 | 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. |
| 5:1 | SIZE | RW | 0x0 | Region Size Mask<br>The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register. Refer to Table 3-10 on page 210 for more information. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ENABLE | RW | 0 | Region Enable |

| Value | Description |
|---|---|
| 0 | The region is disabled. |
| 1 | The region is enabled. |

## 3.7 Floating-Point Unit (FPU) Register Descriptions

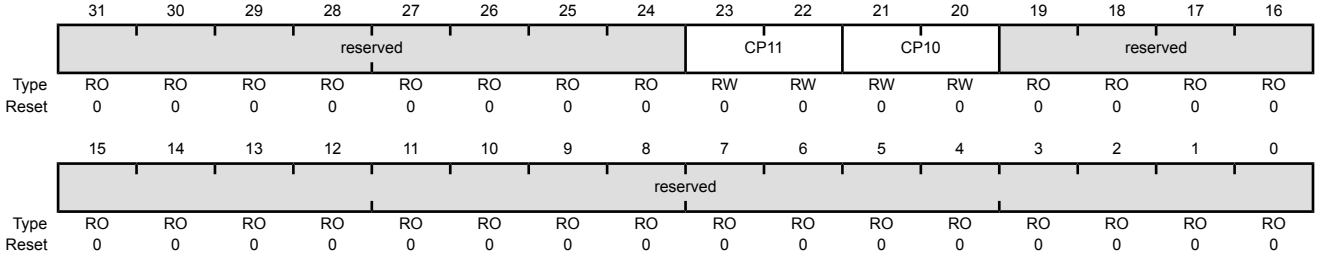
This section lists and describes the Floating-Point Unit (FPU) registers, in numerical order by address offset.

## Register 80: Coprocessor Access Control (CPAC), offset 0xD88

The **CPAC** register specifies the access privileges for coprocessors.

Coprocessor Access Control (CPAC)

Base 0xE000.E000
Offset 0xD88
Type RW, reset 0x0000.0000



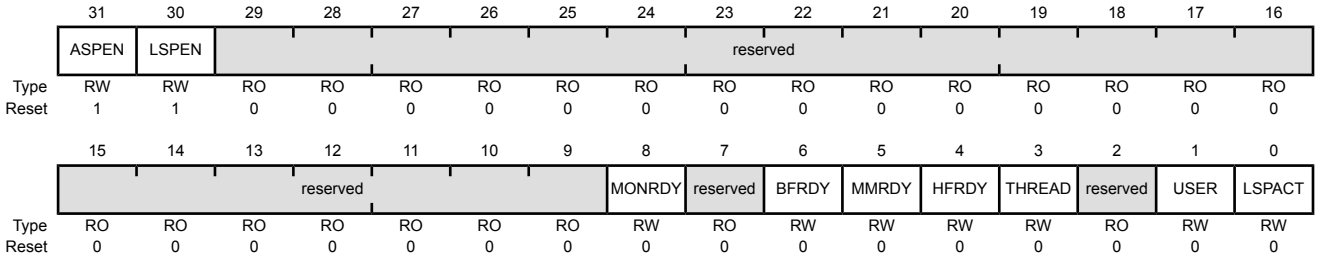

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 23:22 | CP11 | RW | 0x00 | CP11 Coprocessor Access Privilege<br><br>Value Description<br>0x0 Access Denied<br>Any attempted access generates a NOCP Usage Fault.<br>0x1 Privileged Access Only<br>An unprivileged access generates a NOCP fault.<br>0x2 Reserved<br>The result of any access is unpredictable.<br>0x3 Full Access |
| 21:20 | CP10 | RW | 0x00 | CP10 Coprocessor Access Privilege<br><br>Value Description<br>0x0 Access Denied<br>Any attempted access generates a NOCP Usage Fault.<br>0x1 Privileged Access Only<br>An unprivileged access generates a NOCP fault.<br>0x2 Reserved<br>The result of any access is unpredictable.<br>0x3 Full Access |
| 19:0 | reserved | RO | 0x00 | 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. |

## Register 81: Floating-Point Context Control (FPCC), offset 0xF34

The **FPCC** register sets or returns FPU control data.

Floating-Point Context Control (FPCC)

Base 0xE000.E000
Offset 0xF34
Type RW, reset 0xC000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ASPEN | RW | 1 | Automatic State Preservation Enable<br>When set, enables the use of the FRACTV bit in the **CONTROL** register on execution of a floating-point instruction. This results in automatic hardware state preservation and restoration, for floating-point context, on exception entry and exit.<br>**Important:** Two bits control when FPCA can be enabled: the ASPEN bit in the **Floating-Point Context Control (FPCC)** register and the DISFPCA bit in the **Auxiliary Control (ACTLR)** register. |
| 30 | LSPEN | RW | 1 | Lazy State Preservation Enable<br>When set, enables automatic lazy state preservation for floating-point context. |
| 29:9 | reserved | RO | 0x00 | 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. |
| 8 | MONRDY | RW | 0 | Monitor Ready<br>When set, DebugMonitor is enabled and priority permits setting MON_PEND when the floating-point stack frame was allocated. |
| 7 | reserved | RO | 0 | 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. |
| 6 | BFRDY | RW | 0 | Bus Fault Ready<br>When set, BusFault is enabled and priority permitted setting the BusFault handler to the pending state when the floating-point stack frame was allocated. |
| 5 | MMRDY | RW | 0 | Memory Management Fault Ready<br>When set, MemManage is enabled and priority permitted setting the MemManage handler to the pending state when the floating-point stack frame was allocated. |

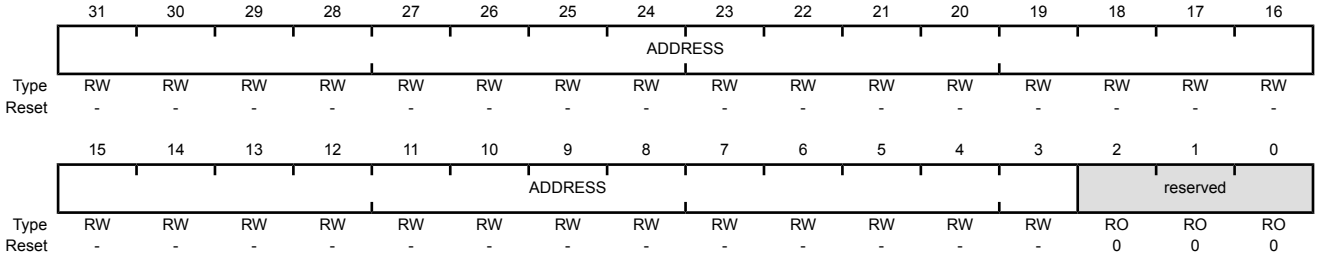
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | HFRDY | RW | 0 | Hard Fault Ready<br>When set, priority permitted setting the HardFault handler to the pending state when the floating-point stack frame was allocated. |
| 3 | THREAD | RW | 0 | Thread Mode<br>When set, mode was Thread Mode when the floating-point stack frame was allocated. |
| 2 | reserved | RO | 0 | 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. |
| 1 | USER | RW | 0 | User Privilege Level<br>When set, privilege level was user when the floating-point stack frame was allocated. |
| 0 | LSPACT | RW | 0 | Lazy State Preservation Active<br>When set, Lazy State preservation is active. Floating-point stack frame has been allocated but saving state to it has been deferred. |

## Register 82: Floating-Point Context Address (FPCA), offset 0xF38

The **FPCA** register holds the location of the unpopulated floating-point register space allocated on an exception stack frame.

Floating-Point Context Address (FPCA)

Base 0xE000.E000
Offset 0xF38
Type RW, reset -



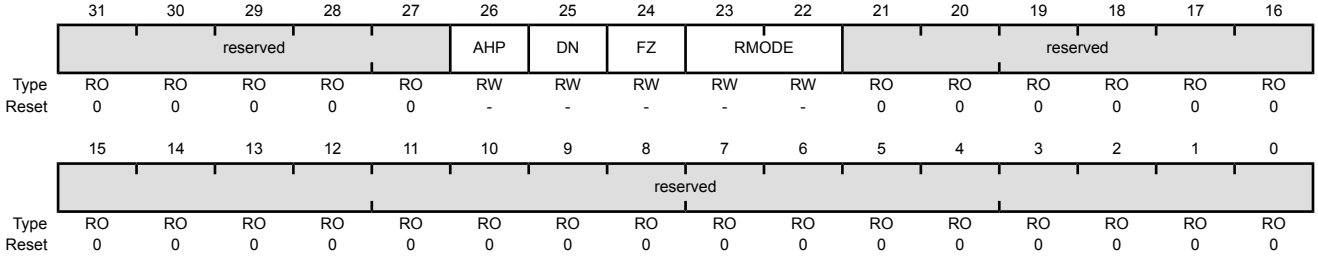

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | ADDRESS | RW | - | Address<br>The location of the unpopulated floating-point register space allocated on an exception stack frame. |
| 2:0 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

## Register 83: Floating-Point Default Status Control (FPDSC), offset 0xF3C

The **FPDSC** register holds the default values for the **Floating-Point Status Control (FPSC)** register.

Floating-Point Default Status Control (FPDSC)

Base 0xE000.E000
Offset 0xF3C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:27 | reserved | RO | 0x00 | 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. |
| 26 | AHP | RW | - | AHP Bit Default<br>This bit holds the default value for the AHP bit in the **FPSC** register. |
| 25 | DN | RW | - | DN Bit Default<br>This bit holds the default value for the DN bit in the **FPSC** register. |
| 24 | FZ | RW | - | FZ Bit Default<br>This bit holds the default value for the FZ bit in the **FPSC** register. |
| 23:22 | RMODE | RW | - | RMODE Bit Default<br>This bit holds the default value for the RMODE bit field in the **FPSC** register.<br><br>Value Description<br>0x0 Round to Nearest (RN) mode<br>0x1 Round towards Plus Infinity (RP) mode<br>0x2 Round towards Minus Infinity (RM) mode<br>0x3 Round towards Zero (RZ) mode |
| 21:0 | reserved | RO | 0x00 | 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. |

# 4 JTAG Interface

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and 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 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 JTAG port is comprised of four pins: TCK, TMS, TDI, and TDO. Data is transmitted serially into the controller on TDI and out of the controller on TDO. The interpretation of this data is dependent on the current state of the TAP controller. For detailed information on the operation of the JTAG port and TAP controller, please refer to the *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*.

The TM4C129XKCZAD JTAG controller works with the ARM JTAG controller built into the Cortex-M4F core by multiplexing the TDO outputs from both JTAG controllers. ARM JTAG instructions select the ARM TDO output while JTAG instructions select the TDO output. The multiplexer is controlled by the JTAG controller, which has comprehensive programming for the ARM, Tiva™ C Series microcontroller, and unimplemented JTAG instructions.

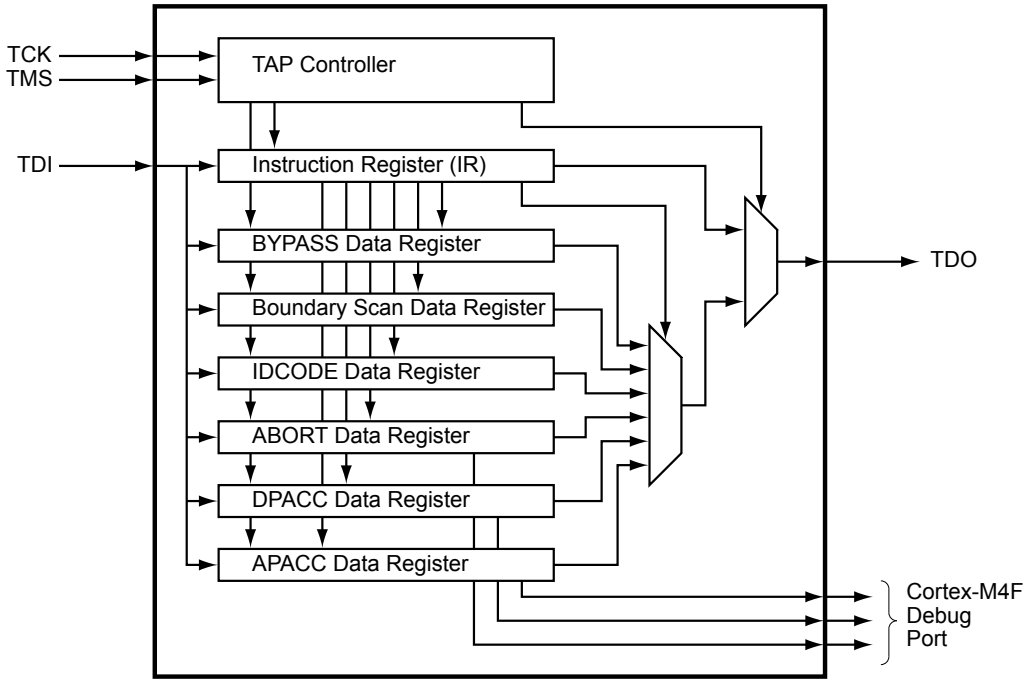
The TM4C129XKCZAD JTAG module has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, and EXTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trace (DWT) unit for implementing watchpoints, trigger resources, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
  - Embedded Trace Macrocell (ETM) for instruction trace capture
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

See the *ARM® Debug Interface V5 Architecture Specification* for more information on the ARM JTAG controller.

## 4.1 Block Diagram

**Figure 4-1. JTAG Module Block Diagram**




## 4.2 Signal Description

The following table lists the external signals of the JTAG/SWD controller and describes the function of each. The JTAG/SWD controller signals are alternate functions for some GPIO signals, however note that the reset state of the pins is for the JTAG/SWD function. The JTAG/SWD controller signals are under commit protection and require a special process to be configured as GPIOs, see “Commit Control” on page 784. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the JTAG/SWD controller signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) is set to choose the JTAG/SWD function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the JTAG/SWD controller signals to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 4-1. JTAG_SWD_SWO Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| SWCLK | B15 | PC0 (1) | I | TTL | JTAG/SWD CLK. |
| SWDIO | C15 | PC1 (1) | I/O | TTL | JTAG TMS and SWDIO. |
| SWO | C14 | PC3 (1) | O | TTL | JTAG TDO and SWO. |
| TCK | B15 | PC0 (1) | I | TTL | JTAG/SWD CLK. |
| TDI | D14 | PC2 (1) | I | TTL | JTAG TDI. |
| TDO | C14 | PC3 (1) | O | TTL | JTAG TDO and SWO. |
| TMS | C15 | PC1 (1) | I | TTL | JTAG TMS and SWDIO. |

## 4.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 4-1 on page 219. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the TCK and TMS inputs. The current state of the TAP controller depends on the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI towards TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

The serial shift chains with parallel load registers are comprised of a single Instruction Register (IR) chain and multiple Data Register (DR) chains. The current instruction loaded in the parallel load register determines which DR chain is captured, shifted, or updated during the sequencing of the TAP controller.

Some instructions, like EXTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 4-3 on page 227 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 2101 for JTAG timing diagrams.

Depending on the reset source, the effect on the JTAG module varies. The following reset sources reset the entire JTAG Module:

- Externally generated Power-On Reset

The following reset sources reset only the JTAG pin configuration:

- $\overline{\text{RST}}$ pin Power-On Reset
- Brown-Out Power-On Reset
- Watchdog Power-On Reset
- HIB Module Power-On Reset
- $\overline{\text{RST}}$ pin System Reset
- Brown-Out System Reset
- Software System Reset Request (using the SYSRESREQ bit in the **APINT** register)
- Software Peripheral Reset
- Watchdog System Reset
- HIB Module System Reset

### 4.3.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. These pins and their associated state after a power-on reset or reset caused by the $\overline{\text{RST}}$ input are given in Table 4-2. Detailed information on each pin follows.

**Note:** The following pins are configured as JTAG port pins out of reset. Refer to "General-Purpose Input/Outputs (GPIOs)" on page 772 for information on how to reprogram the configuration of these pins.

**Table 4-2. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion**

| Pin Name | Data Direction | Internal Pull-Up | Internal Pull-Down | Drive Strength | Drive Value |
|---|---|---|---|---|---|
| TCK | Input | Enabled | Disabled | N/A | N/A |
| TMS | Input | Enabled | Disabled | N/A | N/A |
| TDI | Input | Enabled | Disabled | N/A | N/A |
| TDO | Output | Enabled | Disabled | 2-mA driver | High-Z |

#### 4.3.1.1 Test Clock Input (TCK)

The TCK pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks and to ensure that multiple JTAG TAP controllers that are daisy-chained together can synchronously communicate serial test data between components. During normal operation, TCK is driven by a free-running clock with a nominal 50% duty cycle. When necessary, TCK can be stopped at 0 or 1 for extended periods of time. While TCK is stopped at 0 or 1, the state of the TAP controller does not change and data in the JTAG Instruction and Data Registers is not lost.

By default, the internal pull-up resistor on the TCK pin is enabled after reset, assuring that no clocking occurs if the pin is not driven from an external source. The internal pull-up and pull-down resistors can be turned off to save internal power as long as the TCK pin is constantly being driven by an external source (see page 809 and page 811).

#### 4.3.1.2 Test Mode Select (TMS)

The TMS pin selects the next state of the JTAG TAP controller. TMS is sampled on the rising edge of TCK. Depending on the current TAP state and the sampled value of TMS, the next state may be entered. Because the TMS pin is sampled on the rising edge of TCK, the *IEEE Standard 1149.1* expects the value on TMS to change on the falling edge of TCK.

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 4-2 on page 223.

By default, the internal pull-up resistor on the TMS pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC1/TMS; otherwise JTAG communication could be lost (see page 809).

#### 4.3.1.3 Test Data Input (TDI)

The TDI pin provides a stream of serial information to the IR chain and the DR chains. TDI is sampled on the rising edge of TCK and, depending on the current TAP state and the current instruction, may present this data to the proper shift register chain. Because the TDI pin is sampled on the rising edge of TCK, the *IEEE Standard 1149.1* expects the value on TDI to change on the falling edge of TCK.

By default, the internal pull-up resistor on the TDI pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC2/TDI; otherwise JTAG communication could be lost (see page 809).

#### 4.3.1.4 Test Data Output (TDO)

The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an inactive drive state when not actively shifting out data. Because TDO can be connected to the TDI of another controller in a daisy-chain configuration, the *IEEE Standard 1149.1* expects the value on TDO to change on the falling edge of TCK.

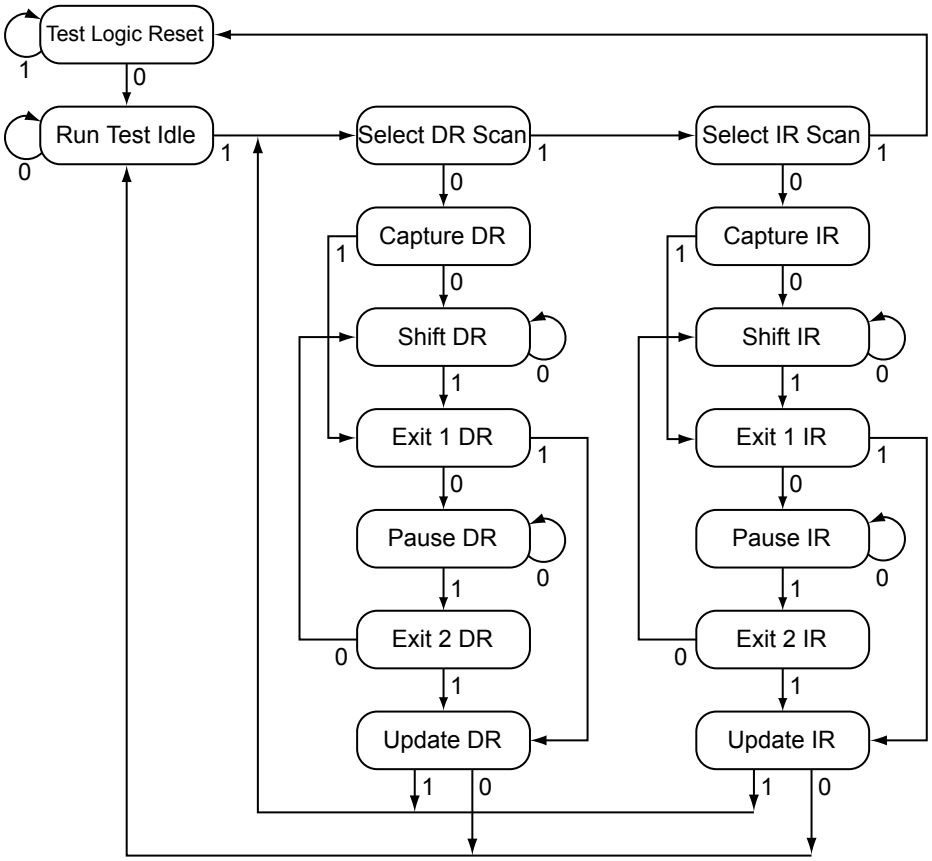
By default, the internal pull-up resistor on the TDO pin is enabled after reset, assuring that the pin remains at a constant logic level when the JTAG port is not being used. The internal pull-up and pull-down resistors can be turned off to save internal power if a High-Z output value is acceptable during certain TAP controller states (see page 809 and page 811).

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

### 4.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 4-2. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). In order to reset the JTAG module after the microcontroller has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

**Figure 4-2. Test Access Port State Machine**




### 4.3.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out on TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in “Register Descriptions” on page 226.

### 4.3.4 Operational Considerations

Certain operational parameters must be considered when using the JTAG module. Because the JTAG pins can be programmed to be GPIOs, board configuration and reset conditions on these pins must be considered. In addition, because the JTAG module has integrated ARM Serial Wire Debug, the method for switching between these two operational modes is described below.

#### 4.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or $\overline{\text{RST}}$, the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (DEN[3:0] set in the **Port C GPIO Digital Enable (GPIODEN)** register), enabling the pull-up resistors (PUE[3:0] set in the **Port C GPIO Pull-Up Select (GPIOPUR)** register), disabling the pull-down resistors (PDE[3:0] cleared in the **Port C GPIO Pull-Down Select (GPIOPDR)** register) and enabling the

alternate hardware function (AFSEL[3:0] set in the **Port C GPIO Alternate Function Select (GPIOAFSEL)** register) on the JTAG/SWD pins. See page 803, page 809, page 811, and page 814.

It is possible for software to configure these pins as GPIOs after reset by clearing AFSEL[3:0] in the **Port C GPIOAFSEL** register. If the user does not require the JTAG/SWD port for debugging or board-level testing, this provides four more GPIOs for use in the design.

---

**Caution – It is possible to create a software sequence that prevents the debugger from connecting to the TM4C129XKCZAD microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger. In the case that the software routine is not implemented and the device is locked out of the part, this issue can be solved by using the TM4C129XKCZAD Flash Programmer "Unlock" feature. Please refer to LMFLASHPROGRAMMER on the TI web for more information.**

---

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 2028 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803), **GPIO Pull Up Select (GPIOPUR)** register (see page 809), **GPIO Pull-Down Select (GPIOPDR)** register (see page 811), and **GPIO Digital Enable (GPIODEN)** register (see page 814) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 816) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 817) have been set.

#### 4.3.4.2 Communication with JTAG/SWD

Because the debug clock and the system clock can be running at different frequencies, care must be taken to maintain reliable communication with the JTAG/SWD interface. In the Capture-DR state, the result of the previous transaction, if any, is returned, together with a 3-bit ACK response. Software should check the ACK response to see if the previous operation has completed before initiating a new transaction. Alternatively, if the system clock is at least 8 times faster than the debug clock (TCK or SWCLK), the previous operation has enough time to complete and the ACK bits do not have to be checked.

#### 4.3.4.3 Recovering a "Locked" Microcontroller

**Note:** Performing the sequence below restores the non-volatile registers discussed in “Non-Volatile Register Programming-- Flash Memory Resident Registers” on page 645 to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the non-volatile registers being restored.

In addition, the EEPROM is erased and its wear-leveling counters are returned to factory default values when performing the sequence below.

If software configures any of the JTAG/SWD pins as GPIO and loses the ability to communicate with the debugger, there is a debug port unlock sequence that can be used to recover the microcontroller. Performing a total of ten JTAG-to-SWD and SWD-to-JTAG switch sequences while holding the microcontroller in reset mass erases the Flash memory. The debug port unlock sequence is:

1. Assert and hold the $\overline{\text{RST}}$ signal.
2. Apply power to the device.

3. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence on the section called “JTAG-to-SWD Switching” on page 225.
4. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on the section called “SWD-to-JTAG Switching” on page 226.
5. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
6. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
7. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
8. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
9. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
10. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
11. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
12. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
13. Release the $\overline{\text{RST}}$ signal.
14. Wait 400 ms.
15. Power-cycle the microcontroller.

#### 4.3.4.4 ARM Serial Wire Debug (SWD)

In order to seamlessly integrate the ARM Serial Wire Debug (SWD) functionality, a serial-wire debugger must be able to connect to the Cortex-M4F core without having to perform, or have any knowledge of, JTAG cycles. This integration is accomplished with a SWD preamble that is issued before the SWD session begins.

The switching preamble used to enable the SWD interface of the SWJ-DP module starts with the TAP controller in the Test-Logic-Reset state. From here, the preamble sequences the TAP controller through the following states: Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, and Test Logic Reset states.

Stepping through this sequence of the TAP state machine enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the *ARM® Debug Interface V5 Architecture Specification*.

Because this sequence is a valid series of JTAG operations that could be issued, the ARM JTAG TAP controller is not fully compliant to the *IEEE Standard 1149.1*. This instance is the only one where the ARM JTAG TAP controller does not meet full compliance with the specification. Due to the low probability of this sequence occurring during normal operation of the TAP controller, it should not affect normal performance of the JTAG interface.

##### *JTAG-to-SWD Switching*

To switch the operating mode of the Debug Access Port (DAP) from JTAG to SWD mode, the external debug hardware must send the switching preamble to the microcontroller. The 16-bit TMS/SWDIO command for switching to SWD mode is defined as b1110.0111.1001.1110, transmitted LSB first. This command can also be represented as 0xE79E when transmitted LSB first. The

complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that both JTAG and SWD are in their reset states.
2. Send the 16-bit JTAG-to-SWD switch command, 0xE79E, on TMS/SWDIO.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in SWD mode before sending the switch sequence, the SWD goes into the line reset state.

To verify that the Debug Access Port (DAP) has switched to the Serial Wire Debug (SWD) operating mode, perform a SWD READID operation. The ID value can be compared against the device's known ID to verify the switch.

***SWD-to-JTAG Switching***

To switch the operating mode of the Debug Access Port (DAP) from SWD to JTAG mode, the external debug hardware must send a switch command to the microcontroller. The 16-bit TMS/SWDIO command for switching to JTAG mode is defined as b1110.0111.0011.1100, transmitted LSB first. This command can also be represented as 0xE73C when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that both JTAG and SWD are in their reset states.
2. Send the 16-bit SWD-to-JTAG switch command, 0xE73C, on TMS/SWDIO.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in JTAG mode before sending the switch sequence, the JTAG goes into the Test Logic Reset state.

To verify that the Debug Access Port (DAP) has switched to the JTAG operating mode, set the JTAG Instruction Register (IR) to the IDCODE instruction and shift out the Data Register (DR). The DR value can be compared against the device's known IDCODE to verify the switch.

## 4.4 Initialization and Configuration

After a Power-On-Reset or an external reset ($\overline{\text{RST}}$), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. To return the pins to their JTAG functions, enable the four JTAG pins (PC[3:0]) for their alternate function using the **GPIOAFSEL** register. In addition to enabling the alternate functions, any other changes to the GPIO pad configurations on the four JTAG pins (PC[3:0]) should be returned to their default settings.

## 4.5 Register Descriptions

The registers in the JTAG TAP Controller or Shift Register chains are not memory mapped and are not accessible through the on-chip Advanced Peripheral Bus (APB). Instead, the registers within the JTAG controller are all accessed serially through the TAP Controller. These registers include the Instruction Register and the six Data Registers.

### 4.5.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG TDI and TDO pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the IR. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the IR bits is shown in Table 4-3. A detailed explanation of each instruction, along with its associated Data Register, follows.

**Table 4-3. JTAG Instruction Register Commands**

| IR[3:0] | Instruction | Description |
|---|---|---|
| 0x0 | EXTEST | Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction onto the pads. |
| 0x2 | SAMPLE / PRELOAD | Captures the current I/O values and shifts the sampled values out of the Boundary Scan Chain while new preload data is shifted in. |
| 0x8 | ABORT | Shifts data into the ARM Debug Port Abort Register. |
| 0xA | DPACC | Shifts data into and out of the ARM DP Access Register. |
| 0xB | APACC | Shifts data into and out of the ARM AC Access Register. |
| 0xE | IDCODE | Loads manufacturing information defined by the *IEEE Standard 1149.1* into the IDCODE chain and shifts it out. |
| 0xF | BYPASS | Connects TDI to TDO through a single Shift Register chain. |
| All Others | Reserved | Defaults to the BYPASS instruction to ensure that TDI is always connected to TDO. |

#### 4.5.1.1 EXTEST Instruction

The EXTEST instruction is not associated with its own Data Register chain. Instead, the EXTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the EXTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the outputs and output enables are used to drive the GPIO pads rather than the signals coming from the core. With tests that drive known values out of the controller, this instruction can be used to verify connectivity. While the EXTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

#### 4.5.1.2 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction connects the Boundary Scan Data Register chain between TDI and TDO. This instruction samples the current state of the pad pins for observation and preloads new test data. Each GPIO pad has an associated input, output, and output enable signal. When the TAP controller enters the Capture DR state during this instruction, the input, output, and output-enable signals to each of the GPIO pads are captured. These samples are serially shifted out on TDO while the TAP controller is in the Shift DR state and can be used for observation or comparison in various tests.

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST instruction to drive data into or out of the controller. See “Boundary Scan Data Register” on page 229 for more information.

#### 4.5.1.3 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. See the "ABORT Data Register" on page 230 for more information.

#### 4.5.1.4 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. See "DPACC Data Register" on page 230 for more information.

#### 4.5.1.5 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. See "APACC Data Register" on page 229 for more information.

#### 4.5.1.6 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure input and output data streams. IDCODE is the default instruction loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, or the Test-Logic-Reset state is entered. See "IDCODE Data Register" on page 228 for more information.

#### 4.5.1.7 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. See "BYPASS Data Register" on page 229 for more information.

### 4.5.2 Data Registers

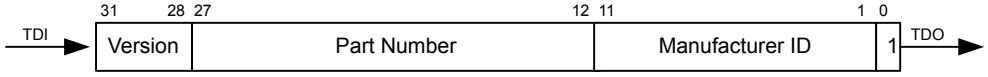
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

#### 4.5.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 4-3. The standard requires that every JTAG-compliant microcontroller implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

The major uses of the JTAG port are for manufacturer testing of component assembly and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x4BA0.0477. This value allows the debuggers to automatically configure themselves to work correctly with the Cortex-M4F during debug.

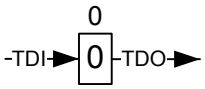
**Figure 4-3. IDCODE Register Format**




### 4.5.2.2 BYPASS Data Register

The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 4-4. The standard requires that every JTAG-compliant microcontroller implement either the BYPASS instruction or the IDCODE instruction as the default instruction. The LSB of the BYPASS Data Register is defined to be a 0 to distinguish it from the IDCODE instruction, which has an LSB of 1. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

**Figure 4-4. BYPASS Register Format**

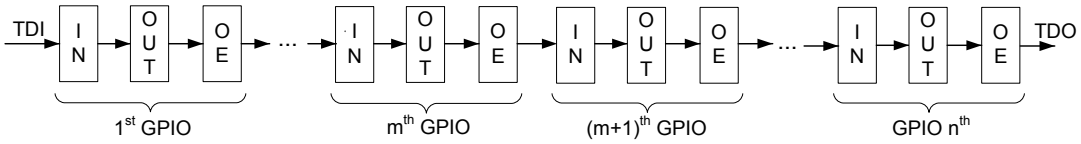



### 4.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 4-5. Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as shown in the figure.

When the Boundary Scan Data Register is accessed with the SAMPLE/PRELOAD instruction, the input, output, and output enable from each digital pad are sampled and then shifted out of the chain to be verified. The sampling of these values occurs on the rising edge of TCK in the Capture DR state of the TAP controller. While the sampled data is being shifted out of the Boundary Scan chain in the Shift DR state of the TAP controller, new data can be preloaded into the chain for use with the EXTEST instruction. The EXTEST instruction forces data out of the controller.

**Figure 4-5. Boundary Scan Register Format**




### 4.5.2.4 APACC Data Register

The format for the 35-bit APACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

#### 4.5.2.5 DPACC Data Register

The format for the 35-bit DPACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

#### 4.5.2.6 ABORT Data Register

The format for the 35-bit ABORT Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

# 5 System Control

System control configures the overall operation of the device and provides information about the device. Configurable features include reset control, NMI operation, power control, clock control, and low-power modes.

## 5.1 Signal Description

The following table lists the external signals of the System Control module and describes the function of each. The NMI signal is the alternate function for two GPIO signals, which default to GPIO after reset. The NMI pins are under commit protection and require a special process to be configured as any alternate function or to subsequently return to the GPIO function. See “Commit Control” on page 784 for more information. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the NMI signal. The number in parentheses next to the pin placement listed is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the NMI signal to the specified GPIO port pin. In addition, the AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the NMI function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772. The remaining signals listed in the table (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

**Table 5-1. System Control & Clocks Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| DIVSCLK | A13 | PQ4 (7) | O | TTL | An optionally divided reference clock output based on a selected clock source. Note that this signal is not synchronized to the System Clock. |
| GNDX2 | D18 | fixed | - | Power | GND for the MOSC. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| NMI | B2<br>B7 | PD7 (8)<br>PE7 (8) | I | TTL | Non-maskable interrupt. |
| OSC0 | E19 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | D19 | fixed | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| RST | P18 | fixed | I | TTL | System reset input. |

## 5.2 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 232
- Configurable control of reset, power, and clock sources.
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 251

### 5.2.1 Device Identification

Read-only registers in the system control module provide information about the microcontroller, such as version, part number, pin count, operating temperature range and available peripherals on the device. The **Device Identification 0 (DID0)** (page 267) and **Device Identification 1 (DID1)** (page 269) registers provide details about the device's version, package, temperature range, and so on. The Peripheral Present registers starting at system control offset 0x300, such as the **Watchdog Timer Peripheral Present (PPWD)** register, provide information on how many of each type of module are included on the device. Finally, information about the capabilities of the on-chip peripherals are provided at offset 0xFC0 in each peripheral's register space in the Peripheral Properties registers, such as the **GPTM Peripheral Properties (GPTMPP)**. In addition, there are four unique identifier registers, **Unique Identifier n (UNIQUEIDn)**, that provide a 128-bit unique identifier for each device that cannot be modified.

### 5.2.2 Reset Control

This section discusses aspects of hardware functions during reset as well as system software requirements following the reset sequence.

#### 5.2.2.1 Reset Sources

The TM4C129XKCZAD microcontroller has the following reset sources:

1. Power-on reset (POR) (see page 233).
2. External reset input pin ($\overline{\text{RST}}$) assertion (see page 234).
3. A brown-out detection of $V_{DDA}$ (analog voltage source) or $V_{DD}$ (external voltage source) dropping below its acceptable operating range. (see page 236).
4. Software-initiated reset (with the software reset registers) (see page 237).
5. A watchdog timer reset condition violation (see page 237).
6. Hibernation module event
7. A software restart initiated through a Hardware System Service Request (HSSR)
8. MOSC failure (see page 240).

Table 5-2 provides a summary of results of the various reset operations. Note that the external $\overline{\text{RST}}$ pin, the Brown-out detection unit, the HIB module and watchdog timer can all be programmed to generate either a Power-On Reset (POR) or system reset depending on how the **Reset Behavior Control (RESBEHAVCTL)** register at offset 0x1D8 is programmed.

**Table 5-2. Reset Sources**

| Reset Source | Core Reset? | JTAG Reset? | On-Chip Peripherals Reset? |
|---|---|---|---|
| Externally Generated Power-On Reset | Yes | Yes | Yes |
| $\overline{\text{RST}}$ pin Power-On Reset | Yes | Pin Configuration Only | Yes |
| $\overline{\text{RST}}$ pin System Reset | Yes | Pin Configuration Only | Yes |
| Brown-Out Power-On Reset | Yes | Pin Configuration Only | Yes |
| Brown-Out System Reset | Yes | Pin Configuration Only | Yes |

**Table 5-2. Reset Sources *(continued)***

| Reset Source | Core Reset? | JTAG Reset? | On-Chip Peripherals Reset? |
|---|---|---|---|
| Software System Reset Request using the SYSRESREQ bit in the **APINT** register. | Yes | Pin Configuration Only | Yes |
| Software System Reset Request using the VECTRESET bit in the **APINT** register. | Yes | No | No |
| Software Peripheral Reset | No | Pin Configuration Only | Yes[a] |
| Watchdog Power-On Reset | Yes | Pin Configuration Only | Yes |
| Watchdog System Reset | Yes | Pin Configuration Only | Yes |
| HIB Module Power-On Reset | Yes | Pin Configuration Only | Yes |
| HIB Module System Reset | Yes | Pin Configuration Only | Yes |
| HSSR Reset | Yes | Pin Configuration Only | Yes |
| MOSC Failure Reset | Yes | Pin Configuration Only | Yes |

a. Programmable on a module-by-module basis by using the individual peripheral Software Reset Registers starting at System Control offset 0x500

After a reset, the **Reset Cause (RESC)** register is set with the reset cause. The bits in this register are sticky and maintain their state across multiple reset sequences. A bit in the **RESC** register can be cleared by writing a 0.

#### 5.2.2.2 Boot Configuration

After Power-On-Reset (POR) and device initialization occurs, the hardware loads the stack pointer from either flash or ROM based on the presence of an application in flash and the state of the EN bit in the **BOOTCFG** register. If the flash address 0x0000.0004 contains an erased word (value 0xFFFF.FFFF) or the EN bit is of the **BOOTCFG** register is clear, the stack pointer and reset vector pointer are loaded from ROM at address 0x0100.0000 and 0x0100.0004, respectively. The boot loader executes and configures the available boot slave interfaces and waits for an external memory to load its software.

If the check of the Flash at address 0x0000.0004 contains a valid reset vector value and the **BOOTCFG** register does not indicate the boot loader, the boot sequence causes the stack pointer/reset vector fetch from Flash. This application stack pointer and reset vector is loaded and the processor executes the application directly.

**Note:** If the device fails the initialization phase, it toggles the TDO output pin as an indication the device is not executing. This feature is provided for debug purposes.

#### 5.2.2.3 Externally Generated Power-On Reset (POR)

**Note:** The JTAG controller can be reset by a power-on reset or by holding the TMS pin to high for 5 clock cycles.

During an externally generated POR, the internal Power-On Reset (POR) circuit monitors the power supply voltage ($V_{DD}$) and generates a reset signal to all of the internal logic including JTAG when the power supply ramp reaches a threshold value ($V_{POR}$). Reset does not complete if specific voltage parameters are not met as defined in the Electrical Characteristics chapter. For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the $\overline{RST}$ input may be used as discussed in “External $\overline{RST}$ Pin” on page 234. Holding this pin active can keep the initialization process from starting even though power-on reset has occurred.

This is useful in in-circuit testing and other situations where it is desirable to delay the operation of the device until an external supervisor has released.

The Power-On Reset sequence is as follows:

1. The microcontroller waits for internal POR to go inactive.
2. The internal reset is released and the core executes a full initialization of the device. Upon completion, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The internal POR is only active on the initial power-up of the microcontroller, when the microcontroller wakes from hibernation, and when the VDD supply drops below the its defined operating limit. Please refer to the Electrical Characteristics chapter for information on exact values. The Power-On Reset timing is shown in “Power and Brown-Out” on page 2103.

#### 5.2.2.4 External $\overline{\text{RST}}$ Pin

When the external $\overline{\text{RST}}$ pin is asserted it initiates a system reset or Power-On Reset depending on what has been configured in the **Reset Behavior Control (RESBEHAVCTL)** Register. If the EXTRES bit field in **RESBEHAVCTL** is set to 0x3 then a simulated full initialization will begin upon $\overline{\text{RST}}$ assertion. If these bits are programmed to 0x2 then a system reset is issued. When EXTRES is set to a 0x0 or 0x1, then the external $\overline{\text{RST}}$ pin performs its default operation upon assertion, which is issuing a full simulated POR.

An external reset pin ($\overline{\text{RST}}$) that is configured to generate a Power-On Reset resets the microcontroller including the core and all the on-chip peripherals. The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted for the duration specified by $T_{MIN}$ and then deasserted (see “Reset” on page 2108). This generates an internal POR signal.
2. The microcontroller waits for internal POR to go inactive.
3. The internal reset is released and the core executes a full initialization of the device. Upon completion, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution. Refer to “Reset” on page 2108 for internal reset deassertion timing.

An external reset pin ($\overline{\text{RST}}$) that is configured to generate a system reset will reset the microcontroller including the core and all the on-chip peripherals. The external reset sequence is as follows:

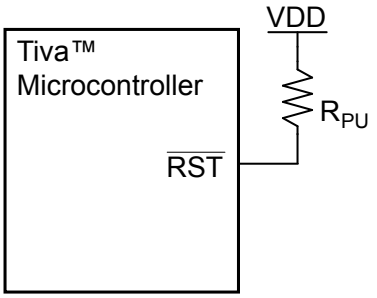
1. The external reset pin ($\overline{\text{RST}}$) is asserted for the duration specified by $T_{MIN}$ and then deasserted (see “Reset” on page 2108).
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

**Note:** It is recommended that the trace for the $\overline{\text{RST}}$ signal must be kept as short as possible. Be sure to place any components connected to the $\overline{\text{RST}}$ signal as close to the microcontroller as possible.

If the application only uses the internal POR circuit, the $\overline{\text{RST}}$ input must be connected to the power supply ($V_{DD}$) through an optional pull-up resistor (0 to 100K Ω) as shown in Figure 5-1 on page 235. The $\overline{\text{RST}}$ input has filtering which requires a minimum pulse width in order for the reset pulse to be recognized, see Table 32-14 on page 2108.

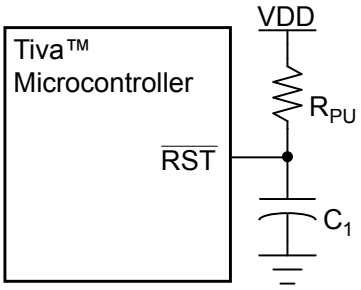
To improve noise immunity and/or to delay reset at power up, the RST input may be connected to an RC network as shown in Figure 5-2 on page 235. If the application requires the use of an external reset switch, Figure 5-3 on page 235 shows the proper circuitry to use. In the figures, the $R_{PU}$ and $C_1$ components define the power-on delay. The external reset timing is shown in Figure 32-11 on page 2109.

**Figure 5-1. Basic $\overline{\text{RST}}$ Configuration**




**Note:** $R_{PU}$ = 0 to 100 kΩ

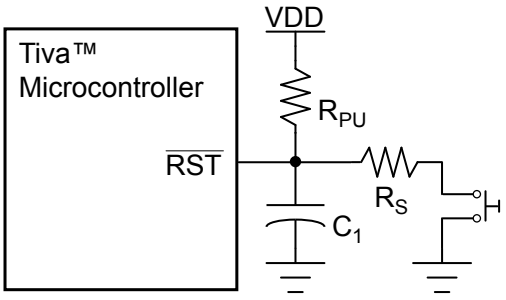
**Figure 5-2. External Circuitry to Extend Power-On Reset**




**Note:** $R_{PU}$ = 1 kΩ to 100 kΩ

$C_1$ = 1 nF to 10 µF

**Figure 5-3. Reset Circuit Controlled by Switch**




**Note:** Typical $R_{PU}$ = 10 kΩ

Typical $R_S$ = 470 Ω

$C_1$ = 10 nF

### 5.2.2.5 Brown-Out Reset (BOR)

The microcontroller provides a brown-out detection circuit that triggers if the $V_{DD}$ (external) or $V_{DDA}$ (analog) power supply drops below its corresponding brown-out threshold voltage. If a brown-out condition is detected, the system may generate an interrupt, a system reset or a Power-On Reset. The default value at reset is to generate an interrupt.

The application can identify the type of BOR event that occurred by reading the **Power-Temperature Cause (PWRTC)** register. The BOR detection circuits can be programmed to generate a reset, System Control interrupt, or NMI in the **Power-Temp Brown Out Control (PTBOCTL)** register. The default settings at reset are as follows:

- $V_{DDA}$ under BOR detection default setting is for no action to occur.
- $V_{DD}$ under BOR detection default setting is to execute a full POR.

If the user has programmed a field in the **PTBOCTL** to generate a reset, then the BOR bit of the **Reset Behavior Control (RESBEHAVCTL)** register can be programmed to further define what type of reset is generated. If the BOR field is programmed to 0x3, a full POR is initiated; if is set to 0x2, then a system reset is issued. When the BOR field is set to a 0x0 or 0x1, then the Brown-Out detection circuit will perform its default operation upon assertion, which is issuing an interrupt.

**Note:** $V_{DDA}$ BOR and $V_{DD}$ BOR events are a combined BOR to the system logic, such that if either BOR event occurs, the following bits are affected:

- BORRIS bit in the **Raw Interrupt Status (RIS**) register, System Control offset 0x050. See page 273.
- BORMIS bit in the **Masked Interrupt Status and Clear (MISC)** register, System Control offset 0x058. This bit is set only if the BORIM bit in the **Interrupt Mask Control (IMC)** register has been set. See page 275 and page 277.
- BOR bit in the **Reset Cause (RESC)** register, System Control offset 0x05C. This bit is set only if either of the BOR events have been configured to initiate a reset. See page 279.

In addition, the following bits control both BOR events:

- BORIM bit in the **Interrupt Mask Control (IMC)** register, System Control offset 0x054.
- VDDA_UBOR0 and VDD_UBOR0 bits in the **Power-Temperature Cause (PWRTC)** register.

Please refer to “System Control” on page 231 for more information on how to configure these registers.

The brown-out POR reset sequence is as follows:

1. When one of the BOR event triggers occurs, an internal Brown-Out Reset condition is set.
2. If the BOR event has been programmed to generate a reset in the **PTBOCTL** register and the BOR bit of the **RESBEHAVCTL** has been set to 0x3, an internal POR reset is asserted.
3. The internal reset is released and the core executes a full initialization of the device. Upon completion, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution. The application starts after deassertion of internal POR. Refer to “Reset” on page 2108 for BOR internal reset deassertion timing.

The brown-out system reset sequence is as follows:

1. When one of the BOR event triggers occurs, an internal Brown-Out Reset condition is set.
2. If the BOR event has been programmed to generate a reset in the **PTBOCTL** register and the BOR bit of the **RESBEHAVCTL** has been set to 0x2, an internal reset is asserted.
3. The internal reset is released and the microcontroller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

The result of a brown-out reset is equivalent to that of an assertion of the external $\overline{\text{RST}}$ input, and the reset is held active until the proper voltage level is restored. The **RESC** register can be examined in the reset interrupt handler to determine if a Brown-Out condition was the cause of the reset, thus allowing software to determine what actions are required to recover.

The internal Brown-Out Reset timing is shown in “Power and Brown-Out” on page 2103.

#### 5.2.2.6 Software Reset

Software can reset a specific peripheral or generate a reset to the entire microcontroller.

Peripherals can be individually reset by software via peripheral-specific reset registers available beginning at System Control offset 0x500 (for example the **Watchdog Timer Software Reset (SRWD)** register page 368). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset.

The entire microcontroller, including the core, can be reset by software by setting the SYSRESREQ bit in the **Application Interrupt and Reset Control (APINT)** register in the core peripheral memory map space. The software-initiated system reset sequence is as follows:

1. A software microcontroller reset is initiated by setting the SYSRESREQ bit.
2. An internal reset is asserted.
3. The internal reset is deasserted and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The core only can be reset by software by setting the VECTRESET bit in the **APINT** register. The software-initiated core reset sequence is as follows:

1. A core reset is initiated by setting the VECTRESET bit.
2. An internal reset is asserted.
3. The internal reset is deasserted and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The software-initiated system reset timing is shown in Figure 32-12 on page 2109.

#### 5.2.2.7 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The TM4C129XKCZAD microcontroller has two Watchdog Timer modules in case one watchdog clock source fails. One watchdog is run off the system clock and the other is run off the Precision Internal Oscillator (PIOSC).

The watchdog timer can be configured to generate an interrupt or a non-maskable interrupt to the microcontroller on its first time-out and to generate a system reset or power-on reset on its second time-out.

After the watchdog's first time-out event, the 32-bit watchdog counter is reloaded with the value of the **Watchdog Timer Load (WDTLOAD)** register and resumes counting down from that value. If the timer counts down to zero again before the first time-out interrupt is cleared, and watchdog reset generation has been enabled through the RESEN bit in the **Watchdog Control Register (WDTCTL)**, the watchdog timer asserts its reset signal to the microcontroller. The reset generated can be a full Power-On Reset or a system reset depending on the value programmed in WDOGn bit field of the **Reset Behavior Control Register (RESBEHAVCTL)**. If the RESEN bit of the **WDTCTL** register is set to 1 and the WDOGn bit field of the **RESBEHAVCTL** register is programmed to 0x3 a full POR is initiated; if WDOGn set to 0x2, then a system reset is issued. When WDOGn is set to a 0x0 or 0x1, then the watchdog time performs its default operation upon assertion, which is issuing a full POR.

The watchdog timer Power-On Reset sequence is as follows:

1. The watchdog timer times out for the second time without being serviced.
2. An internal POR reset is asserted.
3. The internal reset is released and the core executes a full initialization of the device. Upon completion, the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution. Refer to “Reset” on page 2108 for watchdog timeout internal reset deassertion timing.

The watch dog timer system reset sequence is as follows:

1. The watchdog timer times out for the second time without being serviced.
2. An internal reset is asserted.
3. The internal reset is released and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

For more information on the Watchdog Timer module, see “Watchdog Timers” on page 1178.

The watchdog reset timing is shown in Figure 32-13 on page 2109.

#### 5.2.2.8 Hibernation Module Reset

When the Hibernation module has been configured and powered by an initial "cold" POR and is subsequently put into hibernation mode, a wake event (not including an external reset pin wake) causes the module to generate a system reset. This reset signal resets all circuitry on the device with the exception of the Hibernation module. All Hibernation module registers retain their values after this reset.

When the Hibernation module receives a wake event and $V_{DD}$ is enabled, a system reset sequence occurs as follows:

1. The POR or EXT bit in the **RESC** register is set.
2. An internal reset is asserted.

3. The internal reset is released and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

4. The **HIBRIS** register in the Hibernation module can be read to determine the cause of the reset.

5. The POR or EXT bit in the **RESC** register is cleared by writing a 0.

#### 5.2.2.9 HSSR Reset

The **Hardware System Service Request (HSSR)** register can be used to restore the device back to factory settings. A successful write to the **Hardware System Service Request (HSSR)** register initiates a system reset. The reset initialization process executes before examining the **HSSR** register and processing the command. This register can only be accessed in privileged mode.

Before the return-to-factory settings routine has completed, a system reset sequence executes and the HSSR bit in the **RESC** register is set. After the HSSR function has been processed, the CDOFF field in the **HSSR** register is written with the outcome of the function processing and another HSSR system reset is executed. The HSSR bit can be cleared in the **RESC** register by writing a 0.

For more information regarding use of the **HSSR** register, refer to “Hardware System Service Request” on page 256.

### 5.2.3 Non-Maskable Interrupt

The microcontroller has multiple sources of non-maskable interrupt (NMI):

- The assertion of the NMI signal.
- A main oscillator verification error.
- The NMISET bit in the **Interrupt Control and State (INTCTRL)** register in the Cortex™-M4F (see page 178).
- The Watchdog module time-out interrupt when the INTTYPE bit in the **Watchdog Control (WDTCTL)** register is set (see page 1184).
- Tamper event (see “Hibernation Module” on page 563 for more information).
- Any of the following BOR trigger events:
  - $V_{DDA}$ under BOR setting
  - $V_{DD}$ under BOR setting

Software must check the cause of the interrupt in the **NMI Cause (NMIC)** register in order to distinguish among the sources.

#### 5.2.3.1 NMI Pin

The NMI signal is an alternate function for the GPIO port pin(s) specified in Table 31-3 on page 2048.

The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in “General-Purpose Input/Outputs (GPIOs)” on page 772. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function, similar to the requirements of the GPIO port pins associated with JTAG/SWD functionality, see page 817. The active sense of

the NMI signal is High; asserting the enabled NMI signal above $V_{IH}$ initiates the NMI interrupt sequence.

#### 5.2.3.2 Main Oscillator Verification Failure

The TM4C129XKCZAD microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or too slow. If the main oscillator verification circuit is enabled and a failure occurs, either a power-on reset is generated and control is transferred to the NMI handler, or an interrupt is generated. The MOSCIM bit in the **MOSCCTL** register determines which action occurs. In either case, the system clock source is automatically switched to the PIOSC. If a MOSC failure reset occurs, the NMI handler is used to address the main oscillator verification failure because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled by setting the CVAL bit in the **Main Oscillator Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status (MOSCFAIL) bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in the section called “Main Oscillator Verification Circuit” on page 248.

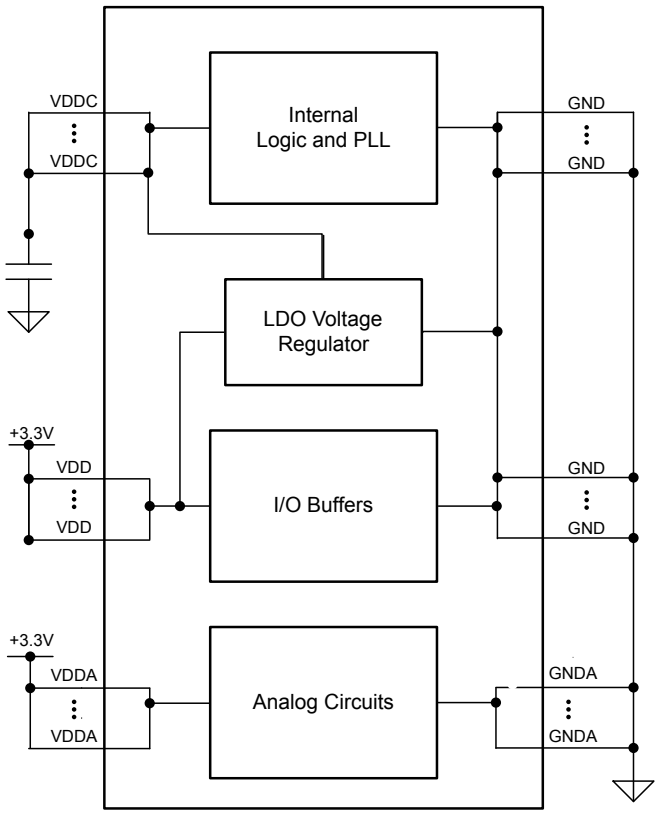
### 5.2.4 Power Control

The TM4C129XKCZAD microcontroller provides an integrated LDO regulator that is used to provide power to the majority of the microcontroller's internal logic. Figure 5-4 shows the power architecture. The voltage output has a maximum voltage of 1.2 V. Refer to “Dynamic Power Management” on page 254 for more information on the LDO operation.

An external LDO may not be used.

**Note:** VDDA must be supplied with 3.3 V, or the microcontroller does not function properly. VDDA is the supply for all of the analog circuitry on the device, including the clock circuitry.

**Figure 5-4. Power Architecture**




**Note:** The $V_{DDA}$ voltage source is typically connected to a filtered voltage source or regulator.

## 5.2.5 Clock Control

The system control module determines the control of clocks in this part.

### 5.2.5.1 Fundamental Clock Sources

There are multiple clock sources for use in the microcontroller. The **Run and Sleep Mode Configuration Register (RSCLKCFG)** can be used to configure the required clock source for the device after Power-On Reset, as well as the system clock divisor encodings. The available clock sources are as follows:

- **Precision Internal Oscillator (PIOSC).** The precision internal oscillator is an on-chip clock source that the microcontroller uses during and following POR. It is the clock source in effect at the start of reset vector fetch and the start of code application execution. It does not require the use of any external components and provides a clock that is 16 MHz ±$F_{PIOSC}$ across temperature (see Table 32-19 on page 2114). The PIOSC allows for a reduced system cost in applications that require an accurate enough clock source. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference. If the Hibernation Module clock source is a 32.768-kHz oscillator, the precision internal oscillator can be trimmed by software based on a reference clock for increased accuracy. Regardless of whether or not the PIOSC is the source for the system clock, the PIOSC can be

configured to be an alternate clock source for some of the peripherals. See the section called "Peripheral Clock Sources" on page 245 for more information on peripherals that can use the PIOSC as an alternate clock.

- **Main Oscillator (MOSC).** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the crystal value can be any frequency between 5 MHz to 25 MHz (inclusive). Refer to Table 5-7 on page 250 for recommended crystal values and PLL register programming. If the PLL is not being used, the crystal may be any one of the supported frequencies between 4 MHz to 25 MHz. The single-ended clock source range is from DC through the specified speed of the microcontroller.

- **Low-Frequency Internal Oscillator (LFIOSC).** The Low-Frequency Internal Oscillator (LFIOSC) provides a nominal frequency of 33 kHz with percentage variance specified in the Electrical Characteristics section. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode provides reduced internal switching and the ability to power down the MOSC and/or PIOSC while in Deep-Sleep mode through configuration of the **Deep Sleep Clock Configuration Register (DSCLKCFG)** register.

- **Hibernation Module RTC Oscillator (RTCOSC) Clock Source.** The Hibernation Module provides a muxed output of two clocks to the System Control Module, an external 32.768-kHz clock or a low-frequency clock (HIB LFIOSC). The Hibernation module has the option of being clocked by a 32.768-kHz oscillator connected to the XOSC0 pin. The 32.768-kHz oscillator can be used for the system clock, thus eliminating the need for an additional crystal or oscillator. Alternatively, the Hibernation module contains a low-frequency oscillator (HIB LFIOSC) which is intended to provide the system with a real-time clock source and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings. Note that the HIB LFIOSC is a different clock source than the LFIOSC. Refer to the Electrical Characteristic Chapter for more information on frequency range.

The internal system clock (SysClk), is derived from any of the above sources. An internal PLL can also be used by the PIOSC or MOSC clock to generate the system clock and peripheral clocks. Table 5-3 on page 242 shows how the various clock sources can be used in a system.

**Table 5-3. Clock Source Options**

| Clock Source | Drive PLL Capability? | PLL Enabled, RSCLKCFG Bit Encodings | SysClk generation capability? | SysClk generation enabled, RSCLKCFG Bit Encodings |
|---|---|---|---|---|
| Precision Internal Oscillator (PIOSC) | Yes | USEPLL = 1, PLLSRC = 0x0 | Yes | USEPLL = 0, OSCSRC = 0x0 |
| Main Oscillator (MOSC) | Yes | USEPLL = 1, PLLSRC = 0x3 | Yes | USEPLL = 0, OSCSRC = 0x3 |
| Low Frequency Internal Oscillator (LFIOSC)[a] | No | - | Yes | USEPLL = 0, OSCSRC = 0x2 |
| Hibernation Module RTC Oscillator (RTCOSC). 32.768-kHz Oscillator or HIB LFIOSC | No | - | Yes | USEPLL = 0, OSCSRC = 0x4 |

a. LFIOSC frequency is characterized as 33 kHz nominal, 10 kHz minimum and 90 kHz maximum.

### 5.2.5.2 Clock Configuration

The **Run and Sleep Mode Configuration Register (RSCLKCFG)** provides control for the system clock in run and sleep mode. The **Deep Sleep Clock Configuration register (DSCLKCFG)** specifies the behavior of the clock system while in deep sleep mode. These registers control the following clock functionality:

- Source of system clock in run and sleep mode
- Source of system clock in deep-sleep mode
- Enabling/disabling of PLL-derived system clock
- Clock divisors for PLL or oscillator, depending on what is enabled
- Enabling of memory timing parameters for flash

Providing further configuration, the **PLL Frequency n (PLLFREQn)** registers allow the PLL VCO frequency ($f_{VCO}$) to multiplied or divided by programmable values depending on the system clock speed required.

Table 5-4 on page 243 shows the state of the clock sources following a Power-On Reset.

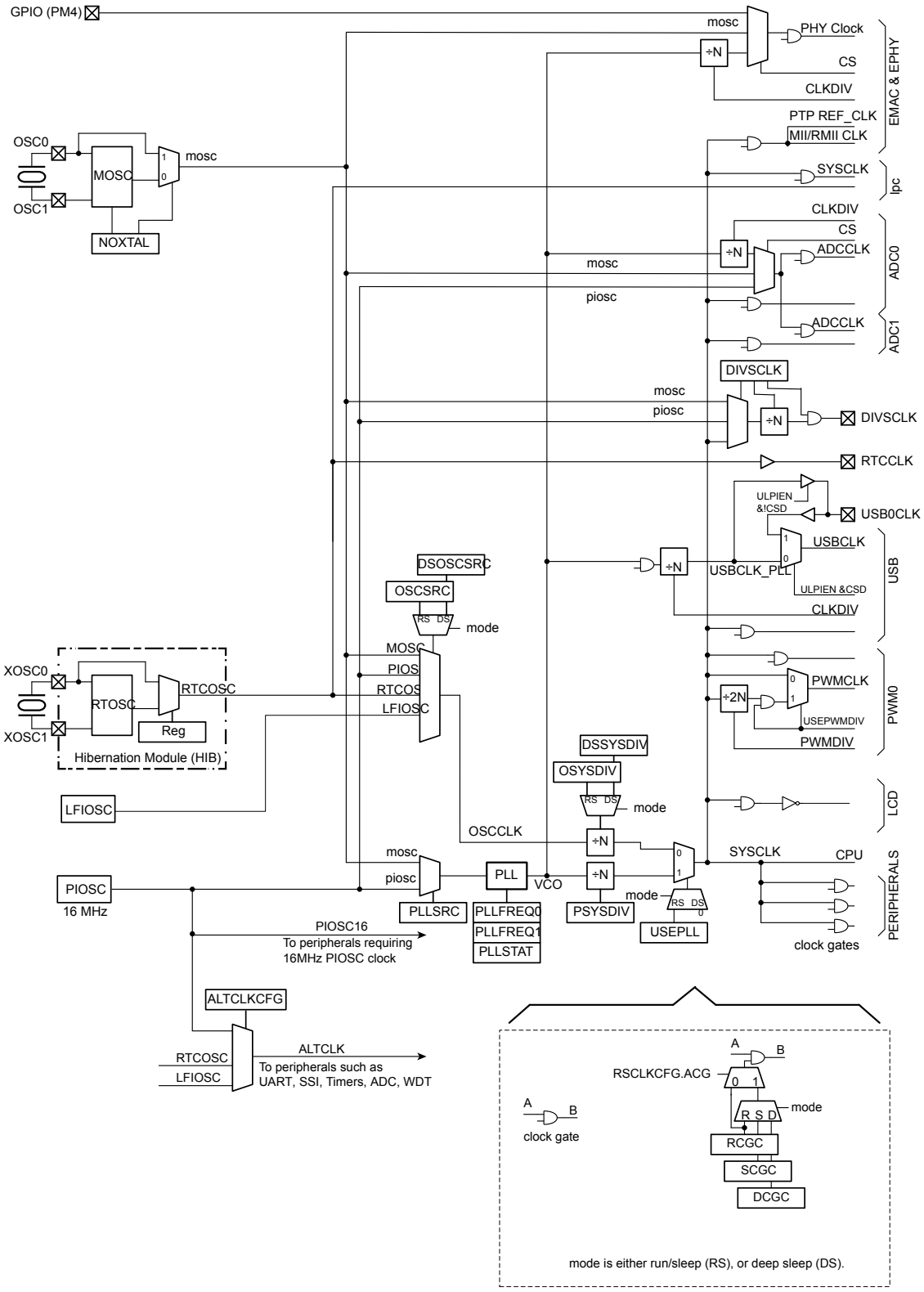
**Table 5-4. Clock Source State Following POR**

| Clock Source | Power-On Reset State |
|---|---|
| PLL | Disabled/Powered Off |
| MOSC | Disabled/Powered Off |
| LFIOSC | Enabled |
| PIOSC | Enabled |
| HIB RTCOSC | Disabled |

Figure 5-5 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled.

**Note:** The clock sources in Figure 5-5 include a superset of peripherals available in the family. Some peripheral clock sources may not be present on your specific device.

**Figure 5-5. Main Clock Tree**



### Peripheral Clock Sources

In addition to the main clock tree described above, the ADC, USB, Ethernet, PWM, UART, and QSSI all have a Clock Control register in their register map at offset 0xFC8 that can be used to control the clock generation for the module.

### ADC Clock Control

The ADC digital block is clocked by the system clock and the ADC analog block is clocked from a separate conversion clock (ADC clock). The ADC clock frequency can be up to 32 MHz to generate a conversion rate of 2 Msps. A 16 MHz ADC clock provides a 1 Msps sampling rate. There are three sources for the ADC clock:

- The PLL VCO ($f_{VCO}$) can be used if the CS bit field is 0x0 in the **ADC Clock Configuration (ADCCC)** register and the CLKDIV bit field is configured in the same register.
- The PIOSC can be used directly to provide a conversion rate near 1 Ms/s. To use the PIOSC, the CS field in the **ADCCC** register needs to be set to 0x1 and the ALTCLK field should be programmed to 0x0 in the **Alternate Clock Configuration (ALTCLKCFG)** register.
- The Main Oscillator (MOSC): The MOSC clock source must be 16 MHz for a 1 Msps conversion rate and 32 MHz for a 2 Msps conversion rate.

**Note:** If the ADC module is not using the PIOSC as the clock source, the system clock must be at least 16 MHz.

### USB Clock Control

When the USB module uses the integrated USB PHY, the MOSC must be the clock source, either with or without using the PLL, and the system clock must be at least 30 MHz. In addition, only integer divisors should be used to achieve the 60 MHz USB clock source. Fractional divisors may increase jitter and compromise USB function. The **USB Clock Control Register (USBCC)** register contains a CLKDIV bit field which can be programmed to specify the divisor used to reduce the PLL VCO output to the 60 MHz clock source required for the serialization/deserialization module of the USB controller.

In ULPI mode, if the clock source to the USB is internal, then the USB0CLK pin is an output to the external ULPI PHY. If the USB clock source is external then, the USB0CLK pin functions as an input from the external ULPI PHY.

### Ethernet Clock Control

Available clock sources are dependent on the interface chosen. The following sections describe the clock control for the various interfaces.

The Ethernet Controller Module and Integrated PHY receive two clock inputs. A gated system clock acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC and must be 20 MHz or greater for correct operation. The SYSCLK frequency for Run, Sleep and Deep Sleep mode is programmed in the System Control module. See “Ethernet Clock Control” on page 1591 for more information.

### PHY Interface Clocking

The Ethernet Controller Module and Integrated PHY receive two clock inputs (see “PHY Interface” on page 1591 for more information):

- A gated system clock acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SysClk frequency for Run, Sleep and Deep Sleep mode is programmed in the System Control module.
- The PHY receives the main oscillator (MOSC) which must be 25 MHz ± 50 ppm for proper operation. The MOSC source can be a single-ended source or a crystal.

### *MII Interface Clocking*

Four clock inputs are driven into the Ethernet MAC when the MII configuration is enabled. The clocks are described as follows (see "MII Interface" on page 1592 for more information):

- Gated system clock (SysClk): The SysClk signal acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SysClk frequency for Run, Sleep and Deep Sleep mode is programmed in the System Control module.
- MOSC: A gated version of the MOSC clock is provided as the Precision Time Protocol (PTP) reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the Precision Timer Protocol (PTP) module has been enabled by setting the PTPCEN bit in the **EMACCC** register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. Refer to "IEEE 1588 and Advanced Timestamp Function" on page 1625 for more information.
- EN0RXCK: This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz depending on whether the device is operating at 10 Mbps or 100 Mbps.
- EN0TXCK This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz depending on whether the device is operating at 10 Mbps or 100 Mbps.

### *RMII Interface Clocking*

There are three clock sources that interface to the Ethernet MAC in an RMII configuration (see "RMII Interface" on page 1593 for more information):

- Gated system clock (SysClk): The SysClk signal acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SysClk frequency for Run, Sleep and Deep Sleep mode is programmed in the System Control module.
- MOSC: A gated version of the MOSC clock is provided as the Precision Time Protocol (PTP) reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the PTP module has been enabled by setting the PTPCEN bit in the **EMACCC** register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. Refer to "IEEE 1588 and Advanced Timestamp Function" on page 1625 for more information.
- EN0REF_CLK: When using RMII, a 50 MHz external reference clock must drive the EN0REF_CLK input signal and the external PHY. Depending on the configuration of the FES bit in the **Ethernet MAC Configuration (EMACCFG)** register, the reference clock input (EN0REF_CLK) is divided by 20 for 10 Mbps or 2 for 100 Mbps operation and used as the clock for receive and transmit data.

#### *PWM Clock Control*

The **PWMCC** register can be used to select the System Clock as the PWM clock source or a divided System Clock. For more information, see page 2003.

#### *Other Peripheral Clock Control*

In the UART and QSSI Clock Control Registers, users can choose between the system clock (SysClk), which is the default source for the baud clock, and an alternate clock. Note that there may be special considerations when configuring the baud clock.

#### *Optional Clock Output Signal (DIVSCLK)*

An optional clock output, DIVSCLK, is provided which can be used as a clock source to an external device but bears no timing relationship to other signals. Note that this signal is not synchronized to the System Clock. By programming the SRC field in the **Divisor and Source Clock Configuration (DIVSCLK)** register, the following clock outputs may be selected for DIVSCLK:

- System Clock
- PIOSC
- MOSC

The DIV field in the **DIVSCLK** register controls the divided output clock frequency. The DIVSCLK signal is selected as an alternate function of a GPIO signal and has the same inherit electrical characteristics of a GPIO as listed in “Electrical Characteristics” on page 2095.

#### *System Clock Frequency*

The system clock (SysClk) is the clock that is distributed to the processor and the integrated peripherals after clock gating. The SysClk frequency is based on the frequency of the clock source and the divisor factor. For example, if the PLL is not being used and the device is not in deep sleep mode, then the OSYSDIV bit field in the **RSCLKCFG** register is the divisor used to determine the system clock. If the PLL is being used, then PSYSDIV bit field in the **RSCLKCFG** register must be programmed as well as the values in the **PLLFREQ0** and **PLLFREQ1** registers. If the device is in deep sleep mode, then the **Deep Sleep Clock Configuration Register (DSCLKCFG)** can be programmed with the divisor bit field DSSYSYDIV to modify the clock source frequency. Table 5-5 on page 247 shows the different system clock frequency calculations based on the operation mode, clock source and PLL encoding.

**Table 5-5. System Clock Frequency**

| Clock Mode | USEPLL (RSCLKCFG) | SYSCLK Value | Divisor Factors Used |
|---|---|---|---|
| Run or Sleep | 1 | $f_{VCO}$/(PSYSDIV + 1) | PSYSDIV bit field in **RSCLKCFG**; MINT, MDIV in **PLLFREQ0**; Q, N bits in **PLLFREQ1** |
| Run or Sleep | 0 | $f_{OSCCLK}$/(OSYSDIV + 1) | OSYSDIV bit field in **RSCLKCFG** |
| Deep Sleep | PLL not enabled in Deep Sleep | $f_{OSCCLK}$/(DSSYSDIV + 1) | DSSYSDIV bit field in **DSCLKCFG** |

### 5.2.5.3 Precision Internal Oscillator Operation (PIOSC)

The microcontroller powers up with the PIOSC running. If another clock source is desired, the PIOSC must remain enabled because it is used for internal functions. The PIOSC can only be disabled during Deep-Sleep mode. It can be powered down by setting the PIOSCPD bit in the **DSCLKCFG** register.

The PIOSC generates a 16-MHz clock with a $\pm F_{PIOSC}$ accuracy (see Table 32-19 on page 2114). At the factory, the PIOSC is set to 16 MHz at room temperature, however, the frequency can be trimmed for other voltage or temperature conditions using software in the following ways:

- Default calibration: clear the UTEN bit and set the UPDATE bit in the **Precision Internal Oscillator Calibration (PIOSCCAL)** register.
- User-defined calibration: The user can program the UT value to adjust the PIOSC frequency. As the UT value increases, the generated period increases. To commit a new UT value, first set the UTEN bit, then program the UT field, and then set the UPDATE bit. The adjustment finishes within a few clock periods and is glitch free.
- Automatic calibration using the enable 32.768-kHz oscillator from the Hibernation module: Set the CAL bit in the **PIOSCCAL** register; the results of the calibration are shown in the RESULT field in the **Precision Internal Oscillator Statistic (PIOSCSTAT)** register. After calibration is complete, the PIOSC is trimmed using the trimmed value returned in the CT field.

#### 5.2.5.4 Main Oscillator (MOSC)

The main oscillator supports the use of crystals from 5 to 25 MHz. The system control's **RSCLKCFG** register can be configured to specify the MOSC as the system clock or as the PLL input source. The MOSC can be selected as the oscillator source by programming the OSCRC bit in the **RSCLKCFG** register. The NOXTAL bit in the **MOSCCTL** register allows the user to turn off power to the MOSC if no crystal is connected reducing power draw from the MOSC circuit.

***Main Oscillator Verification Circuit***

The clock control includes circuitry to ensure that the main oscillator is running at the appropriate frequency. The circuit monitors the main oscillator frequency and signals if the frequency is outside of the allowable band of attached crystals.

The detection circuit is enabled using the CVAL bit in the **Main Oscillator Control (MOSCCTL)** register. If this circuit is enabled and detects an error, and if the MOSCIM bit in the **MOSCCTL** register is clear, then the following sequence is performed by the hardware:

1. The MOSCFAIL bit in the **Reset Cause (RESC)** register is set.
2. The system clock is switched from the main oscillator to the PIOSC.
3. An internal system reset is initiated.
4. Reset is deasserted and the processor is directed to the NMI handler during the reset sequence.

#### 5.2.5.5 PLL

The PLL has two modes of operation: Normal and Power-Down

- Normal: The PLL oscillates based on the values in the **PLLFREQ0** and **PLLFREQ1** registers and drives the output.
- Power-Down: Most of the PLL internal circuitry is disabled and the PLL does not drive the output.

The modes are programmed using the PLLPWR bit in the **PLLFREQ0** register (see page 304).

### PLL Configuration

The PLL is disabled by default during power-on reset and is enabled later by software if required. Software specifies the output divisor to set the system clock frequency and enables the PLL to drive the output. The PLL is controlled using the **PLLFREQ0**, **PLLFREQ1** and **PLLSTAT** registers. Changes made to these registers do not become active until after the NEWFREQ bit in the **RSCLKCFG** register is enabled.

The clock source for the main PLL is selected by configuring the PLLSRC field in the **Run and Sleep Clock Configuration (RSCLKCFG)** register.

The PLL allows for the generation of system clock frequencies in excess of the reference clock provided. The reference clocks for the PLL are the PIOSC and the MOSC. The PLL is controlled by two registers, **PLLFREQ0** and **PLLFREQ1**. The PLL VCO frequency ($f_{VCO}$) is determined through the following calculation:

$f_{VCO} = f_{IN} * MDIV$

where

$f_{IN} = f_{XTAL}/(Q+1)(N+1)$ or $f_{PIOSC}/(Q+1)(N+1)$

$MDIV = MINT + (MFRAC / 1024)$

The Q and N values are programmed in the **PLLFREQ1** register. Note that to reduce jitter, MFRAC should be programmed to 0x0.

When the PLL is active, the system clock frequency (SysClk) is calculated using the following equation:

$SysClk = f_{VCO}/ (PSYSDIV + 1)$

The PLL system divisor factor (PSYSDIV) determines the value of the system clock. Table 5-6 on page 249 shows how the system divisor encodings affect the system clock frequency when the $f_{VCO}$ = 480 MHz.

**Table 5-6. System Divisor Factors for $f_{vco}$=480 MHz**

| System Clock (SYSCLK) (MHz) | $f_{VCO}$ (MHz)= 480 MHz |
|---|---|
| | System Divisors (PSYSDIV +1)[a] |
| 120 | 4 |
| 60 | 8 |
| 48 | 10 |
| 30 | 16 |
| 24 | 20 |
| 12 | 40 |
| 6 | 80 |

a. The use of non-integer divisors introduce additional jitter which may affect interface performance.

If the main oscillator provides the clock reference to the PLL, the translation provided by hardware and used to program the PLL is available for software in the **PLL Frequency n (PLLFREQn)** registers (see page 304). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency. Table 5-7 on page 250 shows the actual PLL frequency and error for a given crystal choice.

Table 5-7 on page 250 provides examples of the programming expected for the **PLLFREQ0** and **PLLFREQ1** registers. The first column specifies the input crystal frequency and the last column displays the PLL frequency given the values of MINT and N, when Q=0.

**Table 5-7. Actual PLL Frequency**[a]

| Crystal Frequency (MHz) | MINT (Decimal Value) | MINT (Hexadecimal Value) | N | Reference Frequency (MHz)[b] | PLL Frequency (MHz) |
|---|---|---|---|---|---|
| 5 | 64 | 0x40 | 0x0 | 5 | 320 |
| 6 | 160 | 0x35 | 0x2 | 2 | 320 |
| 8 | 40 | 0x28 | 0x0 | 8 | 320 |
| 10 | 32 | 0x20 | 0x0 | 10 | 320 |
| 12 | 80 | 0x50 | 0x2 | 4 | 320 |
| 16 | 20 | 0x14 | 0x0 | 16 | 320 |
| 18 | 160 | 0xA0 | 0x8 | 2 | 320 |
| 20 | 16 | 0x10 | 0x0 | 20 | 320 |
| 24 | 40 | 0x28 | 0x2 | 8 | 320 |
| 25 | 64 | 0x40 | 0x4 | 5 | 320 |
| 5 | 96 | 0x60 | 0x0 | 5 | 480 |
| 6 | 80 | 0x50 | 0x0 | 6 | 480 |
| 8 | 60 | 0x3C | 0x0 | 8 | 480 |
| 10 | 48 | 0x30 | 0x0 | 10 | 480 |
| 12 | 40 | 0x28 | 0x0 | 12 | 480 |
| 16 | 30 | 0x1E | 0x0 | 16 | 480 |
| 18 | 80 | 0x50 | 0x2 | 6 | 480 |
| 20 | 24 | 0x18 | 0x0 | 20 | 480 |
| 24 | 20 | 0x14 | 0x0 | 24 | 480 |
| 25 | 96 | 0x60 | 0x4 | 5 | 480 |

a. For all examples listed, Q=0

b. For a given crystal frequency, N should be chosen such that the reference frequency is within 4 to 30 MHz.

### *PLL Operation*

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is $T_{READY}$ (see Table 32-16 on page 2112). During the relock time, the affected PLL is not usable as a clock reference. Software can poll the LOCK bit in the **PLL Status (PLLSTAT)** register to determine when the PLL has locked.

Modification of the PLL VCO frequency may not be performed while the PLL serves as a clock source to the system. All changes to the PLL must be performed using a different clock source until the PLL has locked frequency. Thus, changing the PLL VCO frequency must be done as a sequence from PLL to PIOSC/MOSC and then PIOSC/MOSC to new PLL.

Hardware is provided to keep the PLL from being used as a system clock until the $T_{READY}$ condition is met after one of the two changes above. It is the user's responsibility to have a stable clock source (like the main oscillator) before the **RSCLKCFG** register is re-programmed to enable the PLL. Software can use many methods to ensure that the system is clocked from the PLL, including periodically polling the PLLLRIS bit in the **Raw Interrupt Status (RIS)** register at offset 0x050, and enabling the PLL Lock interrupt in the **Interrupt Mask Control (IMC)** register at offset 0x054.

### 5.2.6 System Control

There are four levels of operation for the microcontroller defined as:

- Run mode
- Sleep mode
- Deep-Sleep mode
- Hibernation mode

For power-savings purposes, the peripheral-specific **RCGCx**, **SCGCx**, and **DCGCx** registers (for example, **RCGCWD**) control the clock-gating logic for that peripheral or block in the system while the microcontroller is in Run, Sleep, and Deep-Sleep mode, respectively. These registers are located in the System Control register map starting at offsets 0x600, 0x700, and 0x800, respectively.

**Note:** A change in the **RCGCx** (or **SCGCx/DCGCx/PCx/SRx**) registers may not have an immediate effect on the clock in all situations. It is recommended that software poll the peripheral's **Peripheral Ready (PRx)** register to determine when a peripheral is ready to be accessed.

**Note:** If a peripheral is configured to be clock-gated during Run, Sleep- or Deep-Sleep mode, then software should ensure that there are no pending transfers or register accesses before or immediately after entering the clock-gated mode.

The following sections describe the different modes in detail.

#### 5.2.6.1 Run Mode

In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the peripheral-specific **RCGC** registers. In run mode (and in sleep mode), the **Run and Sleep Clock Configuration (RSCLKCFG)** register specifies the source of SysClk. The source is either from the VCO output of the PLL divided down by a dedicated divisor (divisor value specified by the PSYSDIV field) or from the output of an oscillator divided down by a dedicated divisor (divisor value specified by the OSYSDIV field). The source is selected using the USEPLL bit in the **RSCLKCFG** register. The PLL has two sources of reference clock as an input: the main oscillator (MOSC) or the precision internal oscillator (PIOSC). The PLL input select is specified by PLLSRC. If the PLL VCO output is not selected as the source of SysClk then the following reference clocks can be programmed as an input:

- Main Oscillator (MOSC)
- Precision Internal Oscillator (PIOSC)
- Low-Frequency Internal Oscillator (LFIOSC)
- Hibernation Module Real-Time Oscillator Source (RTCOSC): The source of this signal can be either a 32.768-kHz oscillator source, an external 32.768-kHz clock source or the internal Hibernation Module Low-Frequency Oscillator (HIBLFIOSC). If this clock source is selected, it has to be enabled in the Hibernation Module as well.

The selection of these alternate sources is through the OSCSRC field in the **RSCLKCFG** register.

#### 5.2.6.2 Sleep Mode

In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered

by the Cortex-M4F core executing a WFI (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See “Power Management” on page 137 for more details.

Peripherals are clocked that are enabled in the peripheral-specific **SCGC** registers when auto-clock gating is enabled or the peripheral-specific **RCGC** registers when the auto-clock gating is disabled. The system clock has the same source and frequency as that during Run mode.

The option to use the PLL VCO or an alternate oscillator source such as MOSC, PIOSC, Hibernation Module real time clock, or the LFIOSC is the same as described in Run Mode. The **RSCLKCFG** register programming applies to Sleep Mode.

Additional sleep modes are available that lower the power consumption of the SRAM and Flash memory. However, the lower power consumption modes have slower sleep and wake-up times.

---

**Caution – If the Cortex-M4F Debug Access Port (DAP) has been enabled, and the device wakes from a low power sleep or deep-sleep mode, the core may start executing code before all clocks to peripherals have been restored to their Run mode configuration. The DAP is usually enabled by software tools accessing the JTAG or SWD interface when debugging or flash programming. If this condition occurs, a Hard Fault is triggered when software accesses a peripheral with an invalid clock.**

**A software delay loop can be used at the beginning of the interrupt routine that is used to wake up a system from a WFI (Wait For Interrupt) instruction. This stalls the execution of any code that accesses a peripheral register that might cause a fault. This loop can be removed for production software as the DAP is most likely not enabled during normal execution.**

**Because the DAP is disabled by default (power on reset), the user can also power cycle the device. The DAP is not enabled unless it is enabled through the JTAG or SWD interface.**

---

### 5.2.6.3 Deep-Sleep Mode

In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Deep-Sleep mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first setting the SLEEPDEEP bit in the **System Control (SYSCTRL)** register (see page 184) and then executing a WFI instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See “Power Management” on page 137 for more details.

**Note:** If the Debug Access Port is enabled in Run Mode and attempts to transition into Deep-Sleep mode, the device is prevented from entering Deep-Sleep.

The Cortex-M4F processor core and the memory subsystem are not clocked in Deep-Sleep mode. Peripherals are clocked that are enabled in the peripheral-specific **DCGC** registers when auto-clock gating is enabled or the peripheral-specific **RCGC** registers when auto-clock gating is disabled. The system clock source is specified in the **DSCLKCFG** register. When the **DSCLKCFG** register is used, the internal oscillator source is powered up, if necessary, and other clocks are powered down. If the PLL is running at the time of the WFI instruction, hardware shuts down the PLL for power savings. For further power savings the PIOSC can be disabled through the PIOSCPD bit in the **DSCLKCFG** register. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration. If the PIOSC is used as the PLL reference clock source, it may continue to provide the clock during Deep-Sleep. See page 293.

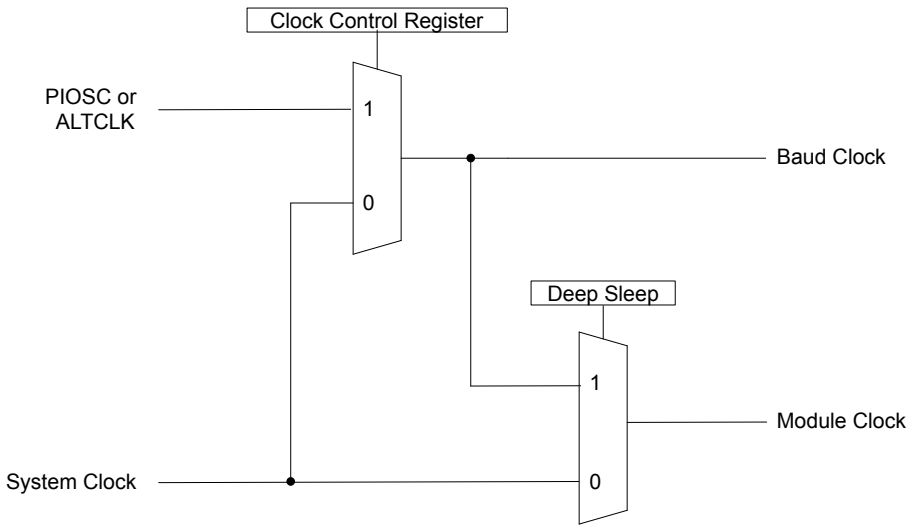
**Note:** If the MOSC is chosen as the Deep-Sleep clock source in the **DSCLKCFG** register, the MOSC must also be configured as the Run and Sleep clock source in the **RSCLKCFG** register prior to entering Deep Sleep. If the PIOSC, LFIOSC, or Hibernation RTC Module

Oscillator (HIBLFIOSC or 32-kHz crystal) is configured as the Run and Sleep clock source in the **RSCLKFCFG** register, and the MOSC is configured as the Deep-Sleep clock source in the **DSCLKCFG** register, then two outcomes are possible:

- If the PIOSC is still powered in Deep Sleep (using the PIOSCPD bit in the **DSCLKCFG** register) then the PIOSC is utilized as the clock source when entering Deep Sleep and the device enters and exits the Deep-Sleep state normally. The MOSC is not used as the clock source in Deep Sleep.
- If the PIOSC has been configured to be powered down in Deep Sleep, then the device can enter the Deep-Sleep state, but cannot exit properly. This situation can be avoided by programming the MOSC as the Run and Sleep clock source in the **RSCLKCFG** register prior to entering Deep Sleep.

To provide the lowest possible Deep-Sleep power consumption as well the ability to wake the processor from a peripheral without reconfiguring the peripheral for a change in clock, some of the communications modules have a Clock Control register at offset 0xFC8 in the module register space. The CS field in the Clock Control register allows the user to select the PIOSC or ALTCLK as the clock source for the module's baud clock. When the microcontroller enters Deep-Sleep mode, the PIOSC or ALTCLK becomes the source for the module clock as well, which allows the transmit and receive FIFOs to continue operation while the part is in Deep-Sleep. Figure 5-6 on page 253 shows how the clocks are selected.

**Figure 5-6. Module Clock Selection**




Additional power management modes are available that lower the power consumption of the peripheral memory, Flash, and SRAM memory. However, the lower power consumption modes have slower deep-sleep and wake-up times.

**Note:** If one or more wait states are configured for Run Mode, then when the device enters Deep-Sleep mode, it will achieve its lowest possible current. If there are no wait states applied in Run mode, then lowest possible current is not achieved.

### 5.2.6.4 Dynamic Power Management

In addition to the Sleep and Deep-Sleep modes and the clock gating for the on-chip modules, there are several additional power mode options that allow the LDO, Flash memory, and SRAM into different levels of power savings while in Sleep or Deep-Sleep modes. In addition, software has the ability to control the LDO settings to gain a power advantage when running at slower speeds. Note that these features may not be available on all devices; the **System Properties (SYSPROP)** register provides information on whether a mode is supported on a given MCU. The following registers provide these capabilities:

- **Peripheral Power Control (PCx):** Controls power to peripheral if that peripheral has the ability to respond to a power request.
- **Peripheral Memory Power Control (xMPC)**: Provides power control to some the peripheral memory arrays.
- **LDO Sleep Power Control (LDOSPCTL)**: Controls the LDO value in Sleep mode
- **LDO Deep-Sleep Power Control (LDODPCTL)**: Controls the LDO value in Deep-Sleep mode
- **LDO Sleep Power Calibration (LDOSPCAL)**: Provides factory recommendations for the LDO value in Sleep mode
- **LDO Deep-Sleep Power Calibration (LDODPCAL)**: Provides factory recommendations for the LDO value in Deep-Sleep mode
- **Sleep Power Configuration (SLPPWRCFG)**: Controls the power saving modes for Flash memory and SRAM in Sleep mode
- **Deep-Sleep Power Configuration (DSLPPWRCFG)**: Controls the power saving modes for Flash memory and SRAM in Deep-Sleep mode
- **Deep-Sleep Clock Configuration (DSCLKCFG)**: Controls the clocking in Deep-Sleep mode
- **Sleep / Deep-Sleep Power Mode Status (SDPMST)**: Provides status information on the various power saving events

***Peripheral Power Control***

The **Peripheral Power Control (PCx)** registers reside at offset 0x900 in the System Control module register space. For modules that reside in a separate power domain, the user has the capability to power down the module by setting the appropriate Pn bit to 0x0. This configuration provides the lowest power consumption state of the module. Currently the following registers can be programmed to disable power to the module:

- **PCCAN** register
- **PCLCD** register
- **PCEMAC** register
- **PCEPHY** register
- **PCUSB** register
- **PCCCM** register

Modification to other PCx registers have no effect, since they are not on their own power domain.

### *Peripheral Memory Power Control*

When Deep-Sleep is entered, users have the capability to reduce power further in peripheral modules which have their own associated memory array. Many of these peripherals can be programmed to enable a low-power retention mode or a power down of their associated peripheral SRAM array. If retention is supported and the PWRCTL bit field of the module's **xMPC** register is programmed to 0x1, the associated peripheral SRAM memory array is put in retention mode in which no accesses can be performed. When the PWRCTL bit is set to 0x0 in Deep-Sleep mode, the memory is powered off, the contents are lost and the SRAM is not accessible. The peripheral's **Power Domain Status (xPDS)** can be read to determine the status of the peripheral's memory array as well as the peripheral's current power domain status. The table below lists the capabilities of peripherals with SRAM arrays during low power modes.

**Table 5-8. Peripheral Memory Power Control**

| Module | Memory Retention Capability? | Memory Array Power Down Capability? |
|---|---|---|
| USB | Yes | Yes |
| EMAC | No | Yes<br>(only when power domain is off, **PCEMAC** register = 0x0) |
| LCD | No | No |
| CAN | No | Yes |

### *LDO Power Control*

**Note:** While the device is connected through JTAG, the LDO control settings for Sleep or Deep-Sleep are not available and will not be applied.

Software can configure the **LDOSPCTL** register (see page 312) and/or the **LDODPCTL** register (see page 315) to dynamically raise or lower the LDO voltage in Sleep and Deep-Sleep mode depending on whether an increase in performance or reduction in power consumption is required. The VLDO field in the **LDOSPCTL** register is set to 1.2 V as default. The **LDODPCTL** register is set to an LDO voltage of 0.9 V as default. If an application requires performance over power consumption in Deep-Sleep, the Deep-Sleep LDO voltage can be configured to a higher voltage than 0.9 V during System Control initialization by setting the VADJEN bit and programming the VLDO field of the **LDODPCTL** register.

Before the LDO level is lowered in Sleep or Deep-Sleep, the system clock must be configured to an acceptable frequency in the **RSCLKCFG** register for Sleep mode and in **DSLPCLKCFG** for Deep-Sleep mode. The following table shows the maximum System Clock and PIOSC frequency with respect to the LDO voltage.

**Table 5-9. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage**

| Operating Voltage (LDO) | Maximum System Clock Frequency | PIOSC |
|---|---|---|
| 1.2 | 120 MHz | 16 MHz |
| 0.9 | 30 MHz | 16 MHz |

The LDO Power Calibration registers, **LDOSPCAL** and **LDODPCAL**, provide suggested values for the LDO in the various modes. If software requests an LDO value that is too low or too high, the value is not accepted and an error is reported in the **SDPMST** register.

**Note:** When using the USB, Ethernet, EPI, and QSSI interfaces, the LDO must be configured to 1.2 V.

#### *Flash Memory and SRAM Power Control*

During Sleep or Deep-Sleep mode, Flash memory can be in either the default active mode or the low power mode; SRAM can be in the default active mode, standby mode, or low power mode. The active mode in each case provides the fastest times to sleep and wake up, but consumes more power. Low power mode provides the lowest power consumption, but takes longer to sleep and wake up.

The SRAM can be programmed to prohibit any power management by configuring the SRAMPM bit in the **Sleep Power Configuration (SLPPWRCFG)** register. This configuration operates in the same way that legacy Stellaris® devices operate and provides the fastest sleep and wake-up times, but consumes the most power while in Sleep and Deep-Sleep mode.

The following power saving options are available in Sleep and Deep-Sleep modes:

- The clocks can be gated according to the settings in the peripheral-specific **SCGC** or **DCGC** registers.
- In Deep-Sleep mode, the clock source can be changed and the PIOSC can be powered off (if no active peripheral requires it) using the **DSCLKCFG** register. These options are not available for Sleep mode.
- The LDO voltage can be changed using the **LDOSPCTL** or **LDODPCTL** register.
- The Flash memory can be put into low power mode.
- The SRAM can be put into standby or low power mode.

For typical power consumption and sleep/wake-up times, refer to “Current Consumption ” on page 2178 and “Sleep Modes” on page 2120.

The **SDPMST** register provides results on the Dynamic Power Management command issued. It also has some real time status that can be viewed by a debugger or the core if it is running. These events do not trigger an interrupt and are meant to provide information to help tune software for power management. The status register gets written at the beginning of every Dynamic Power Management event request that provides error checking. There is no mechanism to clear the bits; they are overwritten on the next event. The data is real time and there is no event to register that information.

### 5.2.6.5 Hibernation Mode

In this mode, the power supplies are turned off to the main part of the microcontroller and only the Hibernation module's circuitry is active. An external wake event or RTC event is required to bring the microcontroller back to Run mode. The Cortex-M4F processor and peripherals outside of the Hibernation module see a normal "power on" sequence and the processor starts running code. If the HIB module has been put in hibernation mode and a reset occurs, the reset handler should check the **HIB Raw Interrupt Status (HIBRIS)** register in the HIB module to determine the cause of the reset.

### 5.2.6.6 Hardware System Service Request

The **Hardware System Service Request (HSSR)** register is used to issue a request that returns a device to factory settings. An HSSR consists of writing the appropriate key and data structure address offset to the **HSSR** register in the System Control Module. Any HSSR initiates a reset event as the first event in the process. Then the **HSSR** register is evaluated.

To write to the **HSSR** register the KEY field must be set to 0xCA. The CDOFF field in the **HSSR** register can have one of the following three values:

- 0x00.0000 – No request and/or the previous request completed successfully
- 0xFF.FFFF – No request and the previous request failed
- Anything else – The offset into SRAM of a HSSR request structure

During the HSSR routine, if anything else is seen in the CDOFF field, then the offset is examined for validity and the structure it points to is examined for validity. If either is invalid, the request has failed and 0xFF.FFFF is written to the CDOFF field.

The offset is valid if all the following conditions are met:

- The CDOFF value is word aligned (that is, the two LSBs are both zero)
- The CDOFF value is at least 0x2000.4000
- The CDOFF value is at most 0x2003.FFF0

Once a valid HSSR offset is determined, the following structure is examined in the SRAM that is indicated by the CDOFF field in the **HSSR** register. In order to initiate a return-to-factory settings function, the data structure must be as follows:

- Request (32 bits) = 0xFEED.0001
- Data 1 (32 bits) = 0x0201.0100
- Data 2 (32 bits) = 0x0D08.0503
- Data 3 (32 bits) = 0x5937.2215

If the data bytes are correct, then the device is returned to factory condition. During the return-to-factory settings function, the following events occur:

- The RAM is erased in the Hibernation module
- The system SRAM is erased
- The **FMPPEn** registers are set to 0xFFFF.FFFF (to allow a Flash erase operation to occur)
- The EEPROM pages are erased
- A mass-erase of the flash array occurs
- The **BOOTCFG** register is written with 0xFFFF.FFFE

Once the return-to-factory settings sequence is completed, the CDOFF field of the **HSSR** register is written with 0x00.0000, indicating a successful completion and activating a system reset.

## 5.3 Initialization and Configuration

The PLL is configured using direct register writes to the **PLLFREQn**, **MEMTIM0**, and **PLLSTAT** registers. The steps for initializing the system clock from POR to use the PLL from the main oscillator is as follows:

1. Once POR has completed, the PIOSC is acting as the system clock.
2. Power up the MOSC by clearing the NOXTAL bit in the **MOSCCTL** register.
3. If single-ended MOSC mode is required, the MOSC is ready to use. If crystal mode is required, clear the PWRDN bit and wait for the MOSCPUPRIS bit to be set in the **Raw Interrupt Status (RIS)**, indicating MOSC crystal mode is ready.
4. Set the OSCSRC field to 0x3 in the **RSCLKCFG** register at offset 0x0B0.
5. If the application also requires the MOSC to be the deep-sleep clock source, then program the DSOSCSRC field in the **DSCLKCFG** register to 0x3.
6. Write the **PLLFREQ0** and **PLLFREQ1** registers with the values of Q, N, MINT, and MFRAC to the configure the desired VCO frequency setting.
7. Write the **MEMTIM0** register to correspond to the new system clock setting.
8. Wait for the **PLLSTAT** register to indicate the PLL has reached lock at the new operating point (or that a timeout period has passed and lock has failed, in which case an error condition exists and this sequence is abandoned and error processing is initiated).
9. Write the **RSCLKCFG** register's PSYSDIV value, set the USEPLL bit to enabled, and MEMTIMU bit.

If it is necessary to keep the MOSC powered on during automatic (deep-sleep) or accidental power down, then the MOSCDPD bit should be set to 0x1. Otherwise, if the MOSCDPD bit is set to 0x0, the MOSC is powered off when deep-sleep is entered or automatic power down occurs. The following table describes the relationship between the PWRDN bit in the **MOSCCTL** register and the MOSCDPD bit in the **DSCLKCFG** register:

**Table 5-10. MOSC Configurations**

| PWRDN bit | MOSCDPD field | Result |
|---|---|---|
| 0 | 0 | MOSC is powered ON in run and sleep modes, but is disabled in accidental power down, when the PWRDN bit is set in the **MOSCCTL** register, or in deep-sleep mode only if it is not the deep-sleep clock source (DSOSCSRC !==0x3). |
| 0 | 1 | MOSC is powered and running in run, sleep and deep-sleep modes. |
| 1 | 0 | MOSC is powered off, and does not run in any mode. Please note, that in this configuration, when the MOSC is disabled, the MOSC must not be chosen as a clock source or indeterminate results occur. |
| 1 | 1 | MOSC runs and does not disable itself in run, sleep, and deep-sleep modes regardless of the fact that the PWRDN bit is set. |

**Note:** The MOSCDPD bit has an effect in all modes of operation

To change the system clock frequency by changing its corresponding PSYSDIV or OSYSDIV value, a user must ensure timing parameters to memory are within range through the following steps:

1. If the change in system clock frequency changes the operational range of the timing parameters, the **MEMTIM0** register must be updated. If so, write the timing configuration register, **MEMTIM0**, setting the value to correspond to the final SYSCLK frequency ($f_{VCO}$/new SYSDIV or $f_{OSC}$). Otherwise the **MEMTIM0** register should not be changed.

2. Write the **RSCLKCFG** register's PSYSDIV value and MEMTIMU bit if the **MEMTIM0** register is updated in the first step. The new SYSDIV is now in effect.

## 5.4 Register Map

Table 5-11 on page 259 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

**Note:** Spaces in the System Control register space that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

Additional Flash and ROM registers defined in the System Control register space are described in the “Internal Memory” on page 632.

**Table 5-11. System Control Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **System Control Registers (System Control Offset)** | | | | | |
| 0x000 | DID0 | RO | - | Device Identification 0 | 267 |
| 0x004 | DID1 | RO | 0x1037.E076 | Device Identification 1 | 269 |
| 0x038 | PTBOCTL | RW | 0x0000.0003 | Power-Temp Brown Out Control | 271 |
| 0x050 | RIS | RO | 0x0000.0000 | Raw Interrupt Status | 273 |
| 0x054 | IMC | RW | 0x0000.0000 | Interrupt Mask Control | 275 |
| 0x058 | MISC | RW1C | 0x0000.0000 | Masked Interrupt Status and Clear | 277 |
| 0x05C | RESC | RW | 0x0000.0002 | Reset Cause | 279 |
| 0x060 | PWRTC | RW1C | 0x0000.0000 | Power-Temperature Cause | 282 |
| 0x064 | NMIC | RW | 0x0000.0000 | NMI Cause Register | 283 |
| 0x07C | MOSCCTL | RW | 0x0000.000C | Main Oscillator Control | 285 |
| 0x0B0 | RSCLKCFG | RW | 0x0000.0000 | Run and Sleep Mode Configuration Register | 287 |
| 0x0C0 | MEMTIM0 | RW | 0x0030.0030 | Memory Timing Parameter Register 0 for Main Flash and EEPROM | 289 |
| 0x138 | ALTCLKCFG | RW | 0x0000.0000 | Alternate Clock Configuration | 292 |
| 0x144 | DSCLKCFG | RW | 0x0000.0000 | Deep Sleep Clock Configuration Register | 293 |
| 0x148 | DIVSCLK | RW | 0x0000.0000 | Divisor and Source Clock Configuration | 296 |
| 0x14C | SYSPROP | RO | 0x0003.1F31 | System Properties | 298 |
| 0x150 | PIOSCCAL | RW | 0x0000.0000 | Precision Internal Oscillator Calibration | 301 |
| 0x154 | PIOSCSTAT | RO | 0x0040.0040 | Precision Internal Oscillator Statistics | 303 |
| 0x160 | PLLFREQ0 | RW | 0x0000.0000 | PLL Frequency 0 | 304 |
| 0x164 | PLLFREQ1 | RW | 0x0000.0000 | PLL Frequency 1 | 305 |
| 0x168 | PLLSTAT | RO | 0x0000.0000 | PLL Status | 306 |

**Table 5-11. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x188 | SLPPWRCFG | RW | 0x0000.0000 | Sleep Power Configuration | 307 |
| 0x18C | DSLPPWRCFG | RW | 0x0000.0000 | Deep-Sleep Power Configuration | 309 |
| 0x1A0 | NVMSTAT | RO | 0x0000.0001 | Non-Volatile Memory Information | 311 |
| 0x1B4 | LDOSPCTL | RW | 0x0000.0018 | LDO Sleep Power Control | 312 |
| 0x1B8 | LDOSPCAL | RO | 0x0000.1818 | LDO Sleep Power Calibration | 314 |
| 0x1BC | LDODPCTL | RW | 0x0000.0012 | LDO Deep-Sleep Power Control | 315 |
| 0x1C0 | LDODPCAL | RO | 0x0000.1212 | LDO Deep-Sleep Power Calibration | 317 |
| 0x1CC | SDPMST | RO | 0x0000.0000 | Sleep / Deep-Sleep Power Mode Status | 318 |
| 0x1D8 | RESBEHAVCTL | RW | 0xFFFF.FFFF | Reset Behavior Control Register | 321 |
| 0x1F4 | HSSR | RW | 0x0000.0000 | Hardware System Service Request | 323 |
| 0x280 | USBPDS | RO | 0x0000.003F | USB Power Domain Status | 324 |
| 0x284 | USBMPC | RW | 0x0000.0003 | USB Memory Power Control | 325 |
| 0x288 | EMACPDS | RO | 0x0000.003F | Ethernet MAC Power Domain Status | 326 |
| 0x28C | EMACMPC | RW | 0x0000.0003 | Ethernet MAC Memory Power Control | 327 |
| 0x290 | LCDPDS | RO | 0x0000.003F | LCD Power Domain Status | 328 |
| 0x294 | LCDMPC | RW | 0x0000.0003 | LCD Memory Power Control | 329 |
| 0x298 | CAN0PDS | RO | 0x0000.003F | CAN 0 Power Domain Status | 330 |
| 0x29C | CAN0MPC | RW | 0x0000.0003 | CAN 0 Memory Power Control | 331 |
| 0x2A0 | CAN1PDS | RO | 0x0000.003F | CAN 1 Power Domain Status | 332 |
| 0x2A4 | CAN1MPC | RW | 0x0000.0003 | CAN 1 Memory Power Control | 333 |
| 0x300 | PPWD | RO | 0x0000.0003 | Watchdog Timer Peripheral Present | 334 |
| 0x304 | PPTIMER | RO | 0x0000.00FF | 16/32-Bit General-Purpose Timer Peripheral Present | 335 |
| 0x308 | PPGPIO | RO | 0x0003.FFFF | General-Purpose Input/Output Peripheral Present | 337 |
| 0x30C | PPDMA | RO | 0x0000.0001 | Micro Direct Memory Access Peripheral Present | 340 |
| 0x310 | PPEPI | RO | 0x0000.0001 | EPI Peripheral Present | 341 |
| 0x314 | PPHIB | RO | 0x0000.0001 | Hibernation Peripheral Present | 342 |
| 0x318 | PPUART | RO | 0x0000.00FF | Universal Asynchronous Receiver/Transmitter Peripheral Present | 343 |
| 0x31C | PPSSI | RO | 0x0000.000F | Synchronous Serial Interface Peripheral Present | 345 |
| 0x320 | PPI2C | RO | 0x0000.03FF | Inter-Integrated Circuit Peripheral Present | 347 |
| 0x328 | PPUSB | RO | 0x0000.0001 | Universal Serial Bus Peripheral Present | 349 |
| 0x330 | PPEPHY | RO | 0x0000.0001 | Ethernet PHY Peripheral Present | 350 |

**Table 5-11. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x334 | PPCAN | RO | 0x0000.0003 | Controller Area Network Peripheral Present | 351 |
| 0x338 | PPADC | RO | 0x0000.0003 | Analog-to-Digital Converter Peripheral Present | 352 |
| 0x33C | PPACMP | RO | 0x0000.0001 | Analog Comparator Peripheral Present | 353 |
| 0x340 | PPPWM | RO | 0x0000.0001 | Pulse Width Modulator Peripheral Present | 354 |
| 0x344 | PPQEI | RO | 0x0000.0001 | Quadrature Encoder Interface Peripheral Present | 355 |
| 0x348 | PPLPC | RO | 0x0000.0000 | Low Pin Count Interface Peripheral Present | 356 |
| 0x350 | PPPECI | RO | 0x0000.0000 | Platform Environment Control Interface Peripheral Present | 357 |
| 0x354 | PPFAN | RO | 0x0000.0000 | Fan Control Peripheral Present | 358 |
| 0x358 | PPEEPROM | RO | 0x0000.0001 | EEPROM Peripheral Present | 359 |
| 0x35C | PPWTIMER | RO | 0x0000.0000 | 32/64-Bit Wide General-Purpose Timer Peripheral Present | 360 |
| 0x370 | PPRTS | RO | 0x0000.0000 | Remote Temperature Sensor Peripheral Present | 361 |
| 0x374 | PPCCM | RO | 0x0000.0001 | CRC and Cryptographic Modules Peripheral Present | 362 |
| 0x390 | PPLCD | RO | 0x0000.0001 | LCD Peripheral Present | 363 |
| 0x398 | PPOWIRE | RO | 0x0000.0001 | 1-Wire Peripheral Present | 364 |
| 0x39C | PPEMAC | RO | 0x0000.0001 | Ethernet MAC Peripheral Present | 365 |
| 0x3A0 | PPPRB | RO | 0x0000.0000 | Power Regulator Bus Peripheral Present | 366 |
| 0x3A4 | PPHIM | RO | 0x0000.0000 | Human Interface Master Peripheral Present | 367 |
| 0x500 | SRWD | RW | 0x0000.0000 | Watchdog Timer Software Reset | 368 |
| 0x504 | SRTIMER | RW | 0x0000.0000 | 16/32-Bit General-Purpose Timer Software Reset | 369 |
| 0x508 | SRGPIO | RW | 0x0000.0000 | General-Purpose Input/Output Software Reset | 371 |
| 0x50C | SRDMA | RW | 0x0000.0000 | Micro Direct Memory Access Software Reset | 375 |
| 0x510 | SREPI | RW | 0x0000.0000 | EPI Software Reset | 376 |
| 0x514 | SRHIB | RW | 0x0000.0000 | Hibernation Software Reset | 377 |
| 0x518 | SRUART | RW | 0x0000.0000 | Universal Asynchronous Receiver/Transmitter Software Reset | 378 |
| 0x51C | SRSSI | RW | 0x0000.0000 | Synchronous Serial Interface Software Reset | 380 |
| 0x520 | SRI2C | RW | 0x0000.0000 | Inter-Integrated Circuit Software Reset | 382 |
| 0x528 | SRUSB | RW | 0x0000.0000 | Universal Serial Bus Software Reset | 384 |
| 0x530 | SREPHY | RW | 0x0000.0000 | Ethernet PHY Software Reset | 385 |
| 0x534 | SRCAN | RW | 0x0000.0000 | Controller Area Network Software Reset | 386 |
| 0x538 | SRADC | RW | 0x0000.0000 | Analog-to-Digital Converter Software Reset | 387 |

**Table 5-11. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x53C | SRACMP | RW | 0x0000.0000 | Analog Comparator Software Reset | 388 |
| 0x540 | SRPWM | RW | 0x0000.0000 | Pulse Width Modulator Software Reset | 389 |
| 0x544 | SRQEI | RW | 0x0000.0000 | Quadrature Encoder Interface Software Reset | 390 |
| 0x558 | SREEPROM | RW | 0x0000.0000 | EEPROM Software Reset | 391 |
| 0x574 | SRCCM | RW | 0x0000.0000 | CRC and Cryptographic Modules Software Reset | 392 |
| 0x590 | SRLCD | RW | 0x0000.0000 | LCD Controller Software Reset | 393 |
| 0x598 | SROWIRE | RW | 0x0000.0000 | 1-Wire Software Reset | 394 |
| 0x59C | SREMAC | RW | 0x0000.0000 | Ethernet MAC Software Reset | 395 |
| 0x600 | RCGCWD | RW | 0x0000.0000 | Watchdog Timer Run Mode Clock Gating Control | 396 |
| 0x604 | RCGCTIMER | RW | 0x0000.0000 | 16/32-Bit General-Purpose Timer Run Mode Clock Gating Control | 397 |
| 0x608 | RCGCGPIO | RW | 0x0000.0000 | General-Purpose Input/Output Run Mode Clock Gating Control | 399 |
| 0x60C | RCGCDMA | RW | 0x0000.0000 | Micro Direct Memory Access Run Mode Clock Gating Control | 402 |
| 0x610 | RCGCEPI | RW | 0x0000.0000 | EPI Run Mode Clock Gating Control | 403 |
| 0x614 | RCGCHIB | RW | 0x0000.0001 | Hibernation Run Mode Clock Gating Control | 404 |
| 0x618 | RCGCUART | RW | 0x0000.0000 | Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control | 405 |
| 0x61C | RCGCSSI | RW | 0x0000.0000 | Synchronous Serial Interface Run Mode Clock Gating Control | 407 |
| 0x620 | RCGCI2C | RW | 0x0000.0000 | Inter-Integrated Circuit Run Mode Clock Gating Control | 408 |
| 0x628 | RCGCUSB | RW | 0x0000.0000 | Universal Serial Bus Run Mode Clock Gating Control | 410 |
| 0x630 | RCGCEPHY | RW | 0x0000.0000 | Ethernet PHY Run Mode Clock Gating Control | 411 |
| 0x634 | RCGCCAN | RW | 0x0000.0000 | Controller Area Network Run Mode Clock Gating Control | 412 |
| 0x638 | RCGCADC | RW | 0x0000.0000 | Analog-to-Digital Converter Run Mode Clock Gating Control | 413 |
| 0x63C | RCGCACMP | RW | 0x0000.0000 | Analog Comparator Run Mode Clock Gating Control | 414 |
| 0x640 | RCGCPWM | RW | 0x0000.0000 | Pulse Width Modulator Run Mode Clock Gating Control | 415 |
| 0x644 | RCGCQEI | RW | 0x0000.0000 | Quadrature Encoder Interface Run Mode Clock Gating Control | 416 |
| 0x658 | RCGCEEPROM | RW | 0x0000.0000 | EEPROM Run Mode Clock Gating Control | 417 |
| 0x674 | RCGCCCM | RW | 0x0000.0000 | CRC and Cryptographic Modules Run Mode Clock Gating Control | 418 |
| 0x690 | RCGCLCD | RW | 0x0000.0000 | LCD Controller Run Mode Clock Gating Control | 419 |
| 0x698 | RCGCOWIRE | RW | 0x0000.0000 | 1-Wire Run Mode Clock Gating Control | 420 |

**Table 5-11. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x69C | RCGCEMAC | RW | 0x0000.0000 | Ethernet MAC Run Mode Clock Gating Control | 421 |
| 0x700 | SCGCWD | RW | 0x0000.0000 | Watchdog Timer Sleep Mode Clock Gating Control | 422 |
| 0x704 | SCGCTIMER | RW | 0x0000.0000 | 16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control | 423 |
| 0x708 | SCGCGPIO | RW | 0x0000.0000 | General-Purpose Input/Output Sleep Mode Clock Gating Control | 425 |
| 0x70C | SCGCDMA | RW | 0x0000.0000 | Micro Direct Memory Access Sleep Mode Clock Gating Control | 428 |
| 0x710 | SCGCEPI | RW | 0x0000.0000 | EPI Sleep Mode Clock Gating Control | 429 |
| 0x714 | SCGCHIB | RW | 0x0000.0001 | Hibernation Sleep Mode Clock Gating Control | 430 |
| 0x718 | SCGCUART | RW | 0x0000.0000 | Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control | 431 |
| 0x71C | SCGCSSI | RW | 0x0000.0000 | Synchronous Serial Interface Sleep Mode Clock Gating Control | 433 |
| 0x720 | SCGCI2C | RW | 0x0000.0000 | Inter-Integrated Circuit Sleep Mode Clock Gating Control | 434 |
| 0x728 | SCGCUSB | RW | 0x0000.0000 | Universal Serial Bus Sleep Mode Clock Gating Control | 436 |
| 0x730 | SCGCEPHY | RW | 0x0000.0000 | Ethernet PHY Sleep Mode Clock Gating Control | 437 |
| 0x734 | SCGCCAN | RW | 0x0000.0000 | Controller Area Network Sleep Mode Clock Gating Control | 438 |
| 0x738 | SCGCADC | RW | 0x0000.0000 | Analog-to-Digital Converter Sleep Mode Clock Gating Control | 439 |
| 0x73C | SCGCACMP | RW | 0x0000.0000 | Analog Comparator Sleep Mode Clock Gating Control | 440 |
| 0x740 | SCGCPWM | RW | 0x0000.0000 | Pulse Width Modulator Sleep Mode Clock Gating Control | 441 |
| 0x744 | SCGCQEI | RW | 0x0000.0000 | Quadrature Encoder Interface Sleep Mode Clock Gating Control | 442 |
| 0x758 | SCGCEEPROM | RW | 0x0000.0000 | EEPROM Sleep Mode Clock Gating Control | 443 |
| 0x774 | SCGCCCM | RW | 0x0000.0000 | CRC and Cryptographic Modules Sleep Mode Clock Gating Control | 444 |
| 0x790 | SCGCLCD | RW | 0x0000.0000 | LCD Controller Sleep Mode Clock Gating Control | 445 |
| 0x798 | SCGCOWIRE | RW | 0x0000.0000 | 1-Wire Sleep Mode Clock Gating Control | 446 |
| 0x79C | SCGCEMAC | RW | 0x0000.0000 | Ethernet MAC Sleep Mode Clock Gating Control | 447 |
| 0x800 | DCGCWD | RW | 0x0000.0000 | Watchdog Timer Deep-Sleep Mode Clock Gating Control | 448 |
| 0x804 | DCGCTIMER | RW | 0x0000.0000 | 16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control | 449 |
| 0x808 | DCGCGPIO | RW | 0x0000.0000 | General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control | 451 |

**Table 5-11. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x80C | DCGCDMA | RW | 0x0000.0000 | Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control | 454 |
| 0x810 | DCGCEPI | RW | 0x0000.0000 | EPI Deep-Sleep Mode Clock Gating Control | 455 |
| 0x814 | DCGCHIB | RW | 0x0000.0001 | Hibernation Deep-Sleep Mode Clock Gating Control | 456 |
| 0x818 | DCGCUART | RW | 0x0000.0000 | Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control | 457 |
| 0x81C | DCGCSSI | RW | 0x0000.0000 | Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control | 459 |
| 0x820 | DCGCI2C | RW | 0x0000.0000 | Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control | 460 |
| 0x828 | DCGCUSB | RW | 0x0000.0000 | Universal Serial Bus Deep-Sleep Mode Clock Gating Control | 462 |
| 0x830 | DCGCEPHY | RW | 0x0000.0000 | Ethernet PHY Deep-Sleep Mode Clock Gating Control | 463 |
| 0x834 | DCGCCAN | RW | 0x0000.0000 | Controller Area Network Deep-Sleep Mode Clock Gating Control | 464 |
| 0x838 | DCGCADC | RW | 0x0000.0000 | Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control | 465 |
| 0x83C | DCGCACMP | RW | 0x0000.0000 | Analog Comparator Deep-Sleep Mode Clock Gating Control | 466 |
| 0x840 | DCGCPWM | RW | 0x0000.0000 | Pulse Width Modulator Deep-Sleep Mode Clock Gating Control | 467 |
| 0x844 | DCGCQEI | RW | 0x0000.0000 | Quadrature Encoder Interface Deep-Sleep Mode Clock Gating Control | 468 |
| 0x858 | DCGCEEPROM | RW | 0x0000.0000 | EEPROM Deep-Sleep Mode Clock Gating Control | 469 |
| 0x874 | DCGCCCM | RW | 0x0000.0000 | CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control | 470 |
| 0x890 | DCGCLCD | RW | 0x0000.0000 | LCD Controller Deep-Sleep Mode Clock Gating Control | 471 |
| 0x898 | DCGCOWIRE | RW | 0x0000.0000 | 1-Wire Deep-Sleep Mode Clock Gating Control | 472 |
| 0x89C | DCGCEMAC | RW | 0x0000.0000 | Ethernet MAC Deep-Sleep Mode Clock Gating Control | 473 |
| 0x900 | PCWD | RW | 0x0000.0003 | Watchdog Timer Power Control | 474 |
| 0x904 | PCTIMER | RW | 0x0000.00FF | 16/32-Bit General-Purpose Timer Power Control | 476 |
| 0x908 | PCGPIO | RW | 0x0003.FFFF | General-Purpose Input/Output Power Control | 479 |
| 0x90C | PCDMA | RW | 0x0000.0001 | Micro Direct Memory Access Power Control | 485 |
| 0x910 | PCEPI | RW | 0x0000.0001 | External Peripheral Interface Power Control | 487 |
| 0x914 | PCHIB | RW | 0x0000.0001 | Hibernation Power Control | 489 |
| 0x918 | PCUART | RW | 0x0000.00FF | Universal Asynchronous Receiver/Transmitter Power Control | 491 |

**Table 5-11. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x91C | PCSSI | RW | 0x0000.000F | Synchronous Serial Interface Power Control | 494 |
| 0x920 | PCI2C | RW | 0x0000.03FF | Inter-Integrated Circuit Power Control | 496 |
| 0x928 | PCUSB | RW | 0x0000.0001 | Universal Serial Bus Power Control | 500 |
| 0x930 | PCEPHY | RW | 0x0000.0000 | Ethernet PHY Power Control | 502 |
| 0x934 | PCCAN | RW | 0x0000.0003 | Controller Area Network Power Control | 504 |
| 0x938 | PCADC | RW | 0x0000.0003 | Analog-to-Digital Converter Power Control | 506 |
| 0x93C | PCACMP | RW | 0x0000.0001 | Analog Comparator Power Control | 508 |
| 0x940 | PCPWM | RW | 0x0000.0001 | Pulse Width Modulator Power Control | 510 |
| 0x944 | PCQEI | RW | 0x0000.0001 | Quadrature Encoder Interface Power Control | 512 |
| 0x958 | PCEEPROM | RW | 0x0000.0001 | EEPROM Power Control | 514 |
| 0x974 | PCCCM | RW | 0x0000.0001 | CRC and Cryptographic Modules Power Control | 516 |
| 0x990 | PCLCD | RW | 0x0000.0001 | LCD Controller Power Control | 518 |
| 0x998 | PCOWIRE | RW | 0x0000.0001 | 1-Wire Power Control | 520 |
| 0x99C | PCEMAC | RW | 0x0000.0001 | Ethernet MAC Power Control | 522 |
| 0xA00 | PRWD | RO | 0x0000.0000 | Watchdog Timer Peripheral Ready | 524 |
| 0xA04 | PRTIMER | RO | 0x0000.0000 | 16/32-Bit General-Purpose Timer Peripheral Ready | 525 |
| 0xA08 | PRGPIO | RO | 0x0000.0000 | General-Purpose Input/Output Peripheral Ready | 527 |
| 0xA0C | PRDMA | RO | 0x0000.0000 | Micro Direct Memory Access Peripheral Ready | 531 |
| 0xA10 | PREPI | RO | 0x0000.0000 | EPI Peripheral Ready | 532 |
| 0xA14 | PRHIB | RO | 0x0000.0001 | Hibernation Peripheral Ready | 533 |
| 0xA18 | PRUART | RO | 0x0000.0000 | Universal Asynchronous Receiver/Transmitter Peripheral Ready | 534 |
| 0xA1C | PRSSI | RO | 0x0000.0000 | Synchronous Serial Interface Peripheral Ready | 536 |
| 0xA20 | PRI2C | RO | 0x0000.0000 | Inter-Integrated Circuit Peripheral Ready | 538 |
| 0xA28 | PRUSB | RO | 0x0000.0000 | Universal Serial Bus Peripheral Ready | 541 |
| 0xA30 | PREPHY | RO | 0x0000.0000 | Ethernet PHY Peripheral Ready | 542 |
| 0xA34 | PRCAN | RO | 0x0000.0000 | Controller Area Network Peripheral Ready | 543 |
| 0xA38 | PRADC | RO | 0x0000.0000 | Analog-to-Digital Converter Peripheral Ready | 544 |
| 0xA3C | PRACMP | RO | 0x0000.0000 | Analog Comparator Peripheral Ready | 545 |
| 0xA40 | PRPWM | RO | 0x0000.0000 | Pulse Width Modulator Peripheral Ready | 546 |
| 0xA44 | PRQEI | RO | 0x0000.0000 | Quadrature Encoder Interface Peripheral Ready | 547 |
| 0xA58 | PREEPROM | RO | 0x0000.0000 | EEPROM Peripheral Ready | 548 |

**Table 5-11. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xA74 | PRCCM | RO | 0x0000.0000 | CRC and Cryptographic Modules Peripheral Ready | 549 |
| 0xA90 | PRLCD | RO | 0x0000.0000 | LCD Controller Peripheral Ready | 550 |
| 0xA98 | PROWIRE | RO | 0x0000.0000 | 1-Wire Peripheral Ready | 551 |
| 0xA9C | PREMAC | RO | 0x0000.0000 | Ethernet MAC Peripheral Ready | 552 |
| 0xF20 | UNIQUEID0 | RO | - | Unique ID 0 | 553 |
| 0xF24 | UNIQUEID1 | RO | - | Unique ID 1 | 553 |
| 0xF28 | UNIQUEID2 | RO | - | Unique ID 2 | 553 |
| 0xF2C | UNIQUEID3 | RO | - | Unique ID 3 | 553 |
| **CCM System Control Registers (CCM Control Offset)** | | | | | |
| 0x204 | CCMCGREQ | RW | 0x0000.0000 | Cryptographic Modules Clock Gating Request | 554 |

## 5.5 System Control Register Descriptions (System Control Offset)

All addresses given are relative to the System Control base address of 0x400F.E000.

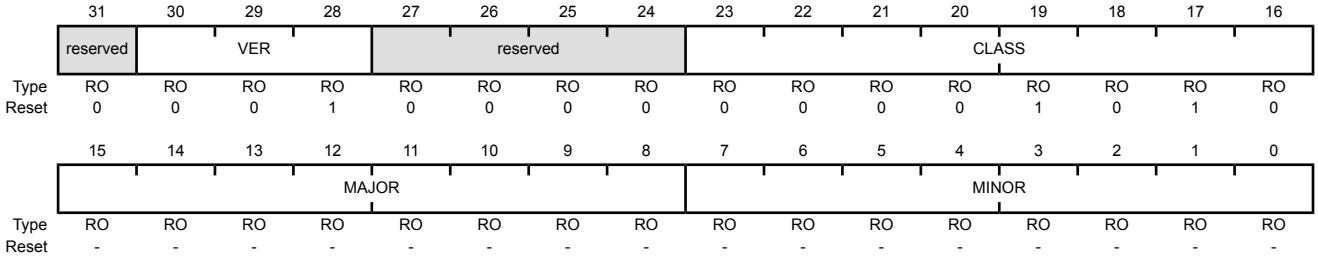
## Register 1: Device Identification 0 (DID0), offset 0x000

This register identifies the version of the microcontroller. Each microcontroller is uniquely identified by the combined values of the CLASS field in the **DID0** register and the PARTNO field in the **DID1** register. The MAJOR and MINOR bit fields indicate the die revision number. Combined, the MAJOR and MINOR bit fields indicate the part revision number.

| MAJOR Bitfield Value | MINOR Bitfield Value | Die Revision | Part Revision |
|---|---|---|---|
| 0x0 | 0x0 | A0 | 1 |
| 0x0 | 0x1 | A1 | 2 |
| 0x0 | 0x2 | A2 | 3 |

Device Identification 0 (DID0)

Base 0x400F.E000
Offset 0x000
Type RO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | reserved | RO | 0 | 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. |
| 30:28 | VER | RO | 0x1 | DID0 Version<br>This field defines the **DID0** register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x1 Second version of the **DID0** register format. |
| 27:24 | reserved | RO | 0 | 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. |
| 23:16 | CLASS | RO | 0x0A | Device Class<br>The CLASS field value identifies the internal design from which all mask sets are generated for all microcontrollers in a particular product line. The CLASS field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the MAJOR or MINOR fields require differentiation from prior microcontrollers. The value of the CLASS field is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x0A Tiva™ Snowflake-class microcontrollers |

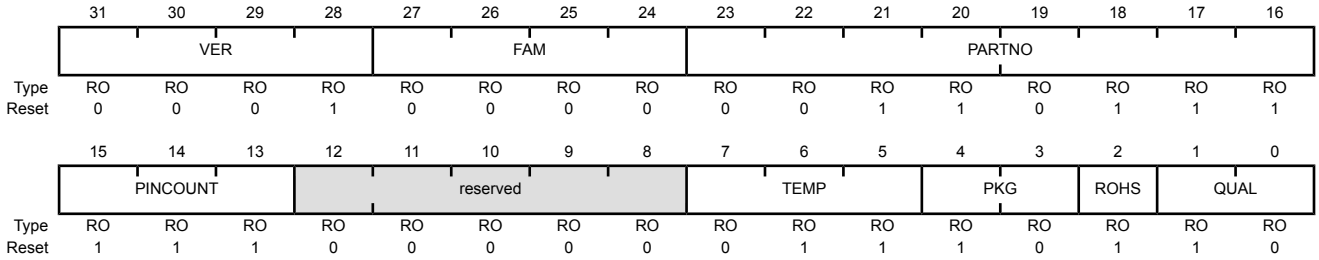
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:8 | MAJOR | RO | - | Major Revision<br>This field specifies the major revision number of the microcontroller. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, and so on). This field is encoded as follows:<br><br>Value Description<br>0x0 Revision A (initial device)<br>0x1 Revision B (first base layer revision)<br>0x2 Revision C (second base layer revision)<br><br>and so on. |
| 7:0 | MINOR | RO | - | Minor Revision<br>This field specifies the minor revision number of the microcontroller. The minor revision reflects changes to the metal layers of the design. The MINOR field value is reset when the MAJOR field is changed. This field is numeric and is encoded as follows:<br><br>Value Description<br>0x0 Initial device, or a major revision update.<br>0x1 First metal layer change.<br>0x2 Second metal layer change.<br><br>and so on. |

## Register 2: Device Identification 1 (DID1), offset 0x004

This register identifies the device family, part number, temperature range, pin count, and package type. Each microcontroller is uniquely identified by the combined values of the CLASS field in the **DID0** register and the PARTNO field in the **DID1** register.

Device Identification 1 (DID1)

Base 0x400F.E000
Offset 0x004
Type RO, reset 0x1037.E076




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | VER | RO | 0x1 | DID1 Version<br>This field defines the **DID1** register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved):<br>Value Description<br>0x0 Initial **DID1** register format definition, indicating a Stellaris LM3Snnn device.<br>0x1 Second version of the **DID1** register format. |
| 27:24 | FAM | RO | 0x0 | Family<br>This field provides the family identification of the device within the Tiva™ product portfolio. The value is encoded as follows (all other encodings are reserved):<br>Value Description<br>0x0 Tiva™ C Series microcontrollers and legacy Stellaris microcontrollers, that is, all devices with external part numbers starting with TM4Cor LM3S. |
| 23:16 | PARTNO | RO | 0x37 | Part Number<br>This field provides the part number of the device within the family. This value indicates the TM4C129XKCZAD microcontroller. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:13 | PINCOUNT | RO | 0x7 | Package Pin Count<br>This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x0 reserved<br>0x1 reserved<br>0x2 100-pin LQFP package<br>0x3 64-pin LQFP package<br>0x4 144-pin LQFP package<br>0x5 157-pin BGA package<br>0x6 128-pin TQFP package<br>0x7 212-pin BGA package |
| 12:8 | reserved | RO | 0x0 | 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. |
| 7:5 | TEMP | RO | 0x3 | Temperature Range<br>This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x0 Commercial temperature range<br>0x1 Industrial temperature range<br>0x2 Extended temperature range |
| 4:3 | PKG | RO | 0x2 | Package Type<br>This field specifies the package type. The value is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x0 reserved<br>0x1 QFP package<br>0x2 BGA package |
| 2 | ROHS | RO | 0x1 | RoHS-Compliance<br>This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant. |
| 1:0 | QUAL | RO | 0x2 | Qualification Status<br>This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x0 Engineering Sample (unqualified)<br>0x1 Pilot Production (unqualified)<br>0x2 Fully Qualified |

## Register 3: Power-Temp Brown Out Control (PTBOCTL), offset 0x038

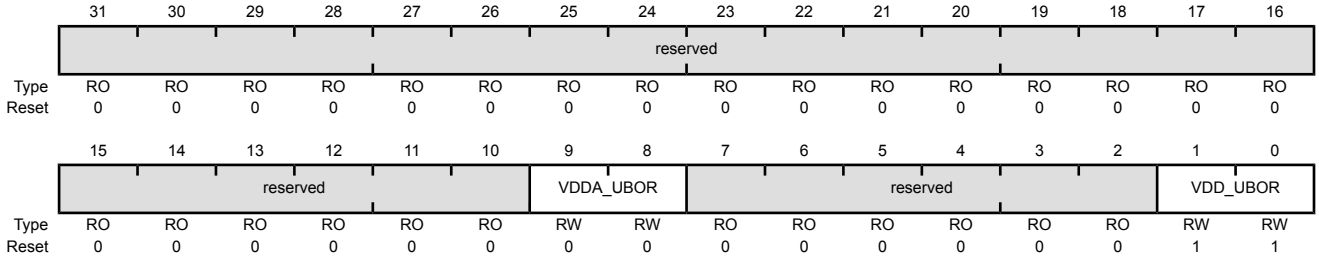
This register determines, based on an individual event level, the appropriate next level of action (for example, NONE, System Control Interrupt, NMI, or reset) when an event occurs.

Power-temperature event actions are directed to the core as a System Control Interrupt or NMI. When a reset occurs, its behavior is controlled by the **Reset Behavior Control (RESBEHAVCTL)** register. If one of the events configured in the **PTBOCTL** register causes a reset, it is registered as a B0R interrupt in the **Reset Cause (RESC)** register.

**Note:** $V_{DDA}$ is the supply voltage to the analog components of the device and $V_{DD}$ is the supply voltage to the digital components of the device.

Power-Temp Brown Out Control (PTBOCTL)

Base 0x400F.E000
Offset 0x038
Type RW, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |
| 9:8 | VDDA_UBOR | RW | 0 | $V_{DDA}$ under BOR Event Action<br>An event occurs when $V_{DDA}$ trips under the $V_{DDA_BOR0}$ threshold found in Table 32-13 on page 2103.<br>This field determines the action to take on the event.<br>Value Description<br>0x0 No Action<br>0x1 System control interrupt<br>0x2 NMI<br>0x3 Reset |
| 7:2 | reserved | RO | 0 | 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. |

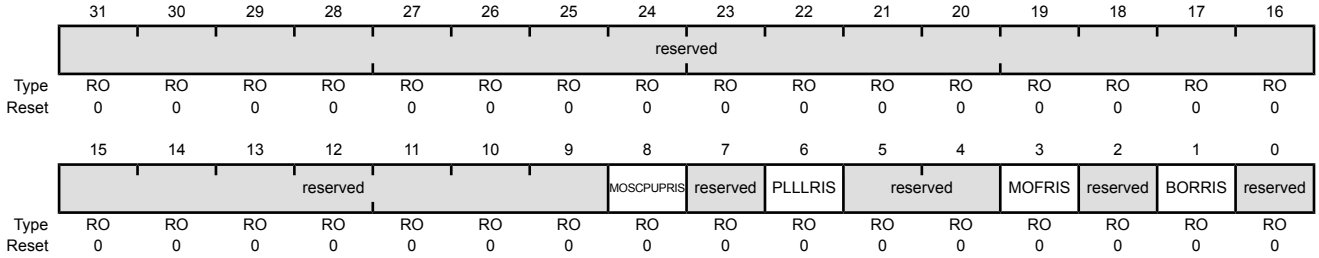
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | VDD_UBOR | RW | 0x3 | $V_{DD}$ under BOR Event Action<br>An event occurs when $V_{DD}$ trips under the $V_{DD_BOR}$ threshold found in Table 32-13 on page 2103.<br>This field determines the action to take on the event.<br><br>Value Description<br>0x0 No Action<br>0x1 System control interrupt<br>0x2 NMI<br>0x3 Reset |

## Register 4: Raw Interrupt Status (RIS), offset 0x050

This register indicates the status for system control raw interrupts. An interrupt is sent to the interrupt controller if the corresponding bit in the **Interrupt Mask Control (IMC)** register is set. Writing a 1 to the corresponding bit in the **Masked Interrupt Status and Clear (MISC)** register clears an interrupt status bit.

Raw Interrupt Status (RIS)

Base 0x400F.E000
Offset 0x050
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | 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. |
| 8 | MOSCPUPRIS | RO | 0 | MOSC Power Up Raw Interrupt Status<br><br>Value Description<br>0 Sufficient time has not passed for the MOSC to reach the expected frequency.<br>1 Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by $T_{MOSC_START}$.<br><br>This bit is cleared by writing a 1 to the `MOSCPUPMIS` bit in the **MISC** register. |
| 7 | reserved | RO | 0 | 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. |
| 6 | PLLLRIS | RO | 0 | PLL Lock Raw Interrupt Status<br><br>Value Description<br>0 The PLL timer has not reached $T_{READY}$.<br>1 The PLL timer has reached $T_{READY}$ indicating that sufficient time has passed for the PLL to lock.<br><br>This bit is cleared by writing a 1 to the `PLLLMIS` bit in the **MISC** register. |
| 5:4 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MOFRIS | RO | 0 | Main Oscillator Failure Raw Interrupt Status<br><br>Value Description<br>0 The main oscillator has not failed.<br>1 The MOSCIM bit in the **MOSCCTL** register is set and the main oscillator has failed.<br><br>This bit is cleared by writing a 1 to the MOFMIS bit in the **MISC** register. |
| 2 | reserved | RO | 0 | 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. |
| 1 | BORRIS | RO | 0 | Brown-Out Reset Raw Interrupt Status<br><br>Value Description<br>0 A brown-out condition is not currently active.<br>1 A brown-out condition is currently active.<br><br>The appropriate BOR bit in the **PTBOCTL** register must be set to an interrupt (0x1) encoding in order to generate an interrupt.<br>This bit is cleared by writing a 1 to the BORMIS bit in the **MISC** register. |
| 0 | reserved | RO | 0 | 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. |

## Register 5: Interrupt Mask Control (IMC), offset 0x054

This register contains the mask bits for system control raw interrupts. A raw interrupt, indicated by a bit being set in the **Raw Interrupt Status (RIS)** register, is sent to the interrupt controller if the corresponding bit in this register is set.

Interrupt Mask Control (IMC)

Base 0x400F.E000
Offset 0x054
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | MOSCPUPIM | reserved | PLLLIM | reserved | | MOFIM | reserved | BORIM | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RW | RO | RW | RO | RO | RW | RO | RW | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | 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. |
| 8 | MOSCPUPIM | RW | 0 | MOSC Power Up Interrupt Mask<br>Value Description<br>0 The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the **RIS** register is set. |
| 7 | reserved | RO | 0 | 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. |
| 6 | PLLLIM | RW | 0 | PLL Lock Interrupt Mask<br>Value Description<br>0 The PLLLRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PLLLRIS bit in the **RIS** register is set. |
| 5:4 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MOFIM | RW | 0 | Main Oscillator Failure Interrupt Mask<br><br>Value Description<br>0 The MOFRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the MOFRIS bit in the **RIS** register is set. |
| 2 | reserved | RO | 0 | 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. |
| 1 | BORIM | RW | 0 | Brown-Out Reset Interrupt Mask<br><br>Value Description<br>0 The BORRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the BORRIS bit in the **RIS** register is set. |
| 0 | reserved | RO | 0 | 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. |

## Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058

On a read, this register gives the current masked status value of the corresponding interrupt in the **Raw Interrupt Status (RIS)** register. All of the bits are RW1C, thus writing a 1 to a bit clears the corresponding raw interrupt bit in the **RIS** register (see page 273).

Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000
Offset 0x058
Type RW1C, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | MOSCPUPMIS | reserved | PLLLMIS | reserved | | MOFMIS | reserved | BORMIS | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RW1C | RO | RW1C | RO | RO | RW1C | RO | RW1C | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | 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. |
| 8 | MOSCPUPMIS | RW1C | 0 | MOSC Power Up Masked Interrupt Status<br>Value Description<br>0 When read, a 0 indicates that sufficient time has not passed for the MOSC PLL to lock.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the MOSC PLL to lock.<br>Writing a 1 to this bit clears it and also the `MOSCPUPRIS` bit in the **RIS** register. |
| 7 | reserved | RO | 0 | 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. |
| 6 | PLLLMIS | RW1C | 0 | PLL Lock Masked Interrupt Status<br>Value Description<br>0 When read, a 0 indicates that sufficient time has not passed for the PLL to lock.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the PLL to lock.<br>Writing a 1 to this bit clears it and also the `PLLLRIS` bit in the **RIS** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | reserved | RO | 0x0 | 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. |
| 3 | MOFMIS | RW1C | 0 | Main Oscillator Failure Masked Interrupt Status<br><br>Value Description<br>0 When read, a 0 indicates that the main oscillator has not failed.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled because the main oscillator failed.<br>Writing a 1 to this bit clears it and also the `MOFRIS` bit in the **RIS** register. |
| 2 | reserved | RO | 0 | 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. |
| 1 | BORMIS | RW1C | 0 | BOR Masked Interrupt Status<br><br>Value Description<br>0 When read, a 0 indicates that a brown-out condition has not occurred.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled because of a brown-out condition.<br>Writing a 1 to this bit clears it and also the `BORRIS` bit in the **RIS** register. |
| 0 | reserved | RO | 0 | 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. |

## Register 7: Reset Cause (RESC), offset 0x05C

This register is set with the reset cause after reset. The bits in this register are sticky and maintain their state across multiple reset sequences. If a full POK-POR is initiated, the POR bit in the **RESC** register is set and all other bits are cleared. If the WDOGn, BOR or EXTRES configuration fields are set to 0x3 in the **RESBEHAVCTL** register and a simulated POR is initiated, the cause of the reset is reflected in the **RESC** register.

**Note:** After the **Reset Cause (RESC)** register is read, the **Hibernate Raw Interrupt Status (HIBRIS)** register in the Hibernation module must be evaluated to determine the full cause of the reset. Although an external reset assertion or POR resulting from a wake event is registered in the **RESC** register, the specific external wake source, including a low battery detect, is only registered in the **HIBRIS** register.

Reset Cause (RESC)

Base 0x400F.E000
Offset 0x05C
Type RW, reset 0x0000.0002

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | MOSCFAIL |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | HSSR | reserved | | | | | | WDT1 | SW | WDT0 | BOR | POR | EXT |
| Type | RO | RO | RO | RW | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0x000 | 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. |
| 16 | MOSCFAIL | RW | - | MOSC Failure Reset<br>Writing a 0 to this bit clears it.<br><br>Value Description<br>0 When read, this bit indicates that a MOSC failure has not generated a reset since the previous power-on reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that the MOSC circuit was enabled for clock validation and failed while the MOSCIM bit in the **MOSCCTL** register is clear, generating a reset event. |
| 15:13 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | HSSR | RW | - | HSSR Reset<br>Value Description<br>0 When read, this bit indicates that a HSSR request has not generated a reset since the previous power-on reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that a HSSR request has generated a reset. |
| 11:6 | reserved | RO | 0 | 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 | WDT1 | RW | 0 | Watchdog Timer 1 Reset<br>Value Description<br>0 When read, this bit indicates that Watchdog Timer 1 has not generated a reset since the previous power-on reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that Watchdog Timer 1 timed out and generated a reset. |
| 4 | SW | RW | 0 | Software Reset<br>Value Description<br>0 When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that a software reset has caused a reset event. |
| 3 | WDT0 | RW | 0 | Watchdog Timer 0 Reset<br>Value Description<br>0 When read, this bit indicates that Watchdog Timer 0 has not generated a reset since the previous power-on reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset. |

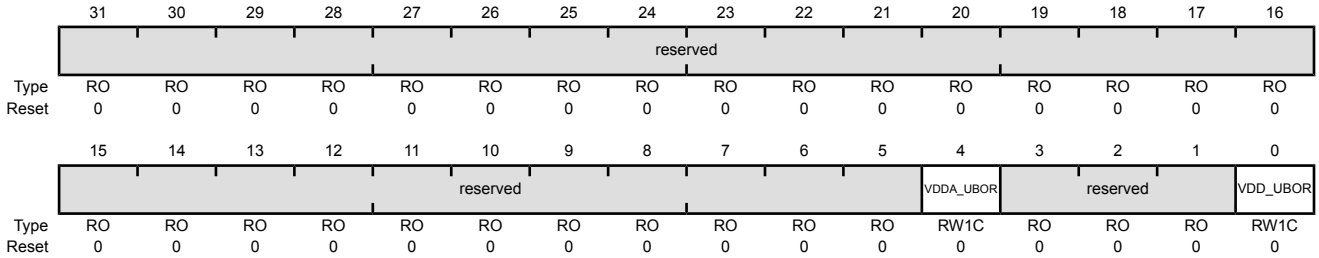
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | BOR | RW | 0 | Brown-Out Reset<br>Note that for this bit, the BOR event that causes the Brown-Out Reset can be any of the following:<br>■ The $V_{DD}$ supply drops below its acceptable operating range.<br>■ The $V_{DDA}$ supply drops below its acceptable operating range.<br>Value Description<br>0 When read, this bit indicates that a brown-out reset has not generated a reset since the previous power-on reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that a brown-out reset has caused a reset event. |
| 1 | POR | RW | 1 | Power-On Reset<br>Value Description<br>0 When read, this bit indicates that a power-on reset has not generated a reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that a power-on reset has caused a reset event. |
| 0 | EXT | RW | 0 | External Reset<br>Value Description<br>0 When read, this bit indicates that an external reset ($\overline{RST}$ assertion) has not caused a reset event since the previous power-on reset.<br>Writing a 0 to this bit clears it.<br>1 When read, this bit indicates that an external reset ($\overline{RST}$ assertion) has caused a reset event. |

## Register 8: Power-Temperature Cause (PWRTC), offset 0x060

This register provides detailed information on the power subsystem event that caused a reset or interrupt. The event sets the condition in this register without regard to whether it is used to generate a System control Interrupt, Reset, NMI, or no action. The **PTBOCTL** register contains the action to be taken on the specific events. The combination of the **PWRTC** register outputs and the **PTBOCTL** register causes the appropriate interrupt or reset condition to occur and the corresponding status bits to be set.

Power-Temperature Cause (PWRTC)

Base 0x400F.E000
Offset 0x060
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0 | 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. |
| 4 | VDDA_UBOR | RW1C | 0 | $V_{DDA}$Under BOR Status<br>Value Description<br>0 $V_{DDA}$ has not tripped under voltage BOR comparison.<br>1 $V_{DDA}$ has tripped under voltage BOR comparison. |
| 3:1 | reserved | RO | 0 | 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. |
| 0 | VDD_UBOR | RW1C | 0 | $V_{DD}$Under BOR Status<br>Value Description<br>0 $V_{DD}$ has not tripped under voltage BOR comparison.<br>1 $V_{DD}$ has tripped under voltage BOR comparison. |

## Register 9: NMI Cause Register (NMIC), offset 0x064

This register provides the detailed information on the cause of an NMI interrupt. These bits are set via hardware when the event occurs AND the higher level control indicates that it should be NMI event.

**Note:** The **NMIC** register has to be cleared by the following sequence:

1. Read the **NMIC** register to identify the source of the NMI.
2. Clear the source of the NMI.
3. Read the **NMIC** register again to check the status.
4. Write a 0 into the **NMIC** register bit that corresponds with the NMI source.
5. Read the **NMIC** to check whether it is cleared. If not, repeat 3 on page 283 and 4 on page 283 again.

NMI Cause Register (NMIC)
Base 0x400F.E000
Offset 0x064
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | MOSCFAIL |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | TAMPER | reserved | | | WDT1 | reserved | WDT0 | POWER | reserved | EXTERNAL |
| Type | RO | RO | RO | RO | RO | RO | RW | RO | RO | RO | RW | RO | RW | RW | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0 | 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. |
| 16 | MOSCFAIL | RW | 0 | MOSC Failure NMI<br>Value Description<br>0 No MOSC failure has occurred.<br>1 An NMI has occurred due to a MOSC failure. |
| 15:10 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | TAMPER | RW | 0 | Tamper Event NMI<br><br>Value Description<br>0 No tamper event has occurred.<br>1 An NMI has occurred due to a tamper event<br><br>See the HIB module tamper registers for more details on the tamper event. |
| 8:6 | reserved | RO | 0 | 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 | WDT1 | RW | 0 | Watch Dog Timer (WDT) 1 NMI<br><br>Value Description<br>0 No WDT 1 timeout has occurred.<br>1 An NMI has occurred due to a Watchdog Timer 1 timeout event. |
| 4 | reserved | RO | 0 | 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. |
| 3 | WDT0 | RW | 0 | Watch Dog Timer (WDT) 0 NMI<br><br>Value Description<br>0 No WDT 0 timeout has occurred.<br>1 An NMI has occurred due to a Watchdog Timer 0 timeout event. |
| 2 | POWER | RW | 0 | Power/Brown Out Event NMI<br><br>Value Description<br>0 No power event has occurred.<br>1 An NMI has occurred due to a power event.<br><br>See **PWRTC** register for exact cause of power/brown-out event. |
| 1 | reserved | RO | 0 | 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. |
| 0 | EXTERNAL | RW | 0 | External Pin NMI<br><br>Value Description<br>0 No `NMI` pin event has occurred.<br>1 The `NMI` pin was asserted by external hardware. |

## Register 10: Main Oscillator Control (MOSCCTL), offset 0x07C

This register provides control over the features of the main oscillator, including the ability to enable the MOSC clock verification circuit, what action to take when the MOSC fails, and whether or not a crystal is connected. When enabled, this circuit monitors the frequency of the MOSC to verify that the oscillator is operating within specified limits. If the clock goes invalid after being enabled, the microcontroller issues a power-on reset and reboots to the NMI handler or generates an interrupt.

**Note:** If the MOSC is chosen as the clock to the Ethernet PHY then software has to enable the MOSC before enabling the Ethernet PHY by setting the P0 bit in the **PCEPHY**.

Main Oscillator Control (MOSCCTL)

Base 0x400F.E000
Offset 0x07C
Type RW, reset 0x0000.000C

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | OSCRNG | PWRDN | NOXTAL | MOSCIM | CVAL |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x0000.000 | 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. |
| 4 | OSCRNG | RW | 0 | Oscillator Range<br>Specifies the frequency range of operation of the oscillator.<br>Value Description<br>0 Low Frequency Range<br>1 High Frequency Range (equal to or greater than 10 MHz). |
| 3 | PWRDN | RW | 1 | Power Down<br>Provides user control over powering down the main oscillator circuit.<br>Value Description<br>0 Power to main oscillator circuit is enabled.<br>1 Main Oscillator circuit is powered down.<br>**Note:** This bit should be cleared when using a crystal and set for single-ended mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | NOXTAL | RW | 1 | No MOSC/Crystal Connected<br>Provides the user control over the power drawn from the main oscillator circuit. This bit should be set when either crystal or single-ended mode is being used.<br>If the application needs MOSC, this bit should be cleared.<br>Value Description<br>0 This bit should be cleared when a crystal or oscillator is connected to the OSC0 and OSC1 inputs, regardless of whether or not the MOSC is used or powered down.<br>**Note:** For proper clock functionality when switching to crystal mode, software must clear this bit and set the PWRDN bit in a single write access.<br>1 This bit should be set when a crystal or external oscillator is not connected to the OSC0 and OSC1 inputs to reduce power consumption. |
| 1 | MOSCIM | RW | 0 | MOSC Failure Action<br>Value Description<br>0 If the MOSC fails, a MOSC failure reset is generated and reboots to the NMI handler.<br>1 If the MOSC fails, an interrupt is generated as indicated by the MOSRIS bit in the **RIS** register.<br>Regardless of the action taken, if the MOSC fails, the oscillator source is switched to the PIOSC automatically. |
| 0 | CVAL | RW | 0 | Clock Validation for MOSC<br>Value Description<br>0 The MOSC monitor circuit is disabled.<br>1 The MOSC monitor circuit is enabled. |

## Register 11: Run and Sleep Mode Configuration Register (RSCLKCFG), offset 0x0B0

**Important:** When transitioning the system clock configuration to use the MOSC as the fundamental clock source, the PWRDN bit must be set in the **MOSCCTL** register prior to reselecting the MOSC for proper operation.

Run and Sleep Mode Configuration Register (RSCLKCFG)

Base 0x400F.E000
Offset 0x0B0
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MEMTIMU | NEWFREQ | ACG | USEPLL | PLLSRC | | | | OSCSRC | | | | OSYSDIV | | | |
| Type | R0/W | R0/W | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OSYSDIV | | | | | | PSYSDIV | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MEMTIMU | R0/W | 0 | Memory Timing Register Update<br>Setting this bit causes the **MEMTIM0** register value to be applied, and the memory timing to be updated. Execution and access is suspended during the change.<br>This bit is automatically cleared by hardware. |
| 30 | NEWFREQ | R0/W | 0 | New PLLFREQ Accept<br>This bit controls the activation of the values in the **PLLFREQ0** and **PLLFREQ1** registers as applied to the PLL. Until NEWFREQ is written to a 1, writes to the **PLLFREQ0** and **PLLFREQ1** are deferred. When written with a 1, the values stored in **PLLFREQ0** and **PLLFREQ1** are applied to the PLL.<br>This bit is automatically cleared by hardware. Software will not check the value after being set. |
| 29 | ACG | RW | 0x0 | Auto Clock Gating<br>This bit specifies whether the system uses the **Sleep-Mode Clock Gating Control (SCGCn)** registers and **Deep-Sleep-Mode Clock Gating Control (DCGCn)** registers if the microcontroller enters a Sleep or Deep-Sleep mode (respectively).<br>Value Description<br>0 The **Run-Mode Clock Gating Control (RCGCn)** registers are used when the microcontroller enters a sleep mode.<br>1 If the microcontroller is in sleep mode, the **SCGCn** registers are used to control the clocks distributed to the peripherals. If the microcontroller is in deep-sleep mode, the **DCGCn** registers are used to control the clocks distributed to the peripherals. The **SCGCn** and **DCGCn** registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode.<br>The **RCGCn** registers are always used to control the clocks in Run mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 28 | USEPLL | RW | 0 | Use PLL<br>This bit controls whether the clock source is specified by the OSCSRC field or the output of the PLL is provided to the system clock divider and serves as the system clock source.<br><br>Value Description<br>0 Clock source specified by OSCSRC field.<br>1 Clock source specified by the PLL |
| 27:24 | PLLSRC | RW | 0 | PLL Source<br>This field specifies the PLL input clock source<br><br>Value Description<br>0x0 PIOSC is the PLL input clock source<br>0x1-0x2 reserved<br>0x3 MOSC is the PLL input clock source<br>0x4-0xFF Reserved |
| 23:20 | OSCSRC | RW | 0 | Oscillator Source<br>This field specifies the oscillator source that becomes the oscillator clock (OSCCLK) source, which is used when the PLL is bypassed during run or sleep modes.<br><br>Value Description<br>0x0 PIOSC is oscillator source<br>0x1 reserved<br>0x2 LFIOSC is oscillator source<br>0x3 MOSC is oscillator source<br>0x4 Hibernation Module RTC Oscillator (RTCOSC)<br>0x5-0xFF reserved |
| 19:10 | OSYSDIV | RW | 0 | Oscillator System Clock Divisor<br>This field specifies the system clock divisor value for the oscillator path. This field is used when the USEPLL bit is 0.<br>$f_{syclk}=f_{oscclk}/(OSYSDIV+ 1)$<br>The divisor value is the OSYSDIV field value + 1 |
| 9:0 | PSYSDIV | RW | 0 | PLL System Clock Divisor<br>This field specifies the system clock divisor value for the PLL. This field is used when the USEPLL bit is 1.<br>$f_{syclk}=f_{VCO}/(PSYSDIV+ 1)$ |

## Register 12: Memory Timing Parameter Register 0 for Main Flash and EEPROM (MEMTIM0), offset 0x0C0

The **MEMTIM0** register provides timing parameters for the main Flash and EEPROM memories. The timing parameters apply to the memory while the system is in run or sleep mode; the clocking for these modes is consistent and unchanged since the system clock frequency and source remains unchanged during transitions between run-to-sleep and sleep-back-to-run. Writes to **MEMTIM0** do not have any effect on system state; the register contents are applied only when the MEMTIMU bit in the **RSCLKCFG** register is set. Doing so allows the software to execute out of the same memory system for which the timing parameters are being modified.

Depending on the CPU frequency, the application must program specific values into the fields of the **Memory Timing Parameter Register 0 for Main Flash and EEPROM (MEMTIM0)**. The following table details the bit field values that are required for the given CPU frequency ranges.

**Table 5-12. MEMTIM0 Register Configuration versus Frequency**

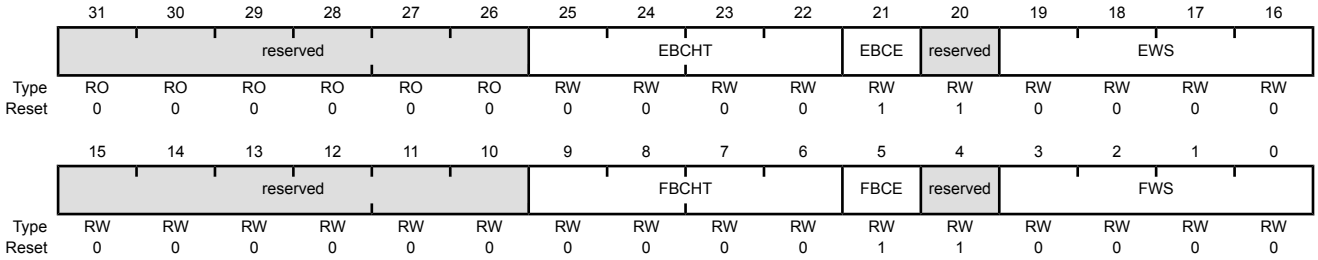
| CPU Frequency range (f) in MHz | Time Period Range (t) in ns | FBCHT/EBCHT | FBCE/EBCE | FWS/EWS |
|---|---|---|---|---|
| 16 | 62.5 | 0x0 | 1 | 0x0 |
| 16 < f ≤ 40 | 62.5 > t ≥ 25 | 0x2 | 0 | 0x1 |
| 40 < f ≤60 | 25 > t ≥ 16.67 | 0x3 | 0 | 0x2 |
| 60< f ≤80 | 16.67 > t ≥ 12.5 | 0x4 | 0 | 0x3 |
| 80 < f ≤100 | 12.5 > t ≥ 10 | 0x5 | 0 | 0x4 |
| 100< f ≤120 | 10 > t ≥ 8.33 | 0x6 | 0 | 0x5 |

**Note:** The associated Flash and EEPROM fields in the **MEMTIM0** register must be programmed to the same values. For example, the FWS field must be programmed to the same value as the EWS field.

Refer to “Flash Memory” on page 636 and “EEPROM” on page 647 for more information about Flash and EEPROM programming.

Memory Timing Parameter Register 0 for Main Flash and EEPROM (MEMTIM0)

Base 0x400F.E000
Offset 0x0C0
Type RW, reset 0x0030.0030




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x0000.000 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25:22 | EBCHT | RW | 0x0 | EEPROM Clock High Time<br>Specifies the length of the EEPROM bank clock high time<br><br>Value Description<br>0x0 1/2 system clock period<br>0x1 1 system clock period<br>0x2 1.5 system clock periods<br>0x3 2 system clock periods<br>0x4 2.5 system clock periods<br>0x5 3 system clock periods<br>0x6 3.5 system clock periods<br>0x7 4 system clock periods<br>0x8 4.5 system clock periods |
| 21 | EBCE | RW | 1 | EEPROM Bank Clock Edge<br>Specifies the relationship of EEPROM clock to system clock<br><br>Value Description<br>0 EEPROM clock rising aligns with system clock rising<br>1 EEPROM clock rising aligns with system clock falling |
| 20 | reserved | RW | 1 | 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. |
| 19:16 | EWS | RW | 0 | EEPROM Wait States<br>This field specifies the number of wait states inserted.<br><br>Value Description<br>0x0 0 wait states<br>0x1 1 wait state<br>0x2 2 wait states<br>0x3 3 wait states<br>0x4 4 wait states<br>0x5 5 wait states<br>0x6 6 wait states<br>0x7 7 wait states<br>0x8-0xF reserved |
| 15:10 | reserved | RW | 0x0 | 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. |

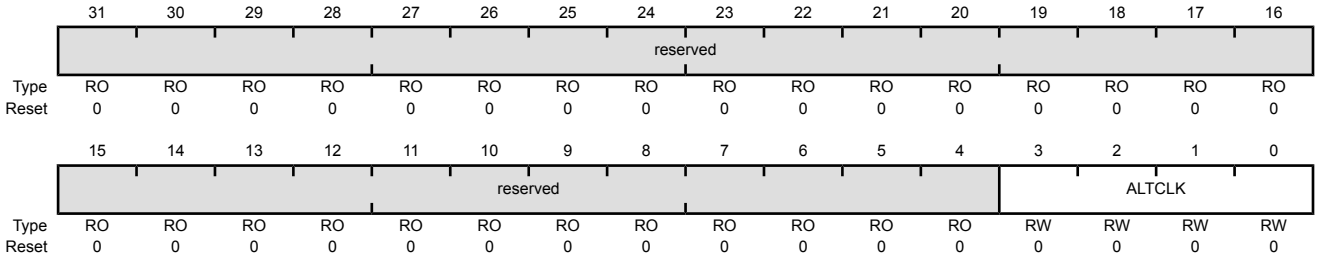
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9:6 | FBCHT | RW | 0x0 | Flash Bank Clock High Time<br>Specifies the length of the flash bank clock high time<br><br>Value Description<br>0x0 1/2 system clock period<br>0x1 1 system clock period<br>0x2 1.5 system clock periods<br>0x3 2 system clock periods<br>0x4 2.5 system clock periods<br>0x5 3 system clock periods<br>0x6 3.5 system clock periods<br>0x7 4 system clock periods<br>0x8 4.5 system clock periods |
| 5 | FBCE | RW | 1 | Flash Bank Clock Edge<br>Specifies the relationship of flash clock to system clock<br><br>Value Description<br>0 Flash clock rising aligns with system clock rising<br>1 Flash clock rising aligns with system clock falling |
| 4 | reserved | RW | 1 | 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. |
| 3:0 | FWS | RW | 0 | Flash Wait State<br>This field specifies the number of wait states inserted.<br><br>Value Description<br>0x0 0 wait states<br>0x1 1 wait state<br>0x2 2 wait states<br>0x3 3 wait states<br>0x4 4 wait states<br>0x5 5 wait states<br>0x6 6 wait states<br>0x7 7 wait states<br>0x8-0xF reserved |

## Register 13: Alternate Clock Configuration (ALTCLKCFG), offset 0x138

The **ALTCLKCFG** register specifies the alternate clock source used by many of the peripherals.

Alternate Clock Configuration (ALTCLKCFG)

Base 0x400F.E000
Offset 0x138
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3:0 | ALTCLK | RW | 0x0 | Alternate Clock Source<br>This provides a clock source of numerous frequencies to the general-purpose timer, SSI, and UART modules. Note that if the Hibernation Real-time Clock Output is selected, the clock source must also be enabled in the Hibernation module.<br><br>Value Description<br>0x0 PIOSC<br>0x1-0x2 reserved<br>0x3 Hibernation Module Real-time clock output (RTCOSC)<br>0x4 Low-frequency internal oscillator (LFIOSC)<br>0x5-0x15 reserved |

## Register 14: Deep Sleep Clock Configuration Register (DSCLKCFG), offset 0x144

The **DSCLKCFG** register specifies the behavior of the clock system while in deep sleep.

Note that the MOSCDPD bit not only affects deep-sleep mode, but all other modes as well depending on the value of the bit. Please refer to the following table when programming this bit:

**Table 5-13. MOSC Configurations**

| PWRDN bit | MOSCDPD field | Result |
|---|---|---|
| 0 | 0 | MOSC is powered ON in run and sleep modes, but is disabled in accidental power down, when the PWRDN bit is set in the **MOSCCTL** register, or in deep-sleep mode only if it is not the deep-sleep clock source (DSOSCSRC !==0x3). |
| 0 | 1 | MOSC is powered and running in run, sleep and deep-sleep modes. |
| 1 | 0 | MOSC is powered off, and does not run in any mode. Please note, that in this configuration, when the MOSC is disabled, the MOSC must not be chosen as a clock source or indeterminate results occur. |
| 1 | 1 | MOSC runs and does not disable itself in run, sleep, and deep-sleep modes regardless of the fact that the PWRDN bit is set. |

**Note:** The MOSCDPD bit has an effect in all modes of operation

**Note:** If the MOSC is chosen as the Deep-Sleep clock source in the **DSCLKCFG** register, the MOSC must also be configured as the Run and Sleep clock source in the **RSCLKCFG** register prior to entering Deep Sleep. If the PIOSC, LFIOSC, or Hibernation RTC Module Oscillator (HIBLFIOSC or 32-kHz crystal) is configured as the Run and Sleep clock source in the **RSCLKFCFG** register, and the MOSC is configured as the Deep-Sleep clock source in the **DSCLKCFG** register, then two outcomes are possible:

- If the PIOSC is still powered in Deep Sleep (using the PIOSCPD bit in the **DSCLKCFG** register) then the PIOSC is utilized as the clock source when entering Deep Sleep and the device enters and exits the Deep-Sleep state normally. The MOSC is not used as the clock source in Deep Sleep.
- If the PIOSC has been configured to be powered down in Deep Sleep, then the device can enter the Deep-Sleep state, but cannot exit properly. This situation can be avoided by programming the MOSC as the Run and Sleep clock source in the **RSCLKCFG** register prior to entering Deep Sleep.

Deep Sleep Clock Configuration Register (DSCLKCFG)

Base 0x400F.E000
Offset 0x144
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PIOSCPD | MOSCDPD | reserved | | | | | | DSOSCSRC | | | | reserved | | | |
| Type | RW | RW | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | DSSYSDIV | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | PIOSCPD | RW | 0 | PIOSC Power Down<br><br>Value Description<br>0 The PIOSC is active during deep sleep mode.<br>1 The PIOSC is disabled during sleep mode for additional power savings. |
| 30 | MOSCDPD | RW | 0 | MOSC Disable Power Down<br><br>This bit inhibits the MOSC from automatic or accidental power down. This bit is defined to ensure the MOSC circuit cannot be interrupted in uses where MOSC supplies a clock to the peripherals (for example, Ethernet PHY).<br><br>Value Description<br>0 During deep-sleep (if DSOSCSRC is not MOSC), accidental power down or when the PWRDWN bit is set in the **MOSCCTL** register, the MOSC is powered down.<br>1 MOSC is not powered off during automatic or accidental power down.<br>**Note:** MOSC is also not powered off if DSOSCRC is programmed to be MOSC.<br>**Note:** This bit should only be set after software configures the **MOSCCTL** register. Setting the MOSCDPD bit masks writes to PWRDN bit in the **MOSCCTL** register. |
| 29:24 | reserved | RO | 0 | 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. |
| 23:20 | DSOSCSRC | RW | 0x0 | Deep Sleep Oscillator Source<br><br>This field specifies the oscillator source that becomes the oscillator clock (OSCCLK) source, which is used when the PLL is bypassed during deep sleep mode.<br><br>Value Description<br>0x0 PIOSC<br>0x1 reserved<br>0x2 LFIOSC<br>0x3 MOSC<br>0x4 Hibernation Module RTCOSC<br>0x5-0xF reserved |
| 19:10 | reserved | RO | 0x000 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9:0 | DSSYSDIV | RW | 0x0 | Deep Sleep Clock Divisor<br>This field specifies the system clock divisor value during deep sleep mode. The clock source selected by DSOSCSRC is divided by DSSYSDIV + 1:<br>$f_{SYSCLK}=f_{OSCCLK}/(DSSYSDIV + 1)$<br>**Note:** Values 0x0 and 0x1 should not be used. If Deep-Sleep clock divide by 1 or divide by 2 is desired, the OSYSDIV bit field of the **RSCLKCFG** register must be configured for the desired Deep-Sleep divider before entering Deep-Sleep. In this case, the Q post-divider bit field in the **PLLFREQ1** register may need to be adjusted to keep the system clock frequency within the maximum clock frequency before entering Deep-Sleep. |

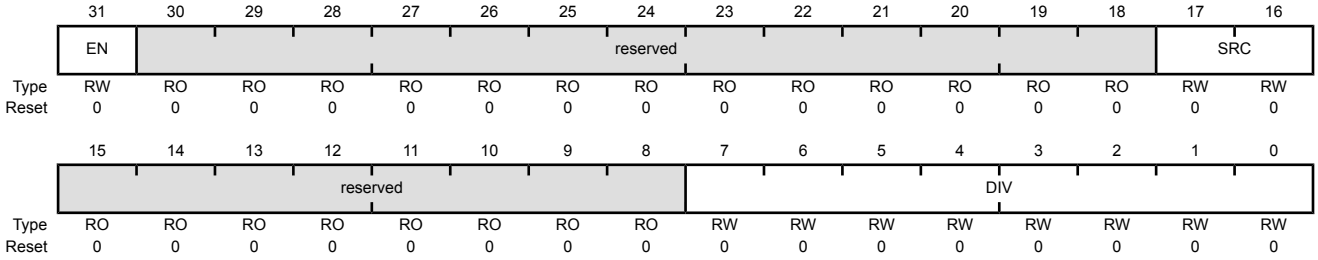
## Register 15: Divisor and Source Clock Configuration (DIVSCLK), offset 0x148

The **DIVSCLK** register specifies the source and divisor of the DIVSCLK reference clock output. This signal can be used as a clock source to an external device but bears no timing relationship to other signals.

**Note:** The DIVSCLK signal output is not synchronized to the System Clock.

Divisor and Source Clock Configuration (DIVSCLK)

Base 0x400F.E000
Offset 0x148
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | EN | RW | 0 | DIVSCLK Enable<br>This bit enables the generation of the DIVSCLK clock output. It resets to 0 to disable the output thereby reducing initial current/power consumption.<br><br>Value Description<br>0 The clock output is disabled<br>1 Clock output is enabled. |
| 30:18 | reserved | RO | 0 | 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. |
| 17:16 | SRC | RW | 0 | Clock Source<br>Selects the reference clock used to generate the output.<br><br>Value Description<br>0x0 System Clock<br>0x1 PIOSC<br>0x2 MOSC<br>0x3 reserved |
| 15:8 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | DIV | RW | 0 | Divisor Value<br>This field controls the ratio of the source clock to the output clock. The output clock frequency is equal to the source clock frequency divided by the DIV field value plus 1. |

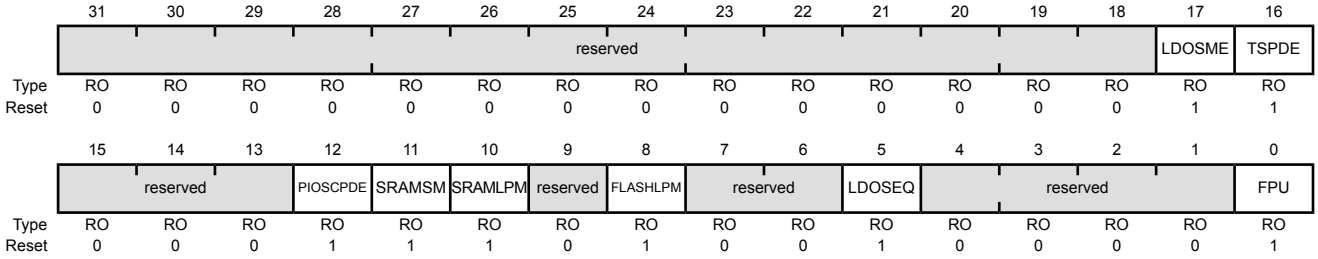
| Value | Description |
|---|---|
| 0x0 | Divided by 1 |
| 0x1 | Divided by 2 |
| .... | ..... |
| N | Divided by N |

## Register 16: System Properties (SYSPROP), offset 0x14C

This register provides information on whether certain System Control properties are present on the microcontroller.

System Properties (SYSPROP)

Base 0x400F.E000
Offset 0x14C
Type RO, reset 0x0003.1F31




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0x0 | 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. |
| 17 | LDOSME | RO | 0x1 | LDO Sleep Mode Enable<br>Value Description<br>0 The LDOSM bit of the **DSLPPWRCFG** register is ignored.<br>1 The LDOSM bit of the **DSLPPWRCFG** register can be set to place the LDO in a low-power mode when the deep sleep state is entered. |
| 16 | TSPDE | RO | 0x1 | Temp Sense Power Down Enable<br>This bit allows the internal temperature sensor in the ADC to be powered off in Deep-Sleep mode.<br>Value Description<br>0 The TSPD bit of the **DSLPPWRCFG** register is ignored.<br>1 The TSPD bit of the **DSLPPWRCFG** register can be set to power off the temperature sensor in deep sleep mode. |
| 15:13 | reserved | RO | 0x0 | 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. |
| 12 | PIOSCPDE | RO | 0x1 | PIOSC Power Down Present<br>This bit determines whether the PIOSCPD bit in the **DSCLKCFG** register can be set to power down the PIOSC in Deep-Sleep mode.<br>Value Description<br>0 The status of the PIOSCPD bit is ignored.<br>1 The PIOSCPD bit can be set to power down the PIOSC in Deep-Sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | SRAMSM | RO | 0x1 | SRAM Sleep/Deep-Sleep Standby Mode Present<br>This bit determines whether the SRAMPM field in the **SLPPWRCFG** and **DSLPPWRCFG** registers can be configured to put the SRAM into Standby mode while in Sleep or Deep-Sleep mode.<br>Value Description<br>0 A value of 0x1 in the SRAMPM fields is ignored.<br>1 The SRAMPM fields can be configured to put the SRAM into Standby mode while in Sleep or Deep-Sleep mode. |
| 10 | SRAMLPM | RO | 0x1 | SRAM Sleep/Deep-Sleep Low Power Mode Present<br>This bit determines whether the SRAMPM field in the **SLPPWRCFG** and **DSLPPWRCFG** registers can be configured to put the SRAM into Low Power mode while in Sleep or Deep-Sleep mode.<br>Refer to "Sleep Modes" on page 2120 for information regarding wake times from Sleep and Deep-Sleep.<br>Value Description<br>0 A value of 0x3 in the SRAMPM fields is ignored.<br>1 The SRAMPM fields can be configured to put the SRAM into Low Power mode while in Sleep or Deep-Sleep mode. |
| 9 | reserved | RO | 0 | 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. |
| 8 | FLASHLPM | RO | 0x1 | Flash Memory Sleep/Deep-Sleep Low Power Mode Present<br>This bit determines whether the FLASHPM field in the **SLPPWRCFG** and **DSLPPWRCFG** registers can be configured to put the Flash memory into Low Power mode while in Sleep or Deep-Sleep mode.<br>Refer to "Sleep Modes" on page 2120 for information regarding wake times from Sleep and Deep-Sleep.<br>Value Description<br>0 A value of 0x2 in the FLASHPM fields is ignored.<br>1 The FLASHPM fields can be configured to put the Flash memory into Low Power mode while in Sleep or Deep-Sleep mode. |
| 7:6 | reserved | RO | 0x0 | 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 | LDOSEQ | RO | 0x1 | Automatic LDO Sequence Control Present<br>This bit indicates that the ability to sequence the LDO output voltage is available during Sleep and Deep-Sleep modes.<br>Value Description<br>0 Software cannot set the VADJEN bit in the **LDOSPCTL** and **LDODPCTL** registers.<br>1 Software can set the VADJEN bit in the **LDOSPCTL** and **LDODPCTL** registers. |

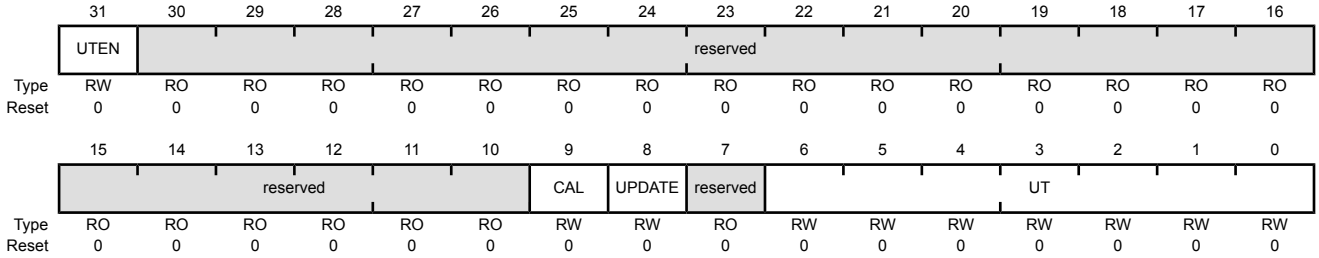
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | FPU | RO | 0x1 | FPU Present<br>This bit indicates if the FPU is present in the Cortex-M4 core.<br>Value Description<br>0 FPU is not present.<br>1 FPU is present. |

## Register 17: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150

This register provides the ability to update or recalibrate the precision internal oscillator. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

Precision Internal Oscillator Calibration (PIOSCCAL)

Base 0x400F.E000
Offset 0x150
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | UTEN | RW | 0 | Use User Trim Value<br><br>Value Description<br>0 The factory calibration value is used for an update trim operation.<br>1 The trim value in bits[6:0] of this register are used for any update trim operation. |
| 30:10 | reserved | RO | 0x0000 | 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. |
| 9 | CAL | RW | 0 | Start Calibration<br><br>Value Description<br>0 No action.<br>1 Starts a new calibration of the PIOSC. Results are in the **PIOSCSTAT** register. The resulting trim value from the operation is active in the PIOSC after the calibration completes. The result overrides any previous update trim operation whether the calibration passes or fails.<br><br>This bit is auto-cleared after it is set. |
| 8 | UPDATE | RW | 0 | Update Trim<br><br>Value Description<br>0 No action.<br>1 Updates the PIOSC trim value with the UT bit or the DT bit in the **PIOSCSTAT** register. Used with UTEN.<br><br>This bit is auto-cleared after the update. |
| 7 | reserved | RO | 0 | 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. |

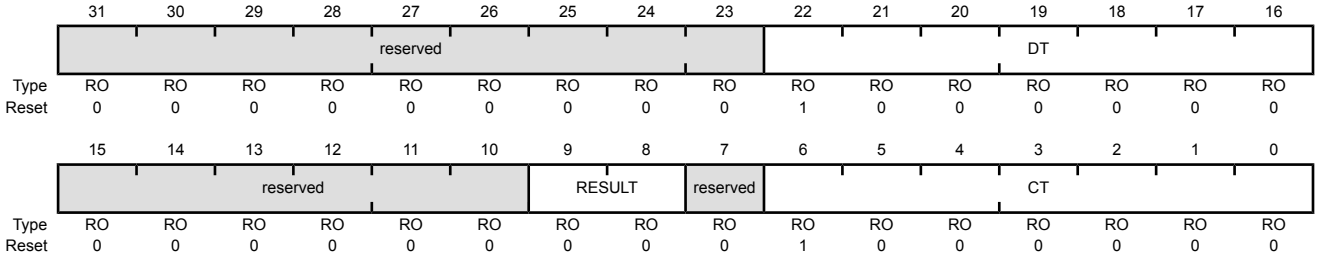
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6:0 | UT | RW | 0x0 | User Trim Value<br>User trim value that can be loaded into the PIOSC. |

## Register 18: Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154

This register provides the user information on the PIOSC calibration. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

Precision Internal Oscillator Statistics (PIOSCSTAT)

Base 0x400F.E000
Offset 0x154
Type RO, reset 0x0040.0040




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:23 | reserved | RO | 0x00 | 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. |
| 22:16 | DT | RO | 0x40 | Default Trim Value<br>This field contains the default trim value. This value is loaded into the PIOSC after every full power-up. |
| 15:10 | reserved | RO | 0x0 | 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. |
| 9:8 | RESULT | RO | 0 | Calibration Result<br>Value Description<br>0x0 Calibration has not been attempted.<br>0x1 The last calibration operation completed to meet 1% accuracy.<br>0x2 The last calibration operation failed to meet 1% accuracy.<br>0x3 Reserved |
| 7 | reserved | RO | 0 | 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. |
| 6:0 | CT | RO | 0x40 | Calibration Trim Value<br>This field contains the trim value from the last calibration operation. After factory calibration CT and DT are the same. |

## Register 19: PLL Frequency 0 (PLLFREQ0), offset 0x160

This register always contains the variables used to configure the PLL. If the PLL is reprogrammed, it must go through a relock sequence which is defined by the parameter $T_{READY}$ in Table 32-16 on page 2112. When controlling this register directly, software must change this value while the PLL is powered down. Writes to **PLLFREQ0** are delayed from affecting the PLL until the **RSCLKCFG** register NEWFREQ bit is written with a 1.

The PLL frequency can be calculated using the following equation:

$f_{VCO} = (f_{IN} * MDIV)$

where

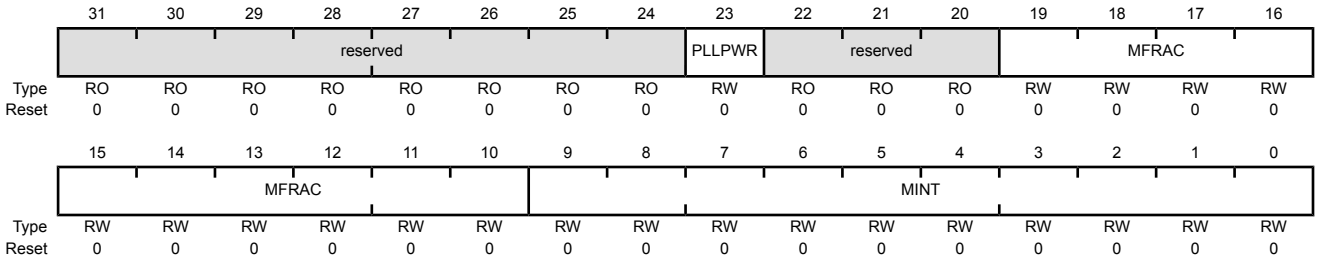
$f_{IN} = f_{XTAL}/(Q+1)(N+1)$ or $f_{PIOSC}/(Q+1)(N+1)$

$MDIV = MINT + (MFRAC / 1024)$

The Q and N values are programmed in the **PLLFREQ1** register. Note that to reduce jitter, MFRAC should be programmed to 0x0.

PLL Frequency 0 (PLLFREQ0)

Base 0x400F.E000
Offset 0x160
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x000 | 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. |
| 23 | PLLPWR | RW | 0 | PLL Power<br>This bit controls power to the PLL. If set, the PLL power is applied and the PLL will oscillate based on the values in the **PLLFREQ0** and **PLLFREQ1** registers. |
| 22:20 | reserved | RO | 0x0 | 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. |
| 19:10 | MFRAC | RW | 0 | PLL M Fractional Value |
| 9:0 | MINT | RW | 0x00 | PLL M Integer Value<br>This field contains the integer value of the PLL M value. |

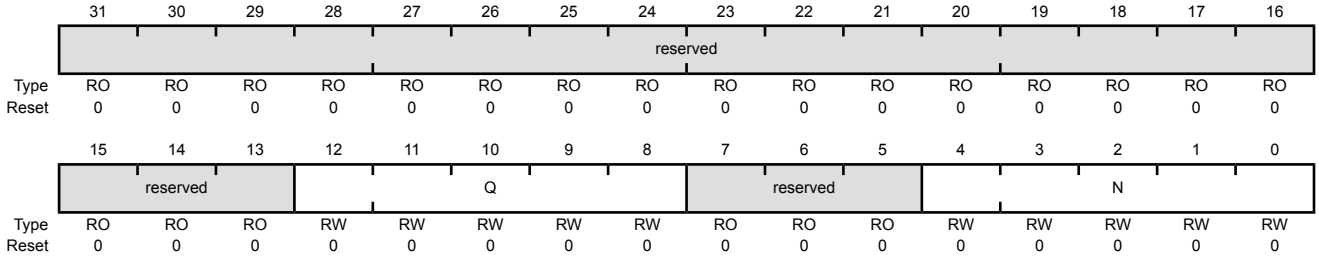
## Register 20: PLL Frequency 1 (PLLFREQ1), offset 0x164

This register always contains the current Q and N values presented to the system PLL. If the PLL is reconfigured, it must go through a relock sequence which takes about 128 PIOSC clocks. When controlling this register directly, software must change this value while the PLL is powered down. Writes to **PLLFREQ0** are delayed from affecting the PLL until the **RSCLKCFG** register NEWFREQ bit is written with a 1.

The MINT and MFRAC fields are present in the **PLLFREQ0** register.

PLL Frequency 1 (PLLFREQ1)

Base 0x400F.E000
Offset 0x164
Type RW, reset 0x0000.0000



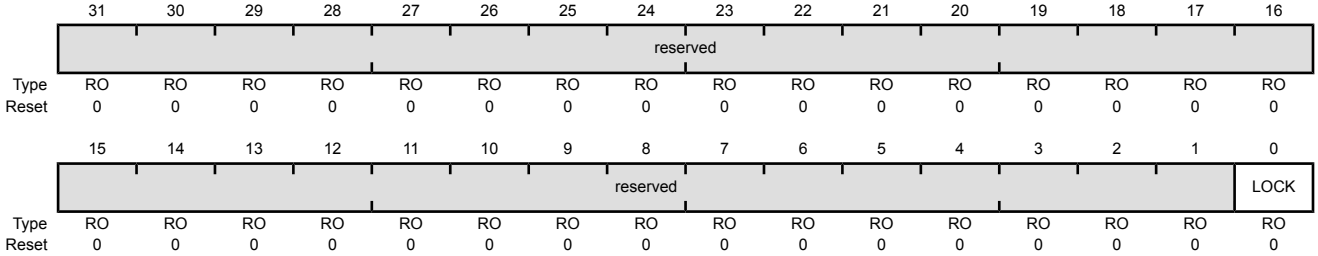

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:13 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 12:8 | Q | RW | 0x0 | PLL Q Value<br>This field contains the PLL Q value. |
| 7:5 | reserved | RO | 0x0 | 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. |
| 4:0 | N | RW | 0x0 | PLL N Value<br>This field contains the PLL N value. |

## Register 21: PLL Status (PLLSTAT), offset 0x168

This register shows the direct status of the PLL lock.

PLL Status (PLLSTAT)

Base 0x400F.E000
Offset 0x168
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | LOCK | RO | 0x0 | PLL Lock |

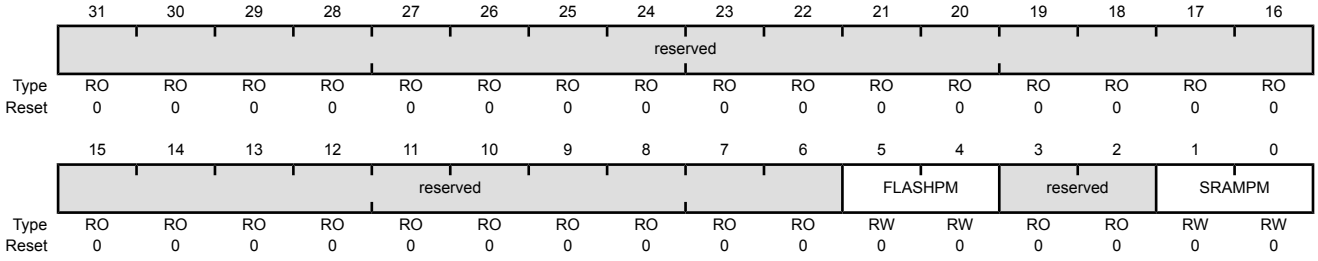
| Value | Description |
|---|---|
| 0 | The PLL is unpowered or is not yet locked. |
| 1 | The PLL powered and locked. |

## Register 22: Sleep Power Configuration (SLPPWRCFG), offset 0x188

This register provides configuration information for the power control of the SRAM and Flash memory while in Sleep mode.

Sleep Power Configuration (SLPPWRCFG)

Base 0x400F.E000
Offset 0x188
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.00 | 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 | FLASHPM | RW | 0x0 | Flash Power Modes<br><br>Value Description<br>0x0 Active Mode<br>Flash memory is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Sleep mode.<br>0x1 Reserved<br>0x2 Low Power Mode<br>Flash memory is placed in low power mode. This mode provides the lowers power consumption but requires more time to come out of Sleep mode.<br>0x3 Reserved |
| 3:2 | reserved | RO | 0x0 | 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. |

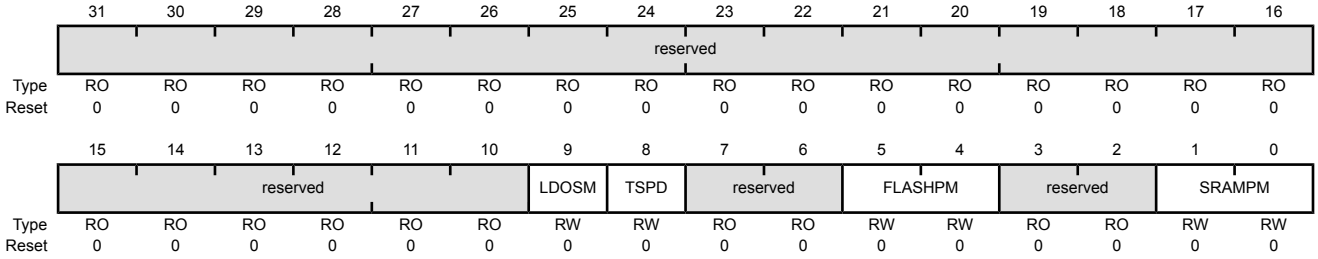
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | SRAMPM | RW | 0x0 | SRAM Power Modes<br>This field controls the low power modes of the on-chip SRAM , including the USB SRAM while the microcontroller is in Sleep mode.<br><br>Value Description<br>0x0 Active Mode<br>SRAM is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Sleep mode.<br>0x1 Standby Mode<br>SRAM is placed in standby mode while in Sleep mode.<br>0x2 Reserved<br>0x3 Low Power Mode<br>SRAM is placed in low power mode. This mode provides the slowest time to sleep and wakeup but the lowest power consumption while in Sleep mode. |

## Register 23: Deep-Sleep Power Configuration (DSLPPWRCFG), offset 0x18C

This register provides configuration information for the power control of the SRAM and Flashmemory while in Deep-Sleep mode.

Deep-Sleep Power Configuration (DSLPPWRCFG)

Base 0x400F.E000
Offset 0x18C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9 | LDOSM | RW | 0 | LDO Sleep Mode<br><br>Value Description<br>0 LDO is disabled in sleep-mode.<br>1 LDO is placed in a low power mode when deep sleep mode is entered. |
| 8 | TSPD | RW | 0 | Temperature Sense Power Down<br>This bit controls low power mode for the internal temperature sensor in the ADC.<br><br>Value Description<br>0 Temperature sensor in the ADC is disabled in sleep-mode.<br>1 The internal temperature sensor in the ADC is placed in a low power mode when deep sleep mode is entered. |
| 7:6 | reserved | RO | 0x0 | 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. |

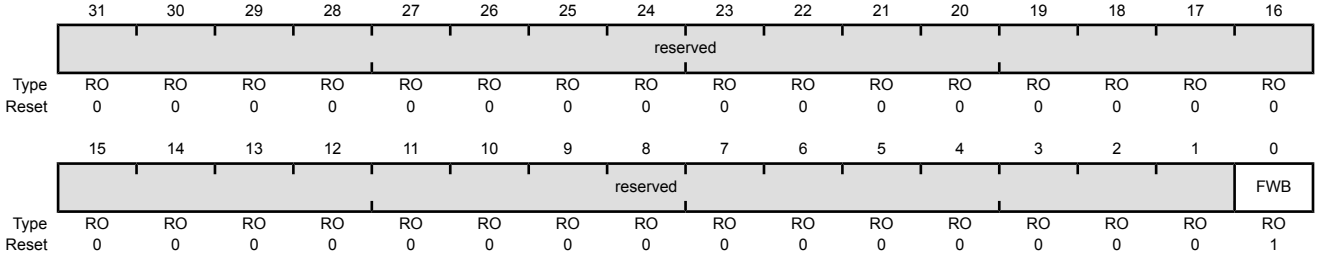
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | FLASHPM | RW | 0x0 | Flash Power Modes<br>This field enables the Flash to be placed in a Low Power Mode.<br>Refer to "Sleep Modes" on page 2120 for information regarding wake times from Sleep and Deep-Sleep.<br>If using the LFIOSC as the Deep-Sleep clock source, FLASHPM = 0x2 must be used. If FLASHPM = 0x0 and the LFIOSC is used, current could be higher and could vary.<br>Value Description<br>0x0 Active Mode<br>Flash memory is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Deep-Sleep mode.<br>0x1 Reserved<br>0x2 Low Power Mode<br>Flash memory is placed in low power mode. This mode provides the lowers power consumption but requires more time to come out of Deep-Sleep mode.<br>0x3 Reserved |
| 3:2 | reserved | RO | 0x0 | 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. |
| 1:0 | SRAMPM | RW | 0x0 | SRAM Power Modes<br>This field controls the low power modes of the on-chip SRAM , including the USB SRAM while the microcontroller is in Deep-Sleep mode.<br>Refer to "Sleep Modes" on page 2120 for information regarding wake times from Sleep and Deep-Sleep.<br>Value Description<br>0x0 Active Mode<br>SRAM is not placed in a lower power mode. This mode provides the fastest time to sleep and wakeup but the highest power consumption while the microcontroller is in Deep-Sleep mode.<br>0x1 Standby Mode<br>SRAM is place in standby mode while in Deep-Sleep mode.<br>0x2 Reserved<br>0x3 Low Power Mode<br>SRAM is placed in low power mode. This mode provides the slowest time to sleep and wakeup but the lowest power consumption while in Deep-Sleep mode. |

## Register 24: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0

This register is predefined by the part and can be used to verify features.

Non-Volatile Memory Information (NVMSTAT)

Base 0x400F.E000
Offset 0x1A0
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | FWB | RO | 0x1 | 32 Word Flash Write Buffer Available<br>When set, indicates that the 32 word Flash memory write buffer feature is available. |

## Register 25: LDO Sleep Power Control (LDOSPCTL), offset 0x1B4

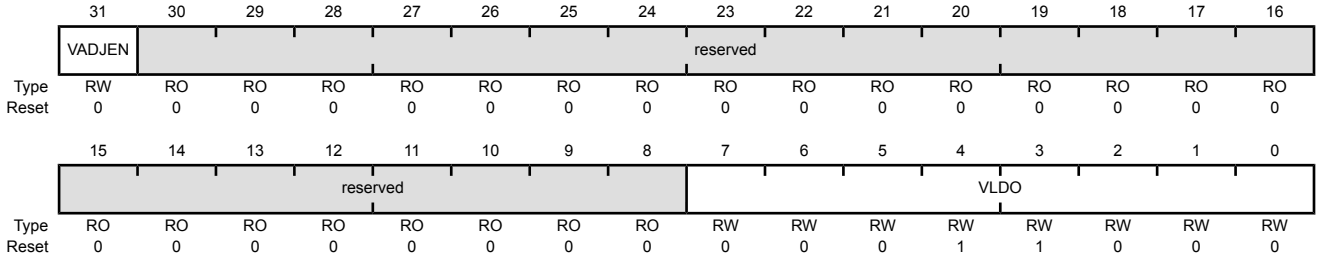
This register specifies the LDO output voltage in Sleep mode. This register should be configured while in Run Mode. If the VADJEN bit is set, writes can be made to the VLDO field within the provided encodings. The following table shows the maximum clock frequencies with respect to LDO Voltage.

**Table 5-14. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage**

| Operating Voltage (LDO) | Maximum System Clock Frequency | PIOSC |
|---|---|---|
| 1.2 | 120 MHz | 16 MHz |
| 0.9 | 30 MHz | 16 MHz |

LDO Sleep Power Control (LDOSPCTL)

Base 0x400F.E000
Offset 0x1B4
Type RW, reset 0x0000.0018




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | VADJEN | RW | 0 | Voltage Adjust Enable<br>This bit enables the value of the VLDO field to be used to specify the output voltage of the LDO in Sleep mode.<br><br>Value Description<br>0 The LDO output voltage is set to the factory default value in Sleep mode. The value of the VLDO field does not affect the LDO operation.<br>1 The LDO output value in Sleep mode is configured by the value in the VLDO field. |
| 30:8 | reserved | RO | 0x000.00 | 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. |

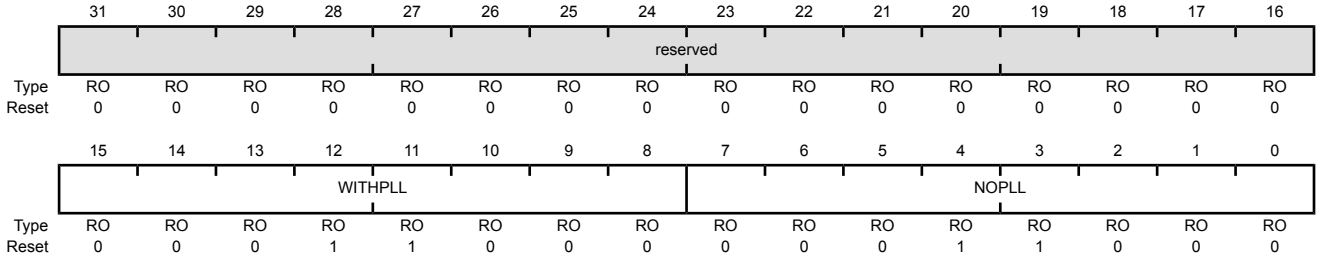
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | VLDO | RW | 0x18 | LDO Output Voltage<br>This field provides program control of the LDO output voltage in Sleep mode. The value of the field is only used for the LDO voltage when the VADJEN bit is set.<br><br>Value — Description<br>0x12 — 0.90 V<br>0x13 — 0.95 V<br>0x14 — 1.00 V<br>0x15 — 1.05 V<br>0x16 — 1.10 V<br>0x17 — 1.15 V<br>0x18 — 1.20 V<br>0x19 - 0xFF — reserved<br><br>**Note:** When using the USB module, the LDO must be configured to 1.2 V. |

## Register 26: LDO Sleep Power Calibration (LDOSPCAL), offset 0x1B8

This register provides factory determined values that are recommended for the VLDO field in the **LDOSPCTL** register while in Sleep mode. The reset value of this register cannot be determined until the product has been characterized.

LDO Sleep Power Calibration (LDOSPCAL)

Base 0x400F.E000
Offset 0x1B8
Type RO, reset 0x0000.1818




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | 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. |
| 15:8 | WITHPLL | RO | 0x18 | Sleep with PLL<br>The value in this field is the suggested value for the VLDO field in the **LDOSPCTL** register when using the PLL. This value provides the lowest recommended LDO output voltage for use with the PLL at the maximum specified value. |
| 7:0 | NOPLL | RO | 0x18 | Sleep without PLL<br>The value in this field is the suggested value for the VLDO field in the **LDOSPCTL** register when not using the PLL. This value provides the lowest recommended LDO output voltage for use without the PLL. |

## Register 27: LDO Deep-Sleep Power Control (LDODPCTL), offset 0x1BC

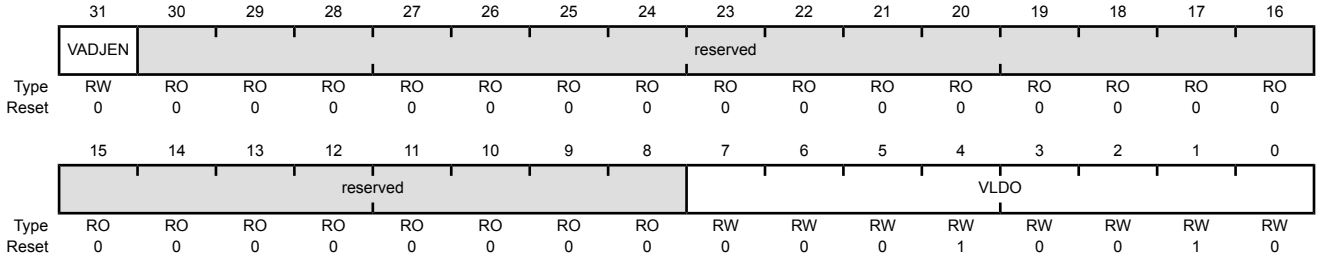
This register specifies the LDO output voltage in Sleep mode. This register should be configured while in Run Mode. If the VADJEN bit is set, writes can be made to the VLDO field within the provided encodings. The following table shows the maximum clock frequencies with respect to LDO Voltage.

**Table 5-15. Maximum System Clock and PIOSC Frequency with Respect to LDO Voltage**

| Operating Voltage (LDO) | Maximum System Clock Frequency | PIOSC |
|---|---|---|
| 1.2 | 120 MHz | 16 MHz |
| 0.9 | 30 MHz | 16 MHz |

LDO Deep-Sleep Power Control (LDODPCTL)

Base 0x400F.E000
Offset 0x1BC
Type RW, reset 0x0000.0012




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | VADJEN | RW | 0 | Voltage Adjust Enable<br>This bit enables the value of the VLDO field to be used to specify the output voltage of the LDO in Deep-Sleep mode.<br><br>Value Description<br>0 The LDO output voltage is set to the factory default value in Deep-Sleep mode. The value of the VLDO field does not affect the LDO operation.<br>1 The LDO output value in Deep-Sleep mode is configured by the value in the VLDO field. |
| 30:8 | reserved | RO | 0x000.00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | VLDO | RW | 0x12 | LDO Output Voltage<br>This field provides program control of the LDO output voltage in Deep-Sleep mode. The value of the field is only used for the LDO voltage when the VADJEN bit is set. |

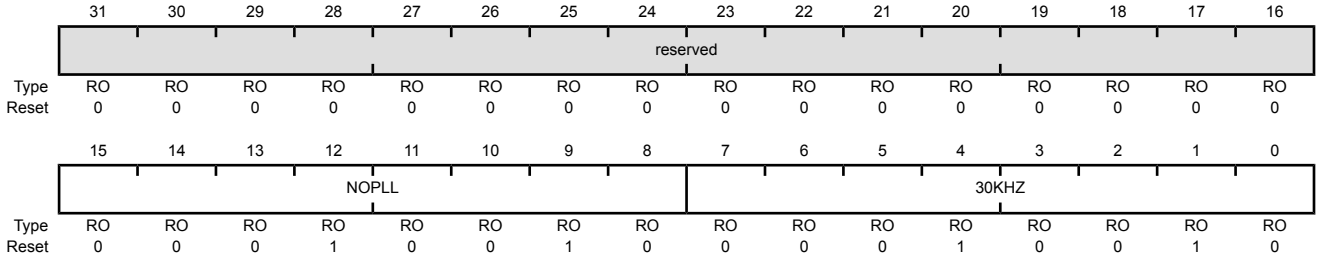
| Value | Description |
|---|---|
| 0x12 | 0.90 V |
| 0x13 | 0.95 V |
| 0x14 | 1.00 V |
| 0x15 | 1.05 V |
| 0x16 | 1.10 V |
| 0x17 | 1.15 V |
| 0x18 | 1.20 V |
| 0x19 - 0xFF | reserved |

## Register 28: LDO Deep-Sleep Power Calibration (LDODPCAL), offset 0x1C0

This register provides factory determined values that are recommended for the VLDO field in the **LDODPCTL** register while in Deep-Sleep mode. The reset value of this register cannot be determined until the product has been characterized.

LDO Deep-Sleep Power Calibration (LDODPCAL)

Base 0x400F.E000
Offset 0x1C0
Type RO, reset 0x0000.1212



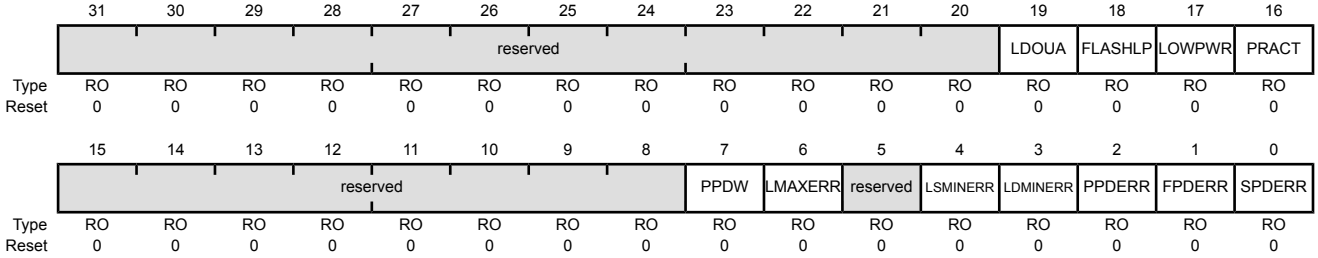

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:8 | NOPLL | RO | 0x12 | Deep-Sleep without PLL<br>The value in this field is the suggested value for the VLDO field in the **LDODPCTL** register when not using the PLL. This value provides the lowest recommended LDO output voltage for use with the system clock. |
| 7:0 | 30KHZ | RO | 0x12 | Deep-Sleep with IOSC<br>The value in this field is the suggested value for the VLDO field in the **LDODPCTL** register when not using the PLL. This value provides the lowest recommended LDO output voltage for use with the low-frequency internal oscillator. |

## Register 29: Sleep / Deep-Sleep Power Mode Status (SDPMST), offset 0x1CC

This register provides status information on the Sleep and Deep-Sleep power modes as well as some real time status that can be viewed by a debugger or the core if it is running. These events do not trigger an interrupt and are meant to provide information that can help tune software for power management. The status register gets written at the beginning of every Dynamic Power Management event request with the results of any error checking. There is no mechanism to clear the bits; they are overwritten on the next event. The LDOUA, FLASHLP, LOWPWR, PRACT bits provide real time data and there are no events to register that information.

Sleep / Deep-Sleep Power Mode Status (SDPMST)

Base 0x400F.E000
Offset 0x1CC
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x000 | 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. |
| 19 | LDOUA | RO | 0 | LDO Update Active<br>Value Description<br>0 The LDO voltage level is not changing.<br>1 The LDO voltage level is changing. |
| 18 | FLASHLP | RO | 0 | Flash Memory in Low Power State<br>Value Description<br>0 The Flash memory is currently in the active state.<br>1 The Flash memory is currently in the low power state as programmed in the **SLPPWRCFG** or **DSLPPWRCFG** register. |
| 17 | LOWPWR | RO | 0 | Sleep or Deep-Sleep Mode<br>Value Description<br>0 The microcontroller is currently in Run mode.<br>1 The microcontroller is currently in Sleep or Deep-Sleep mode and is waiting for an interrupt or is in the process of powering up. The status of this bit is not affected by the power state of the Flash memory or SRAM. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | PRACT | RO | 0 | Sleep or Deep-Sleep Power Request Active<br><br>Value Description<br>0 A power request is not active.<br>1 The microcontroller is currently in Deep-Sleep mode or is in Sleep mode and a request to put the SRAM and/or Flash memory into a lower power mode is currently active as configured by the **SLPPWRCFG** register. |
| 15:8 | reserved | RO | 0x00 | 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. |
| 7 | PPDW | RO | 0 | PIOSC Power Down Request Warning<br><br>Value Description<br>0 No error.<br>1 This bit indicates that the PIOSC was not powered off even though the PIOSCPD bit was set in the **DSLCLKCFG** register because the PIOSC was in use by a peripheral. |
| 6 | LMAXERR | RO | 0 | VLDO Value Above Maximum Error<br><br>Value Description<br>0 No error.<br>1 An error has occurred because software has requested that the LDO voltage be above the maximum value allowed using the VLDO bit in the **LDOSPCTL**, or **LDODPCTL** register.<br>In this situation, the LDO is set to the factory default value. |
| 5 | reserved | RO | 0 | 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. |
| 4 | LSMINERR | RO | 0 | VLDO Value Below Minimum Error in Sleep Mode<br><br>Value Description<br>0 No error.<br>1 An error has occurred because software has requested that the LDO voltage be below the minimum value allowed using the VLDO bit in the **LDOSPCTL** register.<br>In this situation, the LDO voltage is not changed when entering Sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | LDMINERR | RO | 0 | VLDO Value Below Minimum Error in Deep-Sleep Mode<br><br>Value Description<br>0 No error.<br>1 An error has occurred because software has requested that the LDO voltage be below the minimum value allowed using the VLDO bit in the **LDODPCTL** register.<br>In this situation, the LDO voltage is not changed when entering Deep-Sleep mode. |
| 2 | PPDERR | RO | 0 | PIOSC Power Down Request Error<br><br>Value Description<br>0 No error.<br>1 An error has occurred because software has requested that the PIOSC be powered down during Deep-Sleep and it is not possible to power down the PIOSC.<br>In this situation, the PIOSC is not powered down when entering Deep-Sleep mode. |
| 1 | FPDERR | RO | 0 | Flash Memory Power Down Request Error<br><br>Value Description<br>0 No error.<br>1 An error has occurred because software has requested a Flash memory power down mode that is not available using the FLASHPM field in the **SLPPWRCFG** or the **DSLPPWRCFG** register. |
| 0 | SPDERR | RO | 0 | SRAM Power Down Request Error<br><br>Value Description<br>0 No error.<br>1 An error has occurred because software has requested an SRAM power down mode that is not available using the SRAMPM field in the **SLPPWRCFG** or the **DSLPPWRCFG** register. |

## Register 30: Reset Behavior Control Register (RESBEHAVCTL), offset 0x1D8

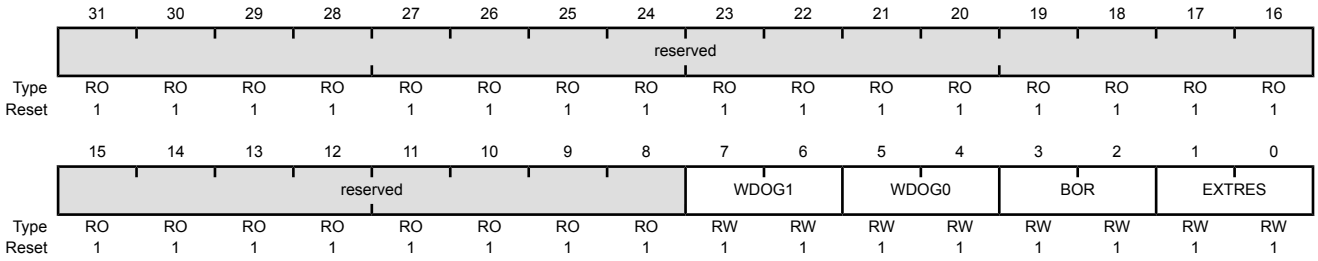
The Reset Behavior Control Register contains system management controls.

The **RESBEHAVCTL** register effect occurs immediately when the register is changed. The next power-on reset sequence returns the reset value.

If any bit field below is set to 0x3 when a reset occurs, a simulated POR will be generated and the appropriate reset cause will be set in the **Reset Cause (RESC)** register. During a simulated POR, registers are reloaded and the bootloader is executed. If a full POR is initiated the POR bit in the **RESC** register will be set and all other bits will be cleared.

Reset Behavior Control Register (RESBEHAVCTL)

Base 0x400F.E000
Offset 0x1D8
Type RW, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0xFFFF.FF | 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. |
| 7:6 | WDOG1 | RW | 0x3 | Watchdog 1 Reset Operation<br>Value Description<br>0x0 - 0x1 Reserved. Default operation is performed.<br>0x2 Watchdog 1 issues a system reset.<br>0x3 Watchdog 1 issues a simulated POR sequence (default). |
| 5:4 | WDOG0 | RW | 0x3 | Watchdog 0 Reset Operation<br>Value Description<br>0x0 - 0x1 Reserved. Default operation is performed.<br>0x2 Watchdog 0 issues a system reset.<br>0x3 Watchdog 0 issues a simulated POR sequence (default). |

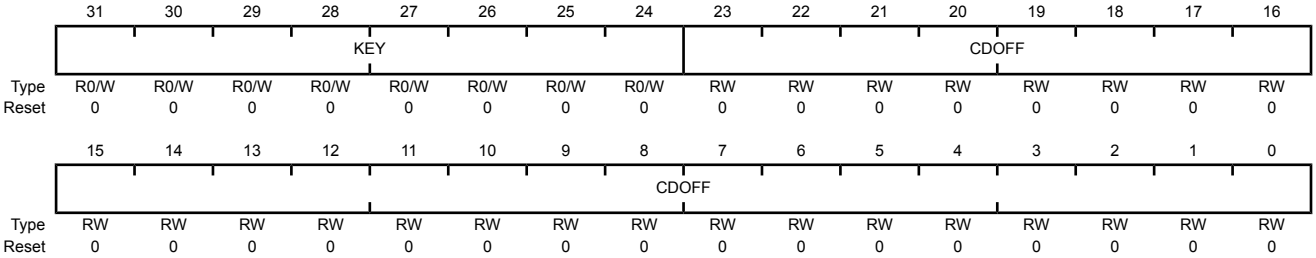
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | BOR | RW | 0x3 | BOR Reset operation<br>This field defines operation of BOR when the USER has defined the BOR operation to be a reset.<br>**Note:** If the BOR operation is defined as an interrupt, this setting has no effect.<br>Value: Description<br>0x0 - 0x1: Reserved. Default operation is performed.<br>0x2: Brown Out Reset issues system reset.<br>0x3: Brown Out Reset issues a simulated POR sequence (default). |
| 1:0 | EXTRES | RW | 0x3 | External $\overline{\text{RST}}$ Pin Operation<br>Value: Description<br>0x0 - 0x1: Reserved. Default operation is performed.<br>0x2: External $\overline{\text{RST}}$ assertion issues a system reset.<br>0x3: External $\overline{\text{RST}}$ assertion issues a simulated POR sequence (default). |

## Register 31: Hardware System Service Request (HSSR), offset 0x1F4

The **HSSR** register is used to control system configuration functions, such as Return-to-Factory settings. A write to the **HSSR** register stores a command descriptor pointer (CDOFF) value if the KEY field is correct (0xCA). A successful write to this register also initiates a system reset. The initialization process executes before examining the **HSSR** register and processing the command. This register can only be accessed in privilege mode. Refer to “Hardware System Service Request” on page 256 for more information on how to use the **HSSR** register.

Hardware System Service Request (HSSR)

Base 0x400F.E000
Offset 0x1F4
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | KEY | R0/W | 0 | Write Key<br>When read, this field returns zero.<br>When written, this field must contain the value of 0xCA in order to register the CDOFF field and initiate a HSSR request. Writes with KEY values other than 0xCA are ignored. |
| 23:0 | CDOFF | RW | 0 | Command Descriptor Pointer<br>This field contains either the status result from the previous HSSR request, or it contains a (word-aligned) memory address where the command descriptor is located.<br>If CDOFF = 0x00.0000, it indicates there is no request and no prcoessing is performed.<br>If CDOFF = 0xFF.FFFF, it indicates that the previous request through HSSR did not complete due to an error.<br>Otherwise, CDOFF contains the offset for a data structure in SRAM. |

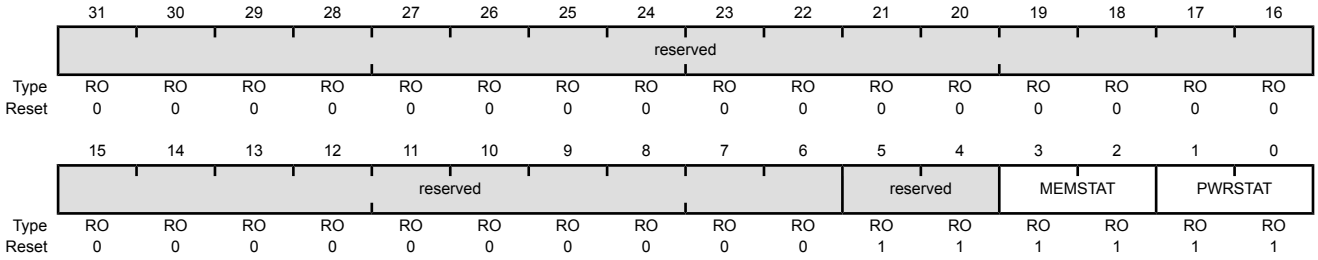
## Register 32: USB Power Domain Status (USBPDS), offset 0x280

This register provides the status of power to the USB SRAM memory array.

**Note:** If the **USBMPC** register's PWRCTL field is set to 0x3 and the power domain to the USB is turned off by writing a 0 to the P0 bit of the **PCUSB** register, then the SRAM memory goes into retention and the MEMSTAT field of the **USBPDS** register reads as 0x1 (retention).

USB Power Domain Status (USBPDS)

Base 0x400F.E000
Offset 0x280
Type RO, reset 0x0000.003F




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0 | 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 | reserved | RO | 0x3 | 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. |
| 3:2 | MEMSTAT | RO | 0x3 | Memory Array Power Status<br>Displays status of USB SRAM memory<br><br>Value Description<br>0x0 Array OFF<br>0x1 SRAM Retention<br>0x2 Reserved<br>0x3 Array On |
| 1:0 | PWRSTAT | RO | 0x3 | Power Domain Status<br><br>Value Description<br>0x0 OFF<br>0x1-0x2 Reserved<br>0x3 ON |

## Register 33: USB Memory Power Control (USBMPC), offset 0x284

This register provides power control to the peripheral memory array.

**Note:** If the **USBMPC** register's PWRCTL field is set to 0x3 and the power domain to the USB is turned off by writing a 0 to the P0 bit of the **PCUSB** register, then the SRAM memory goes into retention and the MEMSTAT field of the **USBPDS** register reads as 0x1 (retention).

USB Memory Power Control (USBMPC)

Base 0x400F.E000
Offset 0x284
Type RW, reset 0x0000.0003

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | PWRCTL | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1:0 | PWRCTL | RW | 0x3 | Memory Array Power Control<br>Allows multiple levels of power control in peripheral's SRAM memory space<br><br>Value Description<br>0x0 Array OFF<br>0x1 SRAM Retention<br>0x2 Reserved<br>0x3 Array On |

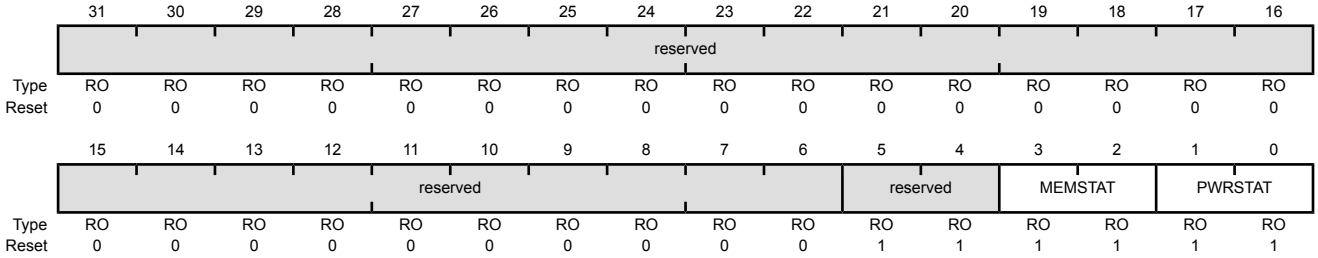
## Register 34: Ethernet MAC Power Domain Status (EMACPDS), offset 0x288

This register provides the status of power to the EMAC SRAM memory array.

**Note:** The EMAC memory array does not support retention and can only be turned ON and OFF. Memory array OFF is supported only when the power domain is off. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to the EMAC is subsequently removed by clearing the P0 bit of the **PCEMAC** register, the event causes the memory array to turn off and the MEMSTAT bit in the **EMACPDS** register to be 0x0 (array OFF).

Ethernet MAC Power Domain Status (EMACPDS)

Base 0x400F.E000
Offset 0x288
Type RO, reset 0x0000.003F




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0 | 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 | reserved | RO | 0x3 | 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. |
| 3:2 | MEMSTAT | RO | 0x3 | Memory Array Power Status<br>Displays status of EMAC SRAM memory<br>Value Description<br>0x0 Array OFF<br>0x1-0x2 Reserved<br>0x3 Array On |
| 1:0 | PWRSTAT | RO | 0x3 | Power Domain Status<br>Value Description<br>0x0 OFF<br>0x1-0x2 Reserved<br>0x3 ON |

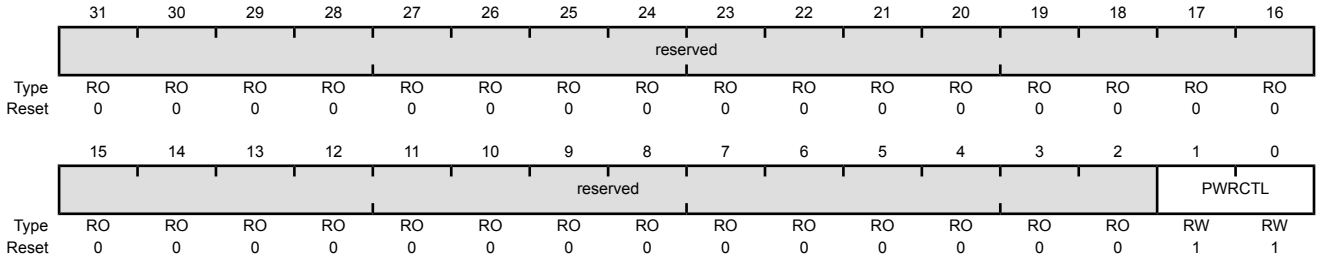
## Register 35: Ethernet MAC Memory Power Control (EMACMPC), offset 0x28C

This register provides power control to the peripheral memory array.

**Note:** The EMAC memory array does not support retention and can only be turned ON and OFF. Memory array OFF is supported only when the power domain is off. If the memory array is turned on (PWRCTL = 0x3) and the power control to the EMAC is removed by clearing the P0 bit of the **PCEMAC** register, the memory array is turned off and the MEMSTAT bit in the **EMACPDS** register is 0x0.

Ethernet MAC Memory Power Control (EMACMPC)

Base 0x400F.E000
Offset 0x28C
Type RW, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1:0 | PWRCTL | RW | 0x3 | Memory Array Power Control |

| Value | Description |
|---|---|
| 0x0 | Array OFF |
| | **Note:** Array OFF Mode is only supported when the P0 bit of the **PCEMAC** register at offset 0x99C is set to 0. |
| 0x1-0x2 | Reserved |
| 0x3 | Array On |

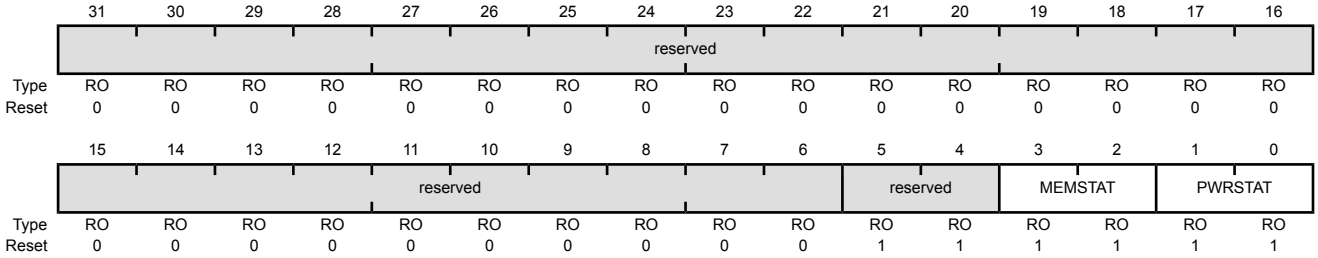
## Register 36: LCD Power Domain Status (LCDPDS), offset 0x290

This register provides the status of power to the LCD SRAM memory array.

**Note:** The LCD memory array does not support retention and can only be turned ON and OFF. If the memory array is currently turned ON (PWRCTL = 0x3) and the power control to the LCD is subsequently removed by clearing the P0 bit of the **PCLCD** register, the event causes the memory array to turn off and the MEMSTAT bit in the **LCDPDS** register to be 0x0 (array OFF).

LCD Power Domain Status (LCDPDS)

Base 0x400F.E000
Offset 0x290
Type RO, reset 0x0000.003F




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.000 | 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 | reserved | RO | 0x3 | 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. |
| 3:2 | MEMSTAT | RO | 0x3 | Memory Array Power Status<br>Displays status of LCD SRAM memory<br>Value Description<br>0x0 Array OFF<br>0x1-0x2 Reserved<br>0x3 Array On |
| 1:0 | PWRSTAT | RO | 0x3 | Power Domain Status<br>Value Description<br>0x0 OFF<br>0x1-0x2 Reserved<br>0x3 ON |

## Register 37: LCD Memory Power Control (LCDMPC), offset 0x294

This register provides power control to the peripheral memory array.

**Note:** The LCD memory array does not support retention and can only be turned ON and OFF. If the memory array is currently turned ON (PWRCTL = 0x3) and the power control to the LCD is subsequently removed by clearing the P0 bit of the **PCLCD** register, the event causes the memory array to turn off and the MEMSTAT bit in the **LCDPDS** register to be 0x0 (array OFF).

LCD Memory Power Control (LCDMPC)

Base 0x400F.E000
Offset 0x294
Type RW, reset 0x0000.0003

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | PWRCTL | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.0000 | 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. |
| 1:0 | PWRCTL | RW | 0x3 | Memory Array Power Control<br>Allows multiple levels of power control in peripheral's SRAM memory space |

| Value | Description |
|---|---|
| 0x0 | Array OFF |
| 0x1-0x2 | Reserved |
| 0x3 | Array On |

## Register 38: CAN 0 Power Domain Status (CAN0PDS), offset 0x298

This register provides the status of power to the CAN0 SRAM memory array.

**Note:** The CAN0 memory array does not support retention and can only be turned ON and OFF. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to the CAN0 is subsequently removed by clearing the P0 bit of the **PCCAN** register, the event causes the memory array to turn off and the MEMSTAT bit in the **CAN0PDS** register to be 0x0 (array OFF).

CAN 0 Power Domain Status (CAN0PDS)

Base 0x400F.E000
Offset 0x298
Type RO, reset 0x0000.003F

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | reserved | | MEMSTAT | | PWRSTAT | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0 | 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 | reserved | RO | 0x3 | 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. |
| 3:2 | MEMSTAT | RO | 0x3 | Memory Array Power Status<br>Displays status of the CAN0 SRAM memory<br>Value Description<br>0x0 Array OFF<br>0x1-0x2 Reserved<br>0x3 Array On |
| 1:0 | PWRSTAT | RO | 0x3 | Power Domain Status<br>Value Description<br>0x0 OFF<br>0x1-0x2 Reserved<br>0x3 ON |

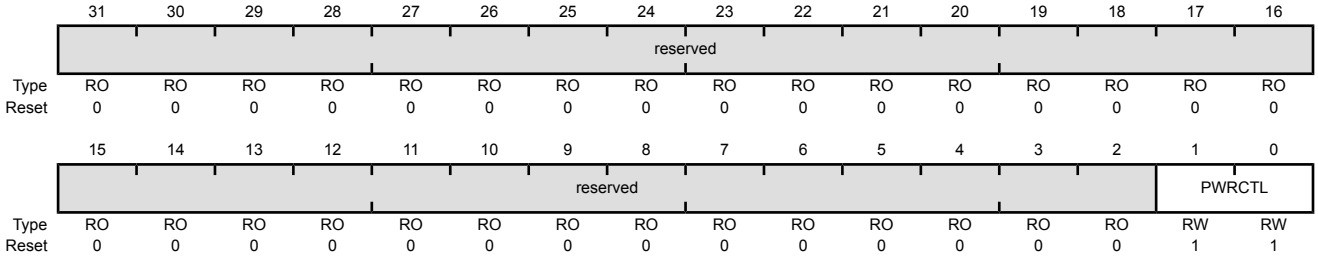
## Register 39: CAN 0 Memory Power Control (CAN0MPC), offset 0x29C

This register provides power control to the peripheral memory array.

**Note:** The CAN0 memory array does not support retention and can only be turned ON and OFF. If the memory array is currently turned ON (PWRCTL = 0x3) and the power control to the CAN0 is subsequently removed by clearing the P0 bit of the **PCCAN** register, the event causes the memory array to turn off and the MEMSTAT bit in the **CAN0PDS** register to be 0x0 (array OFF).

CAN 0 Memory Power Control (CAN0MPC)

Base 0x400F.E000
Offset 0x29C
Type RW, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1:0 | PWRCTL | RW | 0x3 | Memory Array Power Control<br>Allows multiple levels of power control in peripheral's SRAM memory space<br><br>Value Description<br>0x0 Array OFF<br>0x1-0x2 Reserved<br>0x3 Array On |

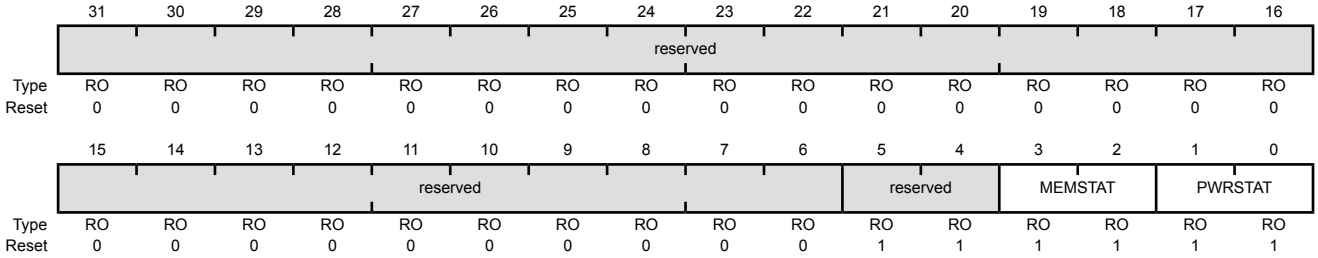
## Register 40: CAN 1 Power Domain Status (CAN1PDS), offset 0x2A0

This register provides the status of power to the CAN 1 SRAM memory array.

**Note:** The CAN1 memory array does not support retention and can only be turned ON and OFF. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to CAN1 is subsequently removed by clearing the P1 bit of the **PCCAN** register, the event causes the memory array to turn off and the MEMSTAT bit in the **CAN1PDS** register to be 0x0 (array OFF).

CAN 1 Power Domain Status (CAN1PDS)

Base 0x400F.E000
Offset 0x2A0
Type RO, reset 0x0000.003F




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0 | 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 | reserved | RO | 0x3 | 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. |
| 3:2 | MEMSTAT | RO | 0x3 | Memory Array Power Status<br>Displays status of CAN1 SRAM memory<br>Value Description<br>0x0 Array OFF<br>0x1-0x2 Reserved<br>0x3 Array On |
| 1:0 | PWRSTAT | RO | 0x3 | Power Domain Status<br>Value Description<br>0x0 OFF<br>0x1-0x2 Reserved<br>0x3 ON |

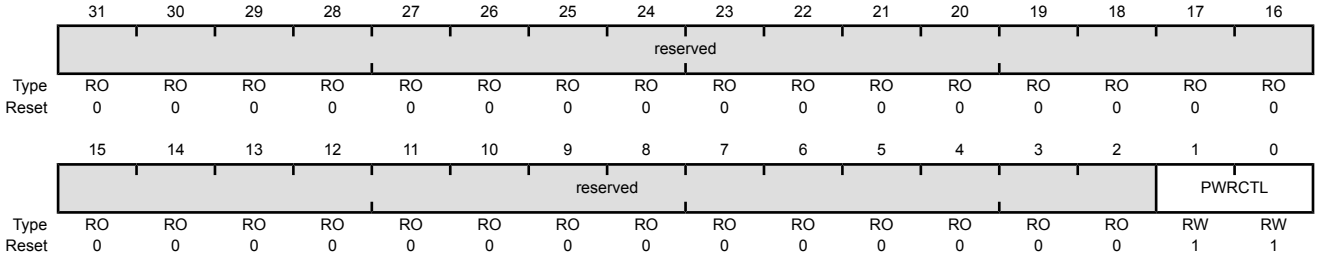
## Register 41: CAN 1 Memory Power Control (CAN1MPC), offset 0x2A4

This register provides power control to the peripheral memory array.

**Note:** The CAN1 memory array does not support retention and can only be turned ON and OFF. If the memory array is currently turned on (PWRCTL = 0x3) and the power control to CAN1 is subsequently removed by clearing the P1 bit of the **PCCAN** register, the event causes the memory array to turn off and the MEMSTAT bit in the **CAN1PDS** register to be 0x0 (array OFF).

CAN 1 Memory Power Control (CAN1MPC)

Base 0x400F.E000
Offset 0x2A4
Type RW, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | 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. |
| 1:0 | PWRCTL | RW | 0x3 | Memory Array Power Control<br>Allows multiple levels of power control in peripheral's SRAM memory space |

| Value | Description |
|---|---|
| 0x0 | Array OFF |
| 0x1-0x2 | Reserved |
| 0x3 | Array On |

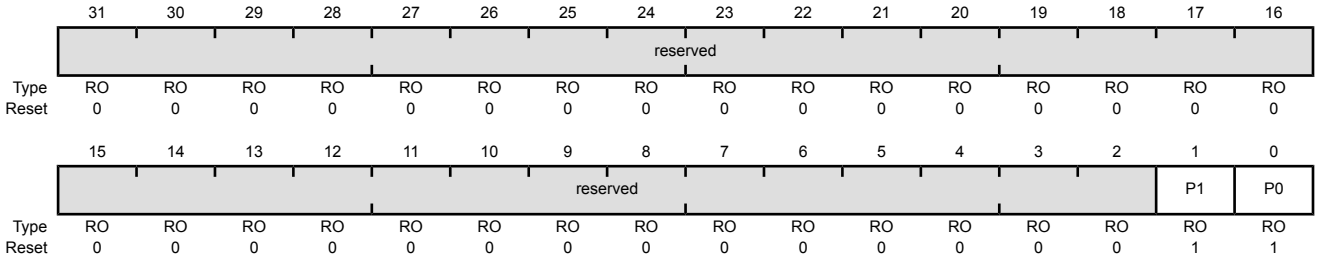
## Register 42: Watchdog Timer Peripheral Present (PPWD), offset 0x300

The **PPWD** register provides software information regarding the watchdog modules.

**Important:** This register should be used to determine which watchdog timers are implemented on this microcontroller.

Watchdog Timer Peripheral Present (PPWD)

Base 0x400F.E000
Offset 0x300
Type RO, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | P1 | RO | 0x1 | Watchdog Timer 1 Present<br>Value Description<br>0 Watchdog module 1 is not present.<br>1 Watchdog module 1 is present. |
| 0 | P0 | RO | 0x1 | Watchdog Timer 0 Present<br>Value Description<br>0 Watchdog module 0 is not present.<br>1 Watchdog module 0 is present. |

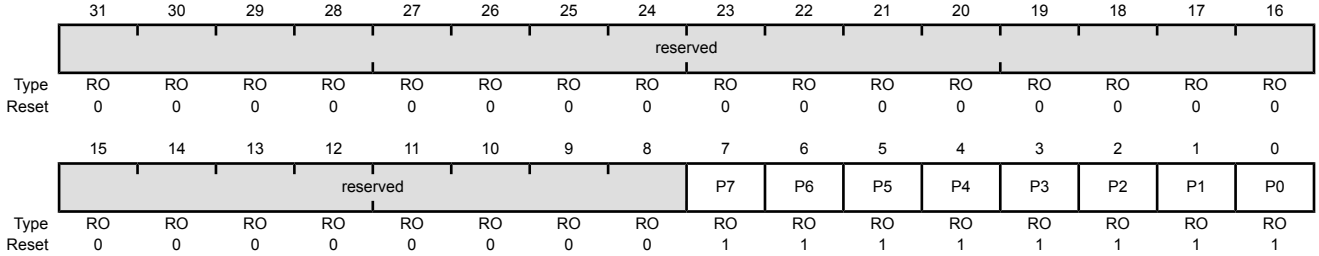
## Register 43: 16/32-Bit General-Purpose Timer Peripheral Present (PPTIMER), offset 0x304

The **PPTIMER** register provides software information regarding the 16/32-bit general-purpose timer modules.

**Important:** This register should be used to determine which timers are implemented on this microcontroller.

16/32-Bit General-Purpose Timer Peripheral Present (PPTIMER)

Base 0x400F.E000
Offset 0x304
Type RO, reset 0x0000.00FF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | P7 | RO | 0x1 | 16/32-Bit General-Purpose Timer 7 Present<br>Value Description<br>0 16/32-bit general-purpose timer module 7 is not present.<br>1 16/32-bit general-purpose timer module 7 is present. |
| 6 | P6 | RO | 0x1 | 16/32-Bit General-Purpose Timer 6 Present<br>Value Description<br>0 16/32-bit general-purpose timer module 6 is not present.<br>1 16/32-bit general-purpose timer module 6 is present. |
| 5 | P5 | RO | 0x1 | 16/32-Bit General-Purpose Timer 5 Present<br>Value Description<br>0 16/32-bit general-purpose timer module 5 is not present.<br>1 16/32-bit general-purpose timer module 5 is present. |
| 4 | P4 | RO | 0x1 | 16/32-Bit General-Purpose Timer 4 Present<br>Value Description<br>0 16/32-bit general-purpose timer module 4 is not present.<br>1 16/32-bit general-purpose timer module 4 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | P3 | RO | 0x1 | 16/32-Bit General-Purpose Timer 3 Present<br><br>Value Description<br>0 16/32-bit general-purpose timer module 3 is not present.<br>1 16/32-bit general-purpose timer module 3 is present. |
| 2 | P2 | RO | 0x1 | 16/32-Bit General-Purpose Timer 2 Present<br><br>Value Description<br>0 16/32-bit general-purpose timer module 2 is not present.<br>1 16/32-bit general-purpose timer module 2 is present. |
| 1 | P1 | RO | 0x1 | 16/32-Bit General-Purpose Timer 1 Present<br><br>Value Description<br>0 16/32-bit general-purpose timer module 1 is not present.<br>1 16/32-bit general-purpose timer module 1 is present. |
| 0 | P0 | RO | 0x1 | 16/32-Bit General-Purpose Timer 0 Present<br><br>Value Description<br>0 16/32-bit general-purpose timer module 0 is not present.<br>1 16/32-bit general-purpose timer module 0 is present. |

## Register 44: General-Purpose Input/Output Peripheral Present (PPGPIO), offset 0x308

The **PPGPIO** register provides software information regarding the general-purpose input/output modules.

**Important:** This register should be used to determine which GPIO ports are implemented on this microcontroller.

General-Purpose Input/Output Peripheral Present (PPGPIO)

Base 0x400F.E000
Offset 0x308
Type RO, reset 0x0003.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | P17 | P16 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | P17 | RO | 0x1 | GPIO Port T Present<br>Value Description<br>0 GPIO Port T is not present.<br>1 GPIO Port T is present. |
| 16 | P16 | RO | 0x1 | GPIO Port S Present<br>Value Description<br>0 GPIO Port S is not present.<br>1 GPIO Port S is present. |
| 15 | P15 | RO | 0x1 | GPIO Port R Present<br>Value Description<br>0 GPIO Port R is not present.<br>1 GPIO Port R is present. |
| 14 | P14 | RO | 0x1 | GPIO Port Q Present<br>Value Description<br>0 GPIO Port Q is not present.<br>1 GPIO Port Q is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | P13 | RO | 0x1 | GPIO Port P Present<br><br>Value Description<br>0 GPIO Port P is not present.<br>1 GPIO Port P is present. |
| 12 | P12 | RO | 0x1 | GPIO Port N Present<br><br>Value Description<br>0 GPIO Port N is not present.<br>1 GPIO Port N is present. |
| 11 | P11 | RO | 0x1 | GPIO Port M Present<br><br>Value Description<br>0 GPIO Port M is not present.<br>1 GPIO Port M is present. |
| 10 | P10 | RO | 0x1 | GPIO Port L Present<br><br>Value Description<br>0 GPIO Port L is not present.<br>1 GPIO Port L is present. |
| 9 | P9 | RO | 0x1 | GPIO Port K Present<br><br>Value Description<br>0 GPIO Port K is not present.<br>1 GPIO Port K is present. |
| 8 | P8 | RO | 0x1 | GPIO Port J Present<br><br>Value Description<br>0 GPIO Port J is not present.<br>1 GPIO Port J is present. |
| 7 | P7 | RO | 0x1 | GPIO Port H Present<br><br>Value Description<br>0 GPIO Port H is not present.<br>1 GPIO Port H is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | P6 | RO | 0x1 | GPIO Port G Present<br>Value Description<br>0 GPIO Port G is not present.<br>1 GPIO Port G is present. |
| 5 | P5 | RO | 0x1 | GPIO Port F Present<br>Value Description<br>0 GPIO Port F is not present.<br>1 GPIO Port F is present. |
| 4 | P4 | RO | 0x1 | GPIO Port E Present<br>Value Description<br>0 GPIO Port E is not present.<br>1 GPIO Port E is present. |
| 3 | P3 | RO | 0x1 | GPIO Port D Present<br>Value Description<br>0 GPIO Port D is not present.<br>1 GPIO Port D is present. |
| 2 | P2 | RO | 0x1 | GPIO Port C Present<br>Value Description<br>0 GPIO Port C is not present.<br>1 GPIO Port C is present. |
| 1 | P1 | RO | 0x1 | GPIO Port B Present<br>Value Description<br>0 GPIO Port B is not present.<br>1 GPIO Port B is present. |
| 0 | P0 | RO | 0x1 | GPIO Port A Present<br>Value Description<br>0 GPIO Port A is not present.<br>1 GPIO Port A is present. |

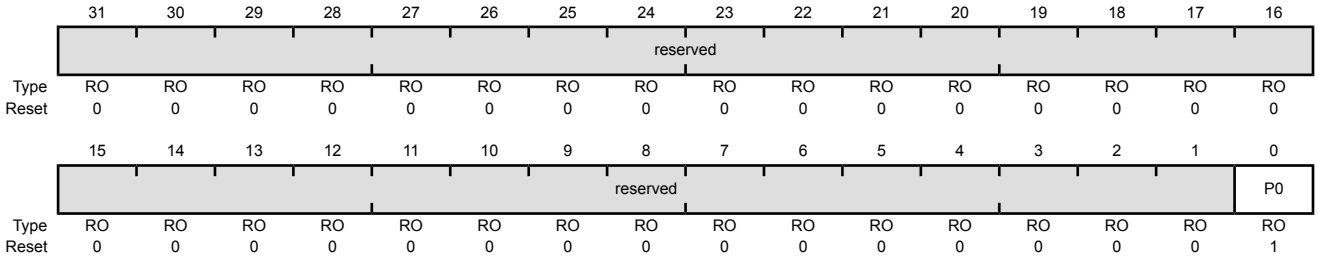
## Register 45: Micro Direct Memory Access Peripheral Present (PPDMA), offset 0x30C

The **PPDMA** register provides software information regarding the µDMA module.

**Important:** This register should be used to determine if the µDMA module is implemented on this microcontroller.

Micro Direct Memory Access Peripheral Present (PPDMA)

Base 0x400F.E000
Offset 0x30C
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | µDMA Module Present<br><br>Value Description<br>0 µDMA module is not present.<br>1 µDMA module is present. |

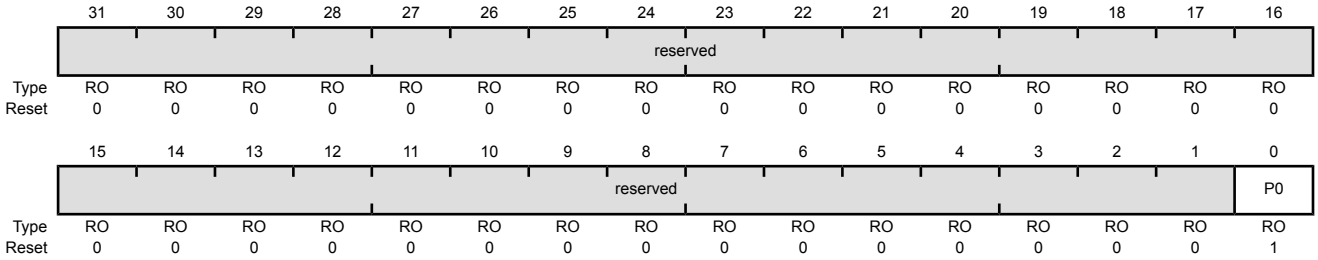
## Register 46: EPI Peripheral Present (PPEPI), offset 0x310

The **PPEPI** register provides software information regarding the EPI module.

**Important:** This register should be used to determine if the EPI module is implemented on this microcontroller.

EPI Peripheral Present (PPEPI)

Base 0x400F.E000
Offset 0x310
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | EPI Module Present<br>Value Description<br>0 EPI module is not present.<br>1 EPI module is present. |

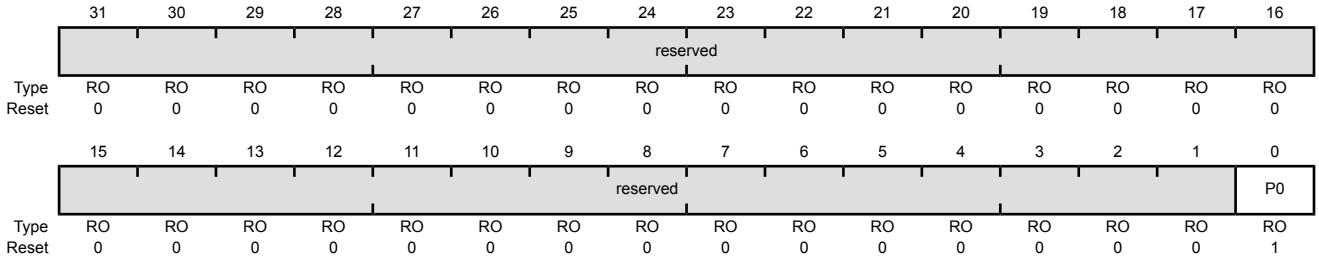
## Register 47: Hibernation Peripheral Present (PPHIB), offset 0x314

The **PPHIB** register provides software information regarding the Hibernation module.

**Important:** This register should be used to determine if the Hibernation module is implemented on this microcontroller.

Hibernation Peripheral Present (PPHIB)

Base 0x400F.E000
Offset 0x314
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | Hibernation Module Present<br>Value Description<br>0 Hibernation module is not present.<br>1 Hibernation module is present. |

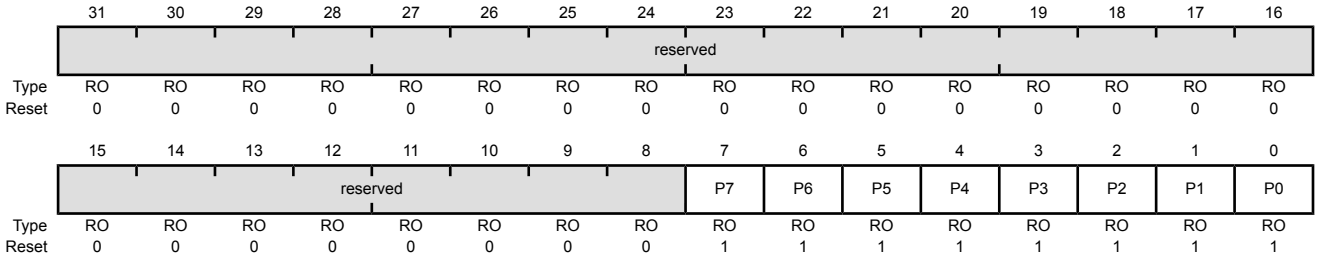
## Register 48: Universal Asynchronous Receiver/Transmitter Peripheral Present (PPUART), offset 0x318

The **PPUART** register provides software information regarding the UART modules.

**Important:** This register should be used to determine which UART modules are implemented on this microcontroller.

Universal Asynchronous Receiver/Transmitter Peripheral Present (PPUART)

Base 0x400F.E000
Offset 0x318
Type RO, reset 0x0000.00FF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | P7 | RO | 0x1 | UART Module 7 Present<br>Value Description<br>0 UART module 7 is not present.<br>1 UART module 7 is present. |
| 6 | P6 | RO | 0x1 | UART Module 6 Present<br>Value Description<br>0 UART module 6 is not present.<br>1 UART module 6 is present. |
| 5 | P5 | RO | 0x1 | UART Module 5 Present<br>Value Description<br>0 UART module 5 is not present.<br>1 UART module 5 is present. |
| 4 | P4 | RO | 0x1 | UART Module 4 Present<br>Value Description<br>0 UART module 4 is not present.<br>1 UART module 4 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | P3 | RO | 0x1 | UART Module 3 Present<br><br>Value Description<br>0 UART module 3 is not present.<br>1 UART module 3 is present. |
| 2 | P2 | RO | 0x1 | UART Module 2 Present<br><br>Value Description<br>0 UART module 2 is not present.<br>1 UART module 2 is present. |
| 1 | P1 | RO | 0x1 | UART Module 1 Present<br><br>Value Description<br>0 UART module 1 is not present.<br>1 UART module 1 is present. |
| 0 | P0 | RO | 0x1 | UART Module 0 Present<br><br>Value Description<br>0 UART module 0 is not present.<br>1 UART module 0 is present. |

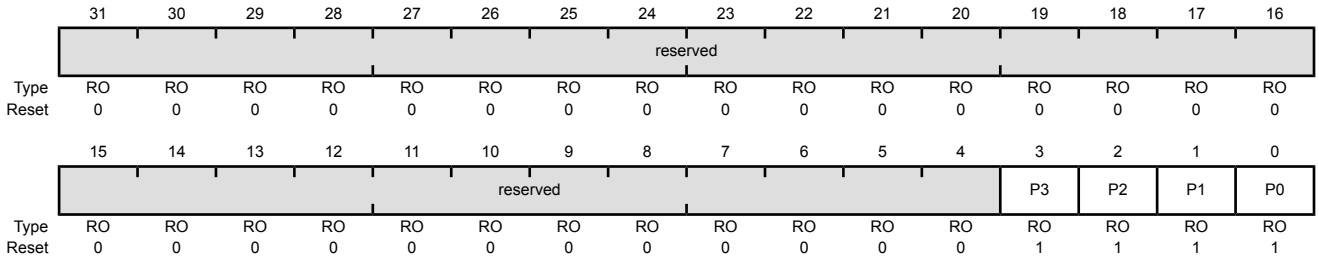
## Register 49: Synchronous Serial Interface Peripheral Present (PPSSI), offset 0x31C

The **PPSSI** register provides software information regarding the SSI modules.

**Important:** This register should be used to determine which SSI modules are implemented on this microcontroller. However, to support legacy software, the **DC2** register is available. A read of the **DC2** register correctly identifies if a legacy SSI module is present. Software must use this register to determine if a module that is not supported by the **DC2** register is present.

Synchronous Serial Interface Peripheral Present (PPSSI)

Base 0x400F.E000
Offset 0x31C
Type RO, reset 0x0000.000F




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | P3 | RO | 0x1 | SSI Module 3 Present<br>Value Description<br>0 SSI module 3 is not present.<br>1 SSI module 3 is present. |
| 2 | P2 | RO | 0x1 | SSI Module 2 Present<br>Value Description<br>0 SSI module 2 is not present.<br>1 SSI module 2 is present. |
| 1 | P1 | RO | 0x1 | SSI Module 1 Present<br>Value Description<br>0 SSI module 1 is not present.<br>1 SSI module 1 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RO | 0x1 | SSI Module 0 Present |

| Value | Description |
|---|---|
| 0 | SSI module 0 is not present. |
| 1 | SSI module 0 is present. |

## Register 50: Inter-Integrated Circuit Peripheral Present (PPI2C), offset 0x320

The **PPI2C** register provides software information regarding the I²C modules.

**Important:** This register should be used to determine which I²C modules are implemented on this microcontroller.

Inter-Integrated Circuit Peripheral Present (PPI2C)

Base 0x400F.E000
Offset 0x320
Type RO, reset 0x0000.03FF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | P9 | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |
| 9 | P9 | RO | 0x1 | I²C Module 9 Present<br>Value Description<br>0 I²C module 9 is not present.<br>1 I²C module 9 is present. |
| 8 | P8 | RO | 0x1 | I²C Module 8 Present<br>Value Description<br>0 I²C module 8 is not present.<br>1 I²C module 8 is present. |
| 7 | P7 | RO | 0x1 | I²C Module 7 Present<br>Value Description<br>0 I²C module 7 is not present.<br>1 I²C module 7 is present. |
| 6 | P6 | RO | 0x1 | I²C Module 6 Present<br>Value Description<br>0 I²C module 6 is not present.<br>1 I²C module 6 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | P5 | RO | 0x1 | $I^2C$ Module 5 Present<br>Value Description<br>0 $I^2C$ module 5 is not present.<br>1 $I^2C$ module 5 is present. |
| 4 | P4 | RO | 0x1 | $I^2C$ Module 4 Present<br>Value Description<br>0 $I^2C$ module 4 is not present.<br>1 $I^2C$ module 4 is present. |
| 3 | P3 | RO | 0x1 | $I^2C$ Module 3 Present<br>Value Description<br>0 $I^2C$ module 3 is not present.<br>1 $I^2C$ module 3 is present. |
| 2 | P2 | RO | 0x1 | $I^2C$ Module 2 Present<br>Value Description<br>0 $I^2C$ module 2 is not present.<br>1 $I^2C$ module 2 is present. |
| 1 | P1 | RO | 0x1 | $I^2C$ Module 1 Present<br>Value Description<br>0 $I^2C$ module 1 is not present.<br>1 $I^2C$ module 1 is present. |
| 0 | P0 | RO | 0x1 | $I^2C$ Module 0 Present<br>Value Description<br>0 $I^2C$ module 0 is not present.<br>1 $I^2C$ module 0 is present. |

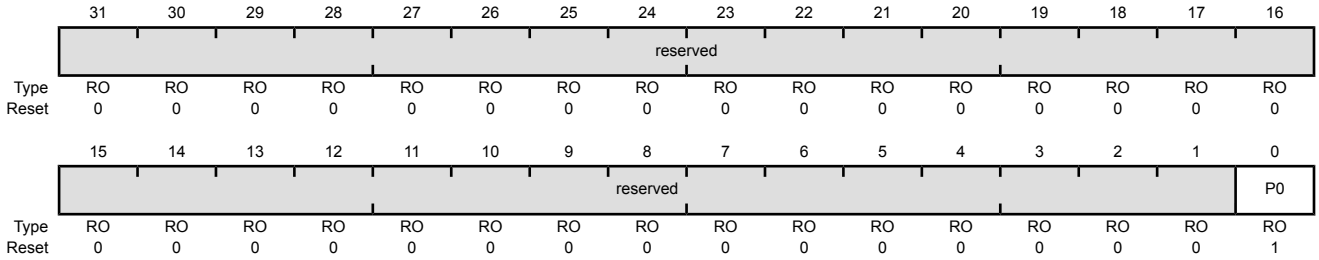
## Register 51: Universal Serial Bus Peripheral Present (PPUSB), offset 0x328

The **PPUSB** register provides software information regarding the USB module.

**Important:** This register should be used to determine if the USB module is implemented on this microcontroller.

Universal Serial Bus Peripheral Present (PPUSB)

Base 0x400F.E000
Offset 0x328
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | USB Module Present<br>Value Description<br>0 USB module is not present.<br>1 USB module is present. |

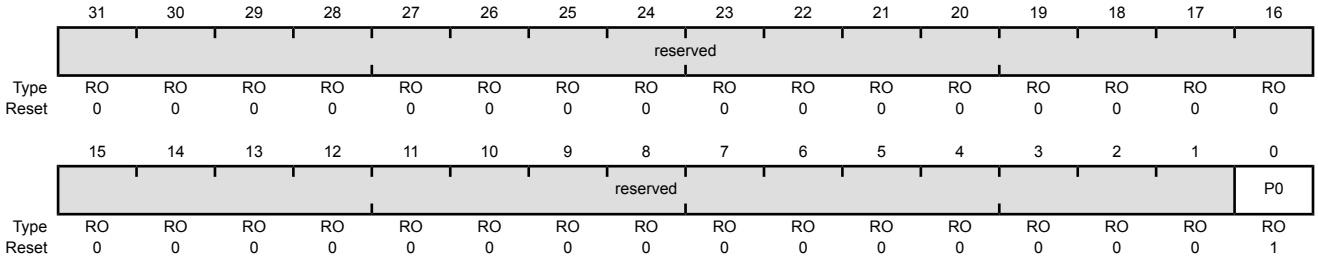
## Register 52: Ethernet PHY Peripheral Present (PPEPHY), offset 0x330

The **PPEPHY** register provides software information regarding the Ethernet PHY module.

**Important:** This register should be used to determine if the Ethernet PHY module is implemented on this microcontroller.

Ethernet PHY Peripheral Present (PPEPHY)

Base 0x400F.E000
Offset 0x330
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | Ethernet PHY Module Present |

| Value | Description |
|---|---|
| 0 | Ethernet PHY module is not present. |
| 1 | Ethernet PHY module is present. |

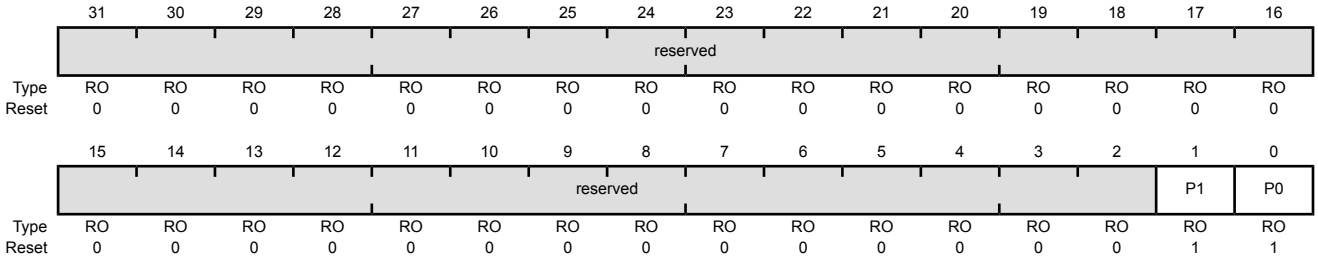
## Register 53: Controller Area Network Peripheral Present (PPCAN), offset 0x334

The **PPCAN** register provides software information regarding the CAN modules.

**Important:** This register should be used to determine which CAN modules are implemented on this microcontroller.

Controller Area Network Peripheral Present (PPCAN)

Base 0x400F.E000
Offset 0x334
Type RO, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | P1 | RO | 0x1 | CAN Module 1 Present<br>Value Description<br>0 CAN module 1 is not present.<br>1 CAN module 1 is present. |
| 0 | P0 | RO | 0x1 | CAN Module 0 Present<br>Value Description<br>0 CAN module 0 is not present.<br>1 CAN module 0 is present. |

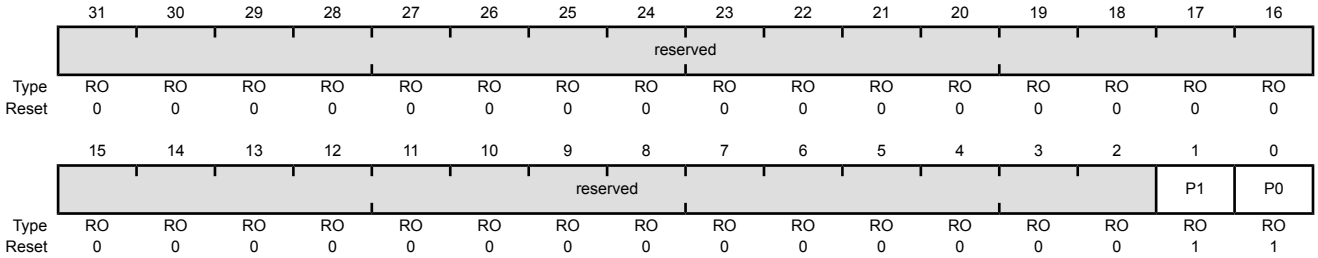
## Register 54: Analog-to-Digital Converter Peripheral Present (PPADC), offset 0x338

The **PPADC** register provides software information regarding the ADC modules.

**Important:** This register should be used to determine which ADC modules are implemented on this microcontroller.

Analog-to-Digital Converter Peripheral Present (PPADC)

Base 0x400F.E000
Offset 0x338
Type RO, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | P1 | RO | 0x1 | ADC Module 1 Present<br><br>Value Description<br>0 ADC module 1 is not present.<br>1 ADC module 1 is present. |
| 0 | P0 | RO | 0x1 | ADC Module 0 Present<br><br>Value Description<br>0 ADC module 0 is not present.<br>1 ADC module 0 is present. |

## Register 55: Analog Comparator Peripheral Present (PPACMP), offset 0x33C

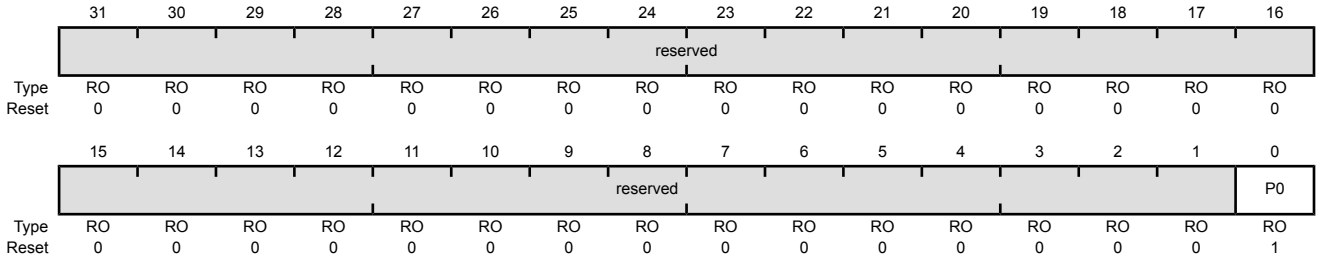
The **PPACMP** register provides software information regarding the analog comparator module.

**Important:** This register should be used to determine if the analog comparator module is implemented on this microcontroller.

Note that the **Analog Comparator Peripheral Properties (ACMPPP)** register indicates how many analog comparator blocks are included in the module.

Analog Comparator Peripheral Present (PPACMP)

Base 0x400F.E000
Offset 0x33C
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | Analog Comparator Module Present<br>Value Description<br>0 Analog comparator module is not present.<br>1 Analog comparator module is present. |

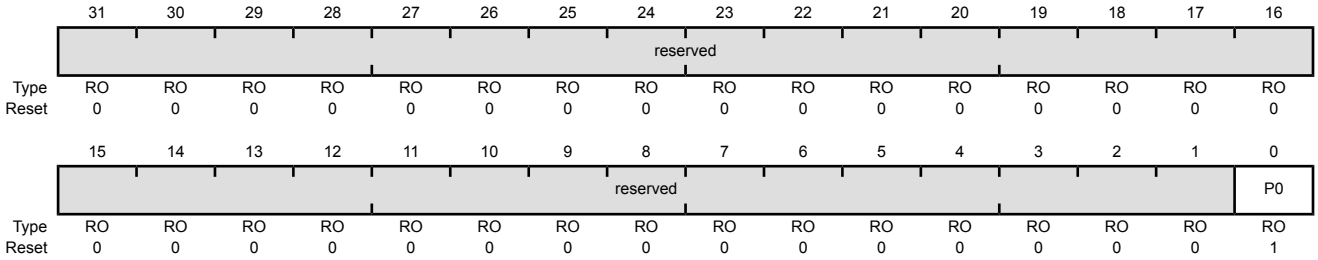
## Register 56: Pulse Width Modulator Peripheral Present (PPPWM), offset 0x340

The **PPPWM** register provides software information regarding the PWM modules.

**Important:** This register should be used to determine which PWM modules are implemented on this microcontroller.

Pulse Width Modulator Peripheral Present (PPPWM)

Base 0x400F.E000
Offset 0x340
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | PWM Module 0 Present |

| Value | Description |
|---|---|
| 0 | PWM module 0 is not present. |
| 1 | PWM module 0 is present. |

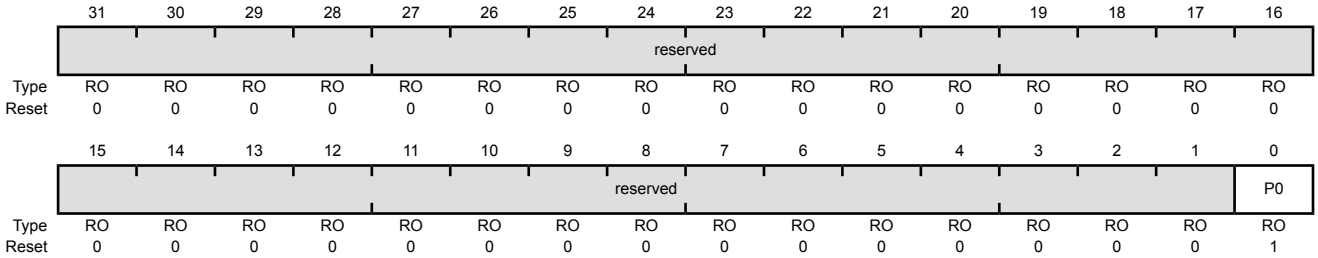
## Register 57: Quadrature Encoder Interface Peripheral Present (PPQEI), offset 0x344

The **PPQEI** register provides software information regarding the QEI modules.

**Important:** This register should be used to determine which QEI modules are implemented on this microcontroller.

Quadrature Encoder Interface Peripheral Present (PPQEI)

Base 0x400F.E000
Offset 0x344
Type RO, reset 0x0000.0001



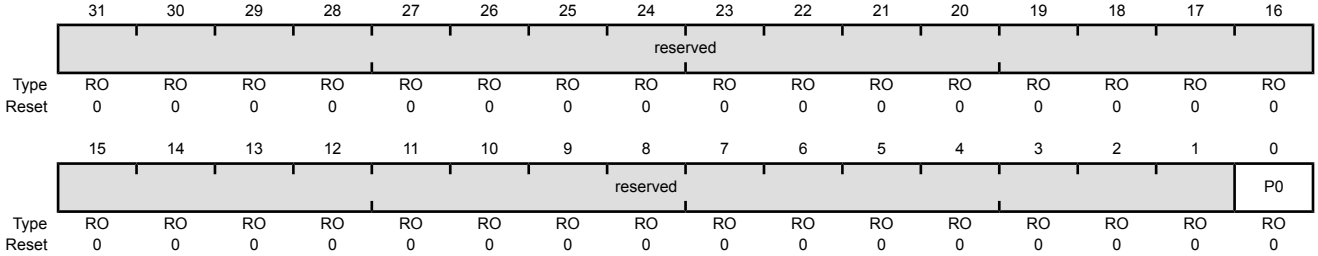

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | P0 | RO | 0x1 | QEI Module 0 Present<br>Value Description<br>0 QEI module 0 is not present.<br>1 QEI module 0 is present. |

## Register 58: Low Pin Count Interface Peripheral Present (PPLPC), offset 0x348

The **PPLPC** register provides software information regarding the LPC module.

Low Pin Count Interface Peripheral Present (PPLPC)

Base 0x400F.E000
Offset 0x348
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x0 | LPC Module Present |

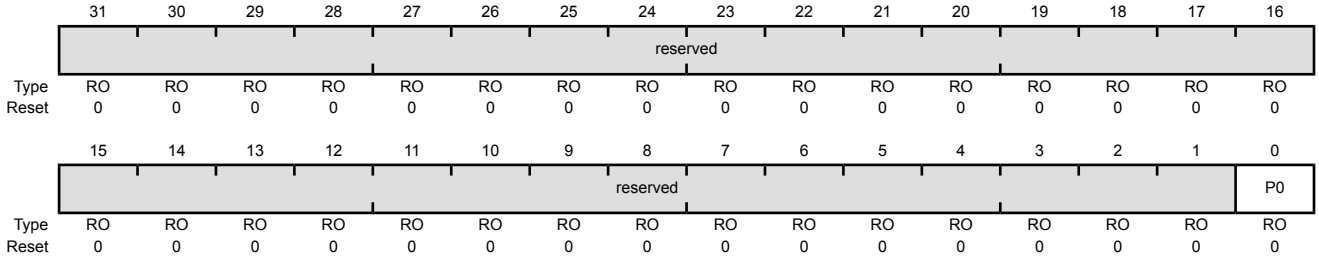
| Value | Description |
|---|---|
| 0 | LPC module is not present. |
| 1 | LPC module is present. |

## Register 59: Platform Environment Control Interface Peripheral Present (PPPECI), offset 0x350

The **PPPECI** register provides software information regarding the PECI module.

Platform Environment Control Interface Peripheral Present (PPPECI)

Base 0x400F.E000
Offset 0x350
Type RO, reset 0x0000.0000



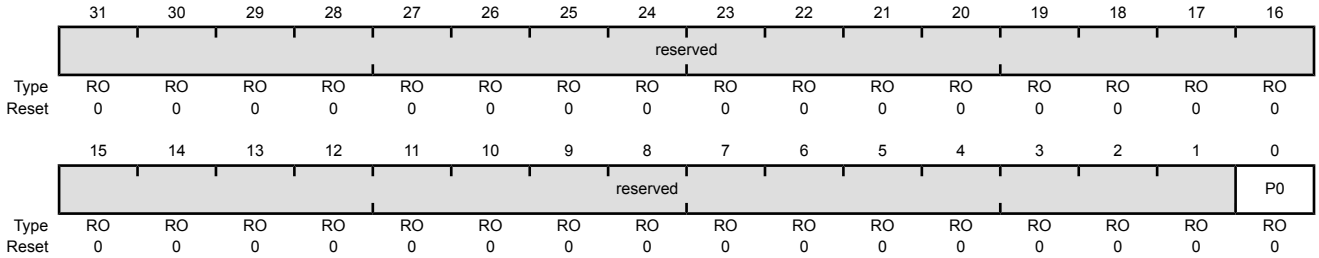

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | P0 | RO | 0x0 | PECI Module Present<br>Value Description<br>0 PECI module is not present.<br>1 PECI module is present. |

## Register 60: Fan Control Peripheral Present (PPFAN), offset 0x354

The **PPFAN** register provides software information regarding the FAN module.

Fan Control Peripheral Present (PPFAN)

Base 0x400F.E000
Offset 0x354
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x0 | FAN Module 0 Present |

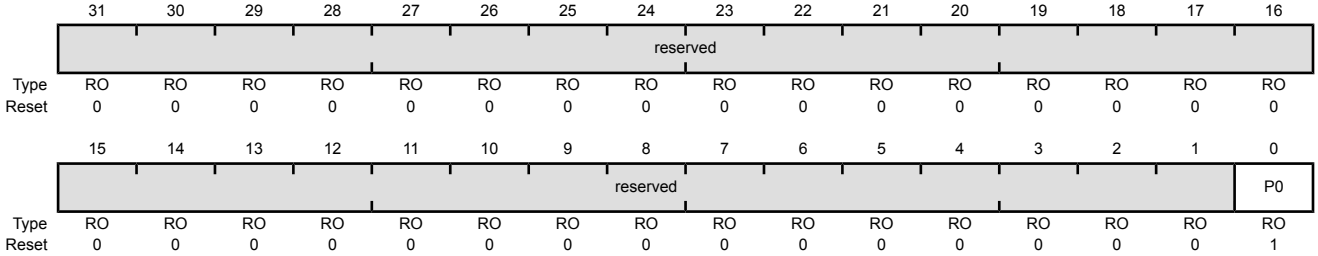
| Value | Description |
|---|---|
| 0 | FAN module is not present. |
| 1 | FAN module is present. |

## Register 61: EEPROM Peripheral Present (PPEEPROM), offset 0x358

The **PPEEPROM** register provides software information regarding the EEPROM module.

EEPROM Peripheral Present (PPEEPROM)

Base 0x400F.E000
Offset 0x358
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | EEPROM 0 Module Present |

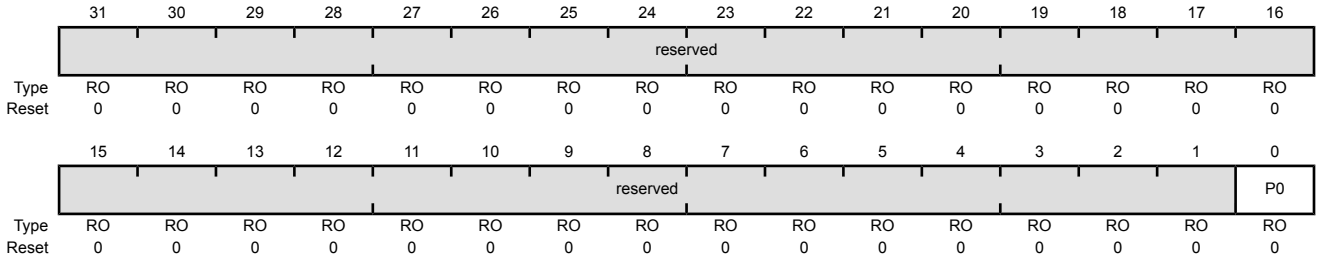
| Value | Description |
|---|---|
| 0 | EEPROM module is not present. |
| 1 | EEPROM module is present. |

## Register 62: 32/64-Bit Wide General-Purpose Timer Peripheral Present (PPWTIMER), offset 0x35C

The **PPWTIMER** register provides software information regarding the 32/64-bit wide general-purpose timer modules.

32/64-Bit Wide General-Purpose Timer Peripheral Present (PPWTIMER)

Base 0x400F.E000
Offset 0x35C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x0 | 32/64-Bit Wide General-Purpose Timer 0 Present<br>Value Description<br>0 32/64-bit wide general-purpose timer module 0 is not present.<br>1 32/64-bit wide general-purpose timer module 0 is present. |

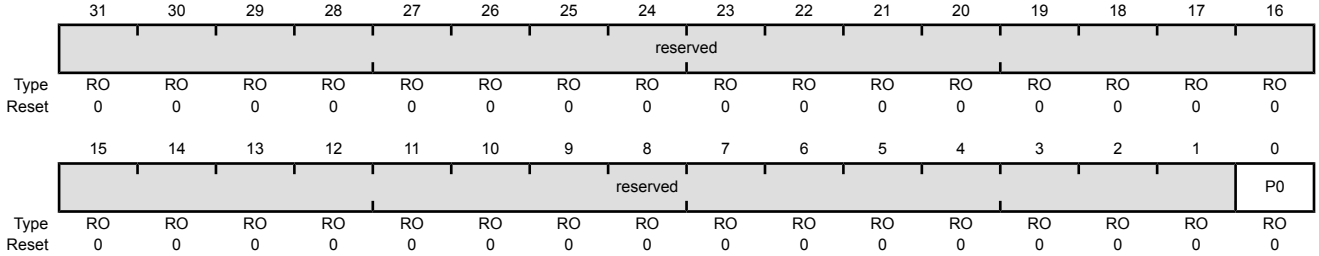
## Register 63: Remote Temperature Sensor Peripheral Present (PPRTS), offset 0x370

The **PPRTS** register provides software information regarding the Remote Temperature Sensor (RTS) module.

**Important:** This register should be used to determine which RTS modules are implemented on this microcontroller.

Remote Temperature Sensor Peripheral Present (PPRTS)

Base 0x400F.E000
Offset 0x370
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x0 | RTS Module Present<br>Value Description<br>0 RTS module is not present.<br>1 RTS module is present. |

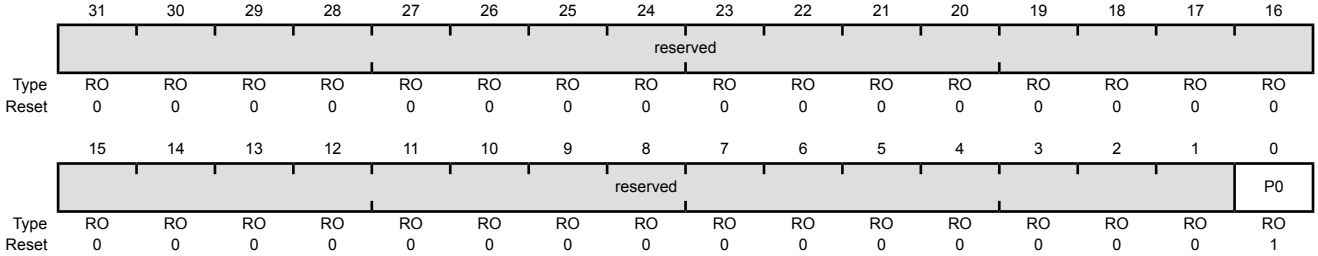
## Register 64: CRC and Cryptographic Modules Peripheral Present (PPCCM), offset 0x374

The **PPCCM** register provides software information regarding the CRC and Cryptographic Modules (AES, DES, and SHA).

**Important:** This register should be used to determine if the CRC and Cryptographic Modules (AES, DES, SHA/MD5) are implemented on this microcontroller.

CRC and Cryptographic Modules Peripheral Present (PPCCM)

Base 0x400F.E000
Offset 0x374
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | CRC and Cryptographic Modules Present<br>Value Description<br>0 The CRC, AES, DES, and SHA/MD modules are not present.<br>1 The CRC, AES, DES, and SHA/MD modules are present. |

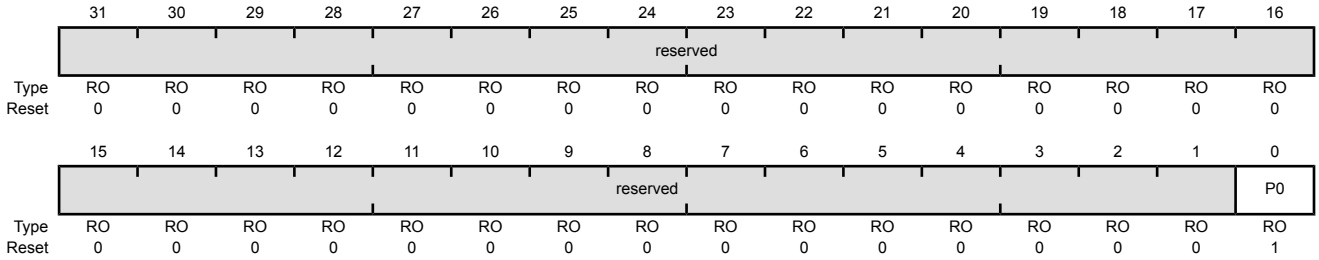
## Register 65: LCD Peripheral Present (PPLCD), offset 0x390

The **PPLCD** register provides software information regarding the LCD module.

**Important:** This register should be used to determine if an LCD controller is implemented on this microcontroller.

LCD Peripheral Present (PPLCD)

Base 0x400F.E000
Offset 0x390
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | LCD Module Present<br>Value Description<br>0 LCD module is not present.<br>1 LCD module is present. |

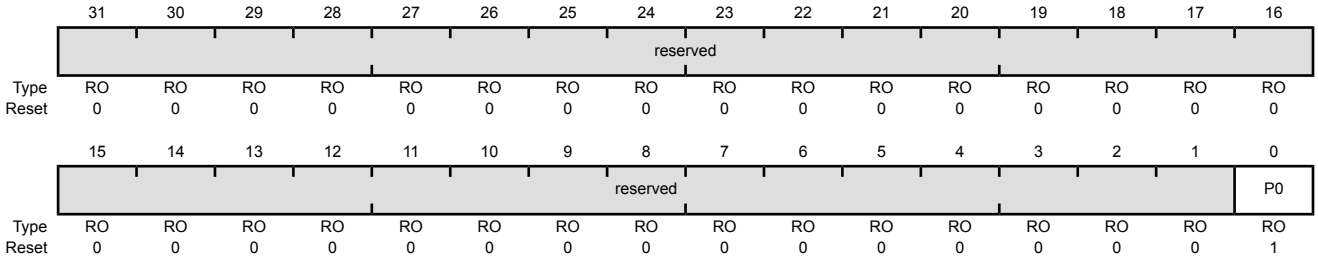
## Register 66: 1-Wire Peripheral Present (PPOWIRE), offset 0x398

The **PPOWIRE** register provides software information regarding the 1-Wire module.

**Important:** This register should be used to determine which 1-Wire modules are implemented on this microcontroller.

1-Wire Peripheral Present (PPOWIRE)

Base 0x400F.E000
Offset 0x398
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | 1-Wire Module Present<br>Value Description<br>0 1-Wire module is not present.<br>1 1-Wire module is present. |

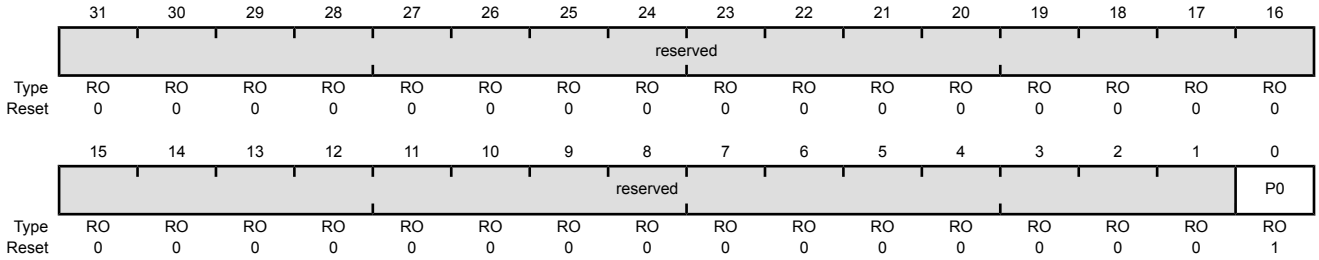
## Register 67: Ethernet MAC Peripheral Present (PPEMAC), offset 0x39C

The **PPEMAC** register provides software information regarding the Ethernet controller module.

**Important:** This register should be used to determine which Ethernet controller modules are implemented on this microcontroller.

Ethernet MAC Peripheral Present (PPEMAC)

Base 0x400F.E000
Offset 0x39C
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x1 | Ethernet Controller Module Present<br><br>Value Description<br>0 Ethernet Controller MAC module is not present.<br>1 Ethernet Controller MAC module is present. |

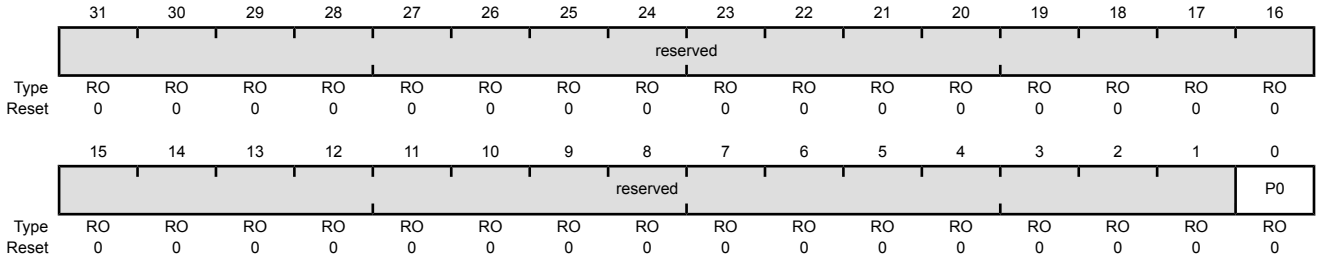
## Register 68: Power Regulator Bus Peripheral Present (PPPRB), offset 0x3A0

The **PPPRB** register provides software information regarding the Power Regulator Bus module.

**Important:** This register should be used to determine which Power Regulator Bus modules are implemented on this microcontroller.

Power Regulator Bus Peripheral Present (PPPRB)

Base 0x400F.E000
Offset 0x3A0
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x0 | PRB Module Present<br>Value Description<br>0 PRB module is not present.<br>1 PRB module is present. |

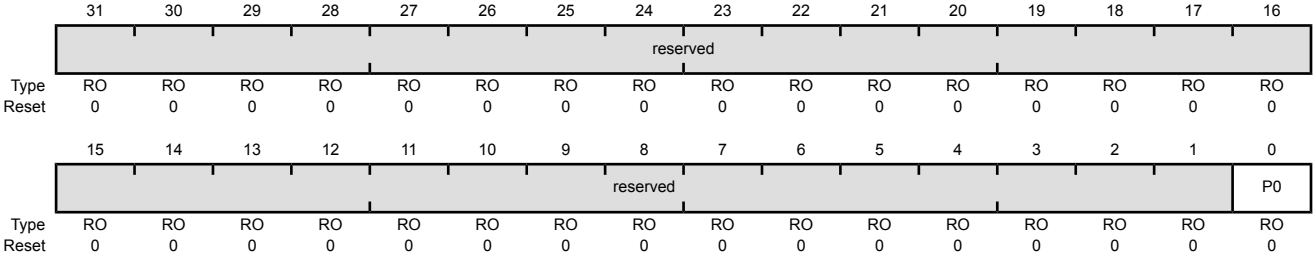
## Register 69: Human Interface Master Peripheral Present (PPHIM), offset 0x3A4

The **PPHIM** register provides software information regarding the Human Interface Master (HIM) module.

**Important:** This register should be used to determine which HIM modules are implemented on this microcontroller.

Human Interface Master Peripheral Present (PPHIM)

Base 0x400F.E000
Offset 0x3A4
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RO | 0x0 | HIM Module Present<br>Value Description<br>0 HIM module is not present.<br>1 HIM module is present. |

## Register 70: Watchdog Timer Software Reset (SRWD), offset 0x500

The **SRWD** register provides software the capability to reset the available watchdog modules.

A peripheral is reset by software using a simple two-step process:

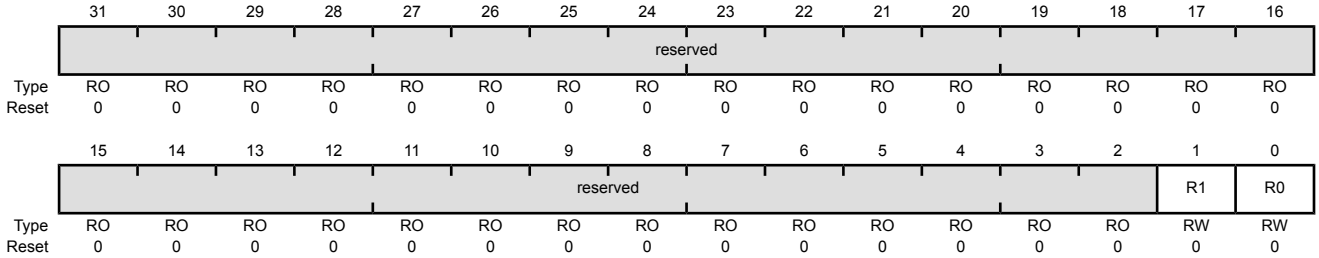
1. Software sets a bit (or bits) in the **SRWD** register. While the **SRWD** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRWD** bit.

There may be latency from the clearing of the **SRWD** bit to when the peripheral is ready for use. Software should check the corresponding **PRWD** bit to verify that the Watchdog Timer Module registers are ready to be accessed.

**Important:** This register should be used to reset the watchdog modules.

Watchdog Timer Software Reset (SRWD)

Base 0x400F.E000
Offset 0x500
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RW | 0 | Watchdog Timer 1 Software Reset<br>Value Description<br>0 Watchdog module 1 is not reset.<br>1 Watchdog module 1 is reset. |
| 0 | R0 | RW | 0 | Watchdog Timer 0 Software Reset<br>Value Description<br>0 Watchdog module 0 is not reset.<br>1 Watchdog module 0 is reset. |

## Register 71: 16/32-Bit General-Purpose Timer Software Reset (SRTIMER), offset 0x504

The **SRTIMER** register provides software the capability to reset the available 16/32-bit timer modules.

A peripheral is reset by software using a simple two-step process:

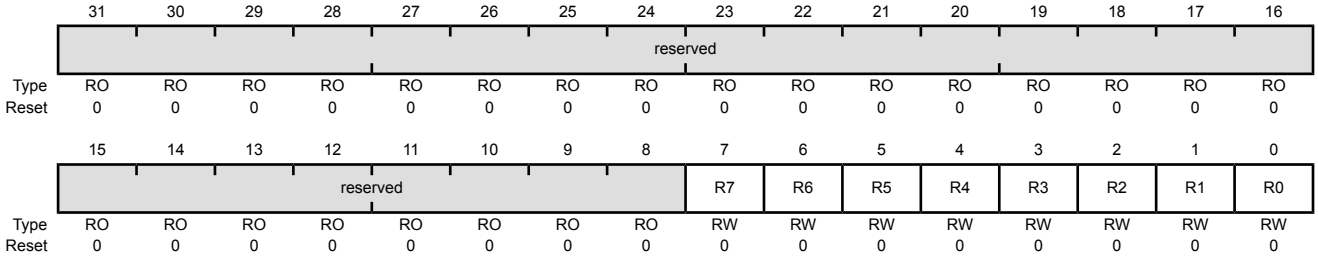
1. Software sets a bit (or bits) in the **SRTIMER** register. While the **SRTIMER** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRTIMER** bit.

There may be latency from the clearing of the **SRTIMER** bit to when the peripheral is ready for use. Software should check the corresponding **PRTIMER** bit to verify that the Timer Module registers are ready to be accessed.

**Important:** This register should be used to reset the timer modules.

16/32-Bit General-Purpose Timer Software Reset (SRTIMER)

Base 0x400F.E000
Offset 0x504
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | R7 | RW | 0 | 16/32-Bit General-Purpose Timer 7 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 7 is not reset.<br>1 16/32-bit general-purpose timer module 7 is reset. |
| 6 | R6 | RW | 0 | 16/32-Bit General-Purpose Timer 6 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 6 is not reset.<br>1 16/32-bit general-purpose timer module 6 is reset. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | R5 | RW | 0 | 16/32-Bit General-Purpose Timer 5 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 5 is not reset.<br>1 16/32-bit general-purpose timer module 5 is reset. |
| 4 | R4 | RW | 0 | 16/32-Bit General-Purpose Timer 4 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 4 is not reset.<br>1 16/32-bit general-purpose timer module 4 is reset. |
| 3 | R3 | RW | 0 | 16/32-Bit General-Purpose Timer 3 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 3 is not reset.<br>1 16/32-bit general-purpose timer module 3 is reset. |
| 2 | R2 | RW | 0 | 16/32-Bit General-Purpose Timer 2 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 2 is not reset.<br>1 16/32-bit general-purpose timer module 2 is reset. |
| 1 | R1 | RW | 0 | 16/32-Bit General-Purpose Timer 1 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 1 is not reset.<br>1 16/32-bit general-purpose timer module 1 is reset. |
| 0 | R0 | RW | 0 | 16/32-Bit General-Purpose Timer 0 Software Reset<br>Value Description<br>0 16/32-bit general-purpose timer module 0 is not reset.<br>1 16/32-bit general-purpose timer module 0 is reset. |

## Register 72: General-Purpose Input/Output Software Reset (SRGPIO), offset 0x508

The **SRGPIO** register provides software the capability to reset the available GPIO modules.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRGPIO** register. While the **SRGPIO** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRGPIO** bit.

There may be latency from the clearing of the **SRGPIO** bit to when the peripheral is ready for use. Software should check the corresponding **PRGPIO** bit to verify that the GPIO Module registers are ready to be accessed.

**Important:** This register should be used to reset the GPIO modules.

General-Purpose Input/Output Software Reset (SRGPIO)

Base 0x400F.E000
Offset 0x508
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | R17 | R16 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | R17 | RW | 0 | GPIO Port T Software Reset<br>Value Description<br>0 GPIO Port T is not reset.<br>1 GPIO Port T is reset. |
| 16 | R16 | RW | 0 | GPIO Port S Software Reset<br>Value Description<br>0 GPIO Port S is not reset.<br>1 GPIO Port S is reset. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | R15 | RW | 0 | GPIO Port R Software Reset<br>Value Description<br>0 GPIO Port R is not reset.<br>1 GPIO Port R is reset. |
| 14 | R14 | RW | 0 | GPIO Port Q Software Reset<br>Value Description<br>0 GPIO Port Q is not reset.<br>1 GPIO Port Q is reset. |
| 13 | R13 | RW | 0 | GPIO Port P Software Reset<br>Value Description<br>0 GPIO Port P is not reset.<br>1 GPIO Port P is reset. |
| 12 | R12 | RW | 0 | GPIO Port N Software Reset<br>Value Description<br>0 GPIO Port N is not reset.<br>1 GPIO Port N is reset. |
| 11 | R11 | RW | 0 | GPIO Port M Software Reset<br>Value Description<br>0 GPIO Port M is not reset.<br>1 GPIO Port M is reset. |
| 10 | R10 | RW | 0 | GPIO Port L Software Reset<br>Value Description<br>0 GPIO Port L is not reset.<br>1 GPIO Port L is reset. |
| 9 | R9 | RW | 0 | GPIO Port K Software Reset<br>Value Description<br>0 GPIO Port K is not reset.<br>1 GPIO Port K is reset. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | R8 | RW | 0 | GPIO Port J Software Reset<br>Value Description<br>0 GPIO Port J is not reset.<br>1 GPIO Port J is reset. |
| 7 | R7 | RW | 0 | GPIO Port H Software Reset<br>Value Description<br>0 GPIO Port H is not reset.<br>1 GPIO Port H is reset. |
| 6 | R6 | RW | 0 | GPIO Port G Software Reset<br>Value Description<br>0 GPIO Port G is not reset.<br>1 GPIO Port G is reset. |
| 5 | R5 | RW | 0 | GPIO Port F Software Reset<br>Value Description<br>0 GPIO Port F is not reset.<br>1 GPIO Port F is reset. |
| 4 | R4 | RW | 0 | GPIO Port E Software Reset<br>Value Description<br>0 GPIO Port E is not reset.<br>1 GPIO Port E is reset. |
| 3 | R3 | RW | 0 | GPIO Port D Software Reset<br>Value Description<br>0 GPIO Port D is not reset.<br>1 GPIO Port D is reset. |
| 2 | R2 | RW | 0 | GPIO Port C Software Reset<br>Value Description<br>0 GPIO Port C is not reset.<br>1 GPIO Port C is reset. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | R1 | RW | 0 | GPIO Port B Software Reset<br><br>Value Description<br>0 GPIO Port B is not reset.<br>1 GPIO Port B is reset. |
| 0 | R0 | RW | 0 | GPIO Port A Software Reset<br><br>Value Description<br>0 GPIO Port A is not reset.<br>1 GPIO Port A is reset. |

## Register 73: Micro Direct Memory Access Software Reset (SRDMA), offset 0x50C

The **SRDMA** register provides software the capability to reset the available µDMA module.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRDMA** register. While the **SRDMA** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRDMA** bit.

There may be latency from the clearing of the **SRDMA** bit to when the peripheral is ready for use. Software should check the corresponding **PRDMA** bit to verify that the µDMA Module registers are ready to be accessed.

**Important:** This register should be used to reset the µDMA module.

Micro Direct Memory Access Software Reset (SRDMA)

Base 0x400F.E000
Offset 0x50C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | µDMA Module Software Reset<br>Value Description<br>0 µDMA module is not reset.<br>1 µDMA module is reset. |

## Register 74: EPI Software Reset (SREPI), offset 0x510

The **SREPI** register provides software the capability to reset the available EPI module.

A peripheral is reset by software using a simple two-step process:

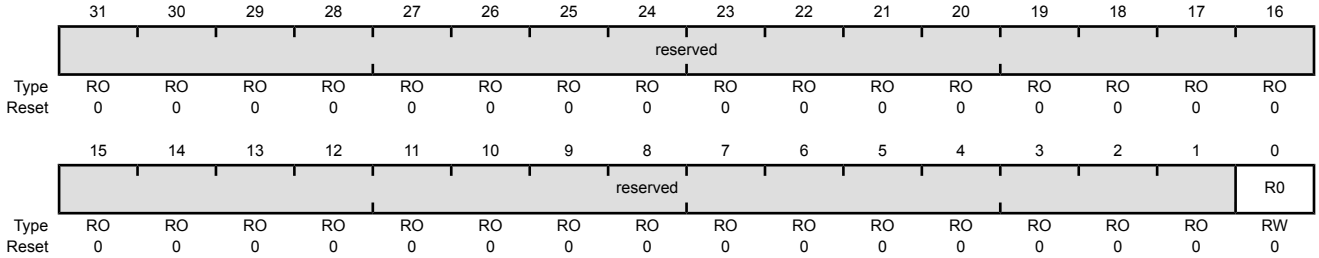
1. Software sets a bit (or bits) in the **SREPI** register. While the **SREPI** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SREPI** bit.

There may be latency from the clearing of the **SREPI** bit to when the peripheral is ready for use. Software should check the corresponding **PREPI** bit to verify that the EPI Module registers are ready to be accessed.

**Important:** This register should be used to reset the EPI module.

EPI Software Reset (SREPI)

Base 0x400F.E000
Offset 0x510
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | EPI Module Software Reset<br>Value Description<br>0 EPI module is not reset.<br>1 EPI module is reset. |

## Register 75: Hibernation Software Reset (SRHIB), offset 0x514

The **SRHIB** register provides software the capability to reset the available Hibernation module.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRHIB** register. While the **SRHIB** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRHIB** bit.

There may be latency from the clearing of the **SRHIB** bit to when the peripheral is ready for use. Software should check the corresponding **PRHIB** bit to verify that the Hibernation Module registers are ready to be accessed.

**Important:** This register should be used to reset the Hibernation module.

Hibernation Software Reset (SRHIB)

Base 0x400F.E000
Offset 0x514
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | Hibernation Module Software Reset<br>Value Description<br>0 Hibernation module is not reset.<br>1 Hibernation module is reset. |

## Register 76: Universal Asynchronous Receiver/Transmitter Software Reset (SRUART), offset 0x518

The **SRUART** register provides software the capability to reset the available UART modules.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRUART** register. While the **SRUART** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRUART** bit.

There may be latency from the clearing of the **SRUART** bit to when the peripheral is ready for use. Software should check the corresponding **PRUART** bit to verify that the UART Module registers are ready to be accessed.

**Important:** This register should be used to reset the UART modules.

Universal Asynchronous Receiver/Transmitter Software Reset (SRUART)

Base 0x400F.E000
Offset 0x518
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | R7 | RW | 0 | UART Module 7 Software Reset<br>Value Description<br>0 UART module 7 is not reset.<br>1 UART module 7 is reset. |
| 6 | R6 | RW | 0 | UART Module 6 Software Reset<br>Value Description<br>0 UART module 6 is not reset.<br>1 UART module 6 is reset. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | R5 | RW | 0 | UART Module 5 Software Reset<br>Value Description<br>0 UART module 5 is not reset.<br>1 UART module 5 is reset. |
| 4 | R4 | RW | 0 | UART Module 4 Software Reset<br>Value Description<br>0 UART module 4 is not reset.<br>1 UART module 4 is reset. |
| 3 | R3 | RW | 0 | UART Module 3 Software Reset<br>Value Description<br>0 UART module 3 is not reset.<br>1 UART module 3 is reset. |
| 2 | R2 | RW | 0 | UART Module 2 Software Reset<br>Value Description<br>0 UART module 2 is not reset.<br>1 UART module 2 is reset. |
| 1 | R1 | RW | 0 | UART Module 1 Software Reset<br>Value Description<br>0 UART module 1 is not reset.<br>1 UART module 1 is reset. |
| 0 | R0 | RW | 0 | UART Module 0 Software Reset<br>Value Description<br>0 UART module 0 is not reset.<br>1 UART module 0 is reset. |

## Register 77: Synchronous Serial Interface Software Reset (SRSSI), offset 0x51C

The **SRSSI** register provides software the capability to reset the available SSI modules.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SRSSI** register. While the **SRSSI** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRSSI** bit.

There may be latency from the clearing of the **SRSSI** bit to when the peripheral is ready for use. Software should check the corresponding **PRSSI** bit to verify that the SSI Module registers are ready to be accessed.

**Important:** This register should be used to reset the SSI modules.

Synchronous Serial Interface Software Reset (SRSSI)

Base 0x400F.E000
Offset 0x51C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | R3 | R2 | R1 | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | R3 | RW | 0 | SSI Module 3 Software Reset<br>Value Description<br>0 SSI module 3 is not reset.<br>1 SSI module 3 is reset. |
| 2 | R2 | RW | 0 | SSI Module 2 Software Reset<br>Value Description<br>0 SSI module 2 is not reset.<br>1 SSI module 2 is reset. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | R1 | RW | 0 | SSI Module 1 Software Reset<br><br>Value Description<br>0 SSI module 1 is not reset.<br>1 SSI module 1 is reset. |
| 0 | R0 | RW | 0 | SSI Module 0 Software Reset<br><br>Value Description<br>0 SSI module 0 is not reset.<br>1 SSI module 0 is reset. |

## Register 78: Inter-Integrated Circuit Software Reset (SRI2C), offset 0x520

The **SRI2C** register provides software the capability to reset the available $I^2C$ modules.

A peripheral is reset by software using a simple two-step process:

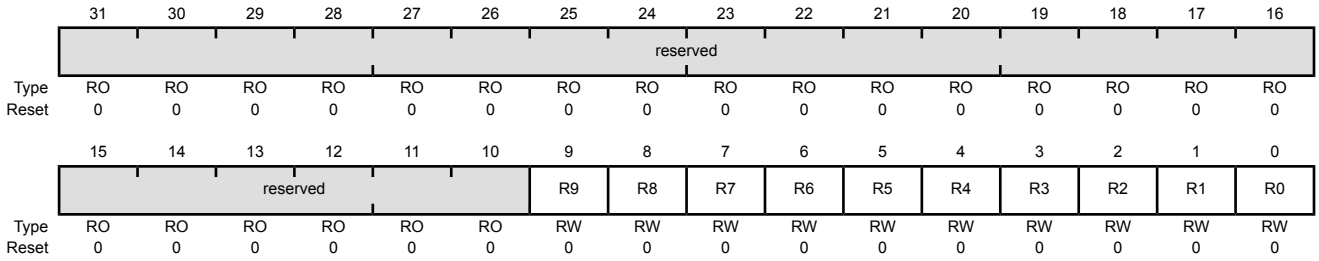
1. Software sets a bit (or bits) in the **SRI2C** register. While the **SRI2C** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRI2C** bit.

There may be latency from the clearing of the **SRI2C** bit to when the peripheral is ready for use. Software should check the corresponding **PRI2C** bit to verify that the $I^2C$ Module registers are ready to be accessed.

**Important:** This register should be used to reset the $I^2C$ modules.

Inter-Integrated Circuit Software Reset (SRI2C)

Base 0x400F.E000
Offset 0x520
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |
| 9 | R9 | RW | 0 | $I^2C$ Module 9 Software Reset<br>Value Description<br>0 $I^2C$ module 9 is not reset.<br>1 $I^2C$ module 9 is reset. |
| 8 | R8 | RW | 0 | $I^2C$ Module 8 Software Reset<br>Value Description<br>0 $I^2C$ module 8 is not reset.<br>1 $I^2C$ module 8 is reset. |
| 7 | R7 | RW | 0 | $I^2C$ Module 7 Software Reset<br>Value Description<br>0 $I^2C$ module 7 is not reset.<br>1 $I^2C$ module 7 is reset. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | R6 | RW | 0 | $I^2C$ Module 6 Software Reset<br>Value Description<br>0 $I^2C$ module 6 is not reset.<br>1 $I^2C$ module 6 is reset. |
| 5 | R5 | RW | 0 | $I^2C$ Module 5 Software Reset<br>Value Description<br>0 $I^2C$ module 5 is not reset.<br>1 $I^2C$ module 5 is reset. |
| 4 | R4 | RW | 0 | $I^2C$ Module 4 Software Reset<br>Value Description<br>0 $I^2C$ module 4 is not reset.<br>1 $I^2C$ module 4 is reset. |
| 3 | R3 | RW | 0 | $I^2C$ Module 3 Software Reset<br>Value Description<br>0 $I^2C$ module 3 is not reset.<br>1 $I^2C$ module 3 is reset. |
| 2 | R2 | RW | 0 | $I^2C$ Module 2 Software Reset<br>Value Description<br>0 $I^2C$ module 2 is not reset.<br>1 $I^2C$ module 2 is reset. |
| 1 | R1 | RW | 0 | $I^2C$ Module 1 Software Reset<br>Value Description<br>0 $I^2C$ module 1 is not reset.<br>1 $I^2C$ module 1 is reset. |
| 0 | R0 | RW | 0 | $I^2C$ Module 0 Software Reset<br>Value Description<br>0 $I^2C$ module 0 is not reset.<br>1 $I^2C$ module 0 is reset. |

## Register 79: Universal Serial Bus Software Reset (SRUSB), offset 0x528

The **SRUSB** register provides software the capability to reset the available USB module.

A peripheral is reset by software using a simple two-step process:

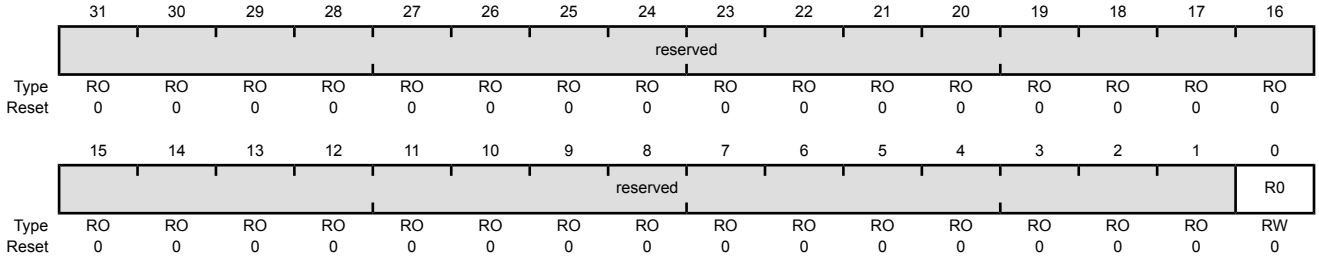
1. Software sets a bit (or bits) in the **SRUSB** register. While the **SRUSB** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRUSB** bit.

There may be latency from the clearing of the **SRUSB** bit to when the peripheral is ready for use. Software should check the corresponding **PRUSB** bit to verify that the USB Module registers are ready to be accessed.

**Important:** This register should be used to reset the USB module.

Universal Serial Bus Software Reset (SRUSB)

Base 0x400F.E000
Offset 0x528
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | USB Module Software Reset |

| Value | Description |
|---|---|
| 0 | USB module is not reset. |
| 1 | USB module is reset. |

## Register 80: Ethernet PHY Software Reset (SREPHY), offset 0x530

The **SREPHY** register provides software the capability to reset the available PHY module.

A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SREPHY** register. While the **SREPHY** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SREPHY** bit.

There may be latency from the clearing of the **SREPHY** bit to when the peripheral is ready for use. Software should check the corresponding **PREPHY** bit to verify that the EPHY Module registers are ready to be accessed.

**Important:** This register should be used to reset the Ethernet PHY module.

Ethernet PHY Software Reset (SREPHY)

Base 0x400F.E000
Offset 0x530
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | Ethernet PHY Module Software Reset<br>Value Description<br>0 Ethernet PHY module is not reset.<br>1 Ethernet PHY module is reset. |

## Register 81: Controller Area Network Software Reset (SRCAN), offset 0x534

The **SRCAN** register provides software the capability to reset the available CAN modules.

A peripheral is reset by software using a simple two-step process:

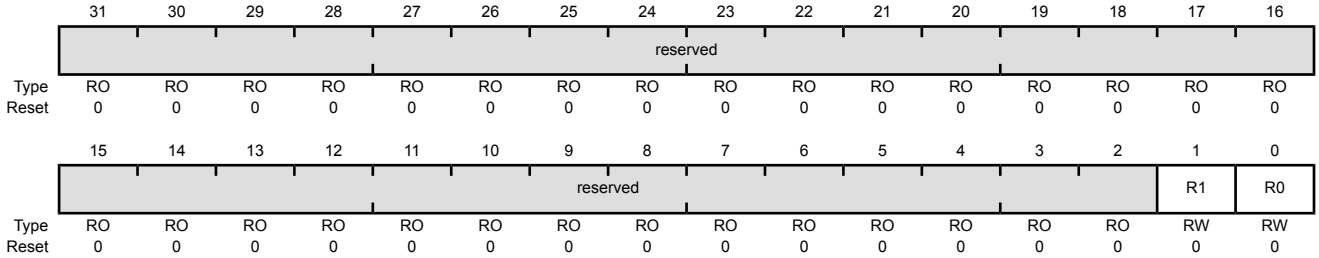
1. Software sets a bit (or bits) in the **SRCAN** register. While the **SRCAN** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRCAN** bit.

There may be latency from the clearing of the **SRCAN** bit to when the peripheral is ready for use. Software should check the corresponding **PRCAN** bit to verify that the CAN Module registers are ready to be accessed.

**Important:** This register should be used to reset the CAN modules.

Controller Area Network Software Reset (SRCAN)

Base 0x400F.E000
Offset 0x534
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RW | 0 | CAN Module 1 Software Reset<br>Value Description<br>0 CAN module 1 is not reset.<br>1 CAN module 1 is reset. |
| 0 | R0 | RW | 0 | CAN Module 0 Software Reset<br>Value Description<br>0 CAN module 0 is not reset.<br>1 CAN module 0 is reset. |

## Register 82: Analog-to-Digital Converter Software Reset (SRADC), offset 0x538

The **SRADC** register provides software the capability to reset the available ADC modules.

A peripheral is reset by software using a simple two-step process:

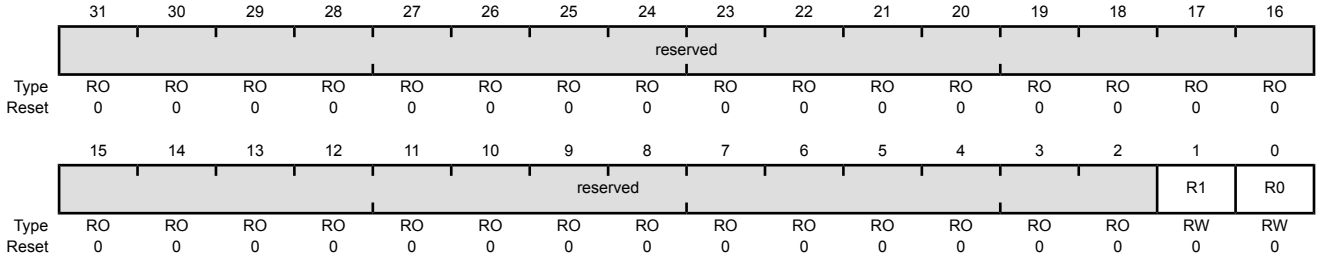
1. Software sets a bit (or bits) in the **SRADC** register. While the **SRADC** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRADC** bit.

There may be latency from the clearing of the **SRADC** bit to when the peripheral is ready for use. Software should check the corresponding **PRADC** bit to verify that the ADC Module registers are ready to be accessed.

**Important:** This register should be used to reset the ADC modules.

Analog-to-Digital Converter Software Reset (SRADC)

Base 0x400F.E000
Offset 0x538
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RW | 0 | ADC Module 1 Software Reset<br>Value Description<br>0 ADC module 1 is not reset.<br>1 ADC module 1 is reset. |
| 0 | R0 | RW | 0 | ADC Module 0 Software Reset<br>Value Description<br>0 ADC module 0 is not reset.<br>1 ADC module 0 is reset. |

## Register 83: Analog Comparator Software Reset (SRACMP), offset 0x53C

The **SRACMP** register provides software the capability to reset the available analog comparator module.

A block is reset by software using a simple two-step process:

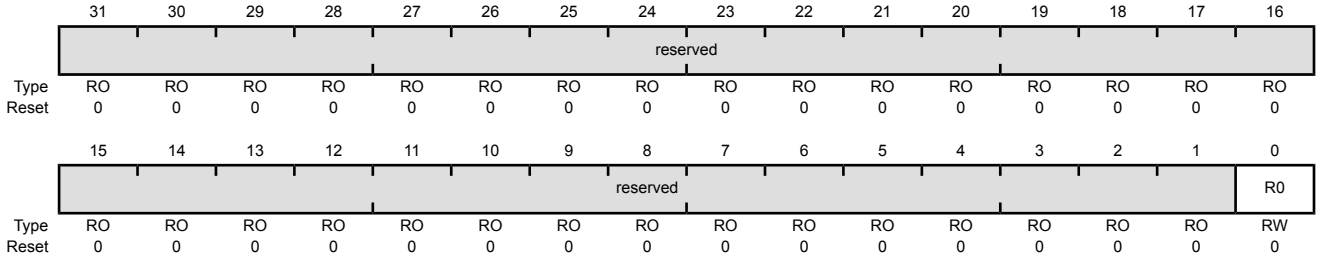
1. Software sets a bit (or bits) in the **SRACMP** register. While the **SRACMP** bit is 1, the module is held in reset.
2. Software completes the reset process by clearing the **SRACMP** bit.

There may be latency from the clearing of the **SRACMP** bit to when the module is ready for use. Software should check the corresponding **PRACMP** bit to verify that the Analog Comparator Module registers are ready to be accessed.

**Important:** This register should be used to reset the analog comparator module.

Analog Comparator Software Reset (SRACMP)

Base 0x400F.E000
Offset 0x53C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | Analog Comparator Module 0 Software Reset<br><br>Value Description<br>0 Analog comparator module is not reset.<br>1 Analog comparator module is reset. |

## Register 84: Pulse Width Modulator Software Reset (SRPWM), offset 0x540

The **SRPWM** register provides software the capability to reset the available PWM modules.

A peripheral is reset by software using a simple two-step process:

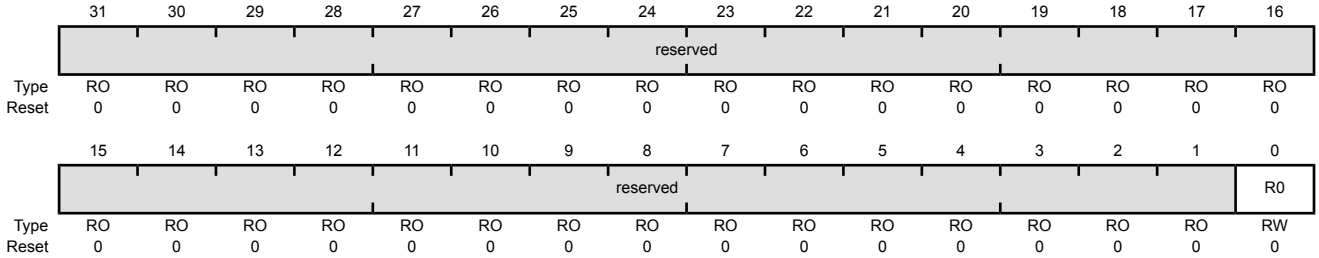
1. Software sets a bit (or bits) in the **SRPWM** register. While the **SRPWM** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRPWM** bit.

There may be latency from the clearing of the **SRPWM** bit to when the peripheral is ready for use. Software should check the corresponding **PRPWM** bit to verify that the PWM Module registers are ready to be accessed.

**Important:** This register should be used to reset the PWM modules.

Pulse Width Modulator Software Reset (SRPWM)

Base 0x400F.E000
Offset 0x540
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | PWM Module 0 Software Reset<br>Value Description<br>0 PWM module 0 is not reset.<br>1 PWM module 0 is reset. |

## Register 85: Quadrature Encoder Interface Software Reset (SRQEI), offset 0x544

The **SRQEI** register provides software the capability to reset the available QEI modules.

A peripheral is reset by software using a simple two-step process:

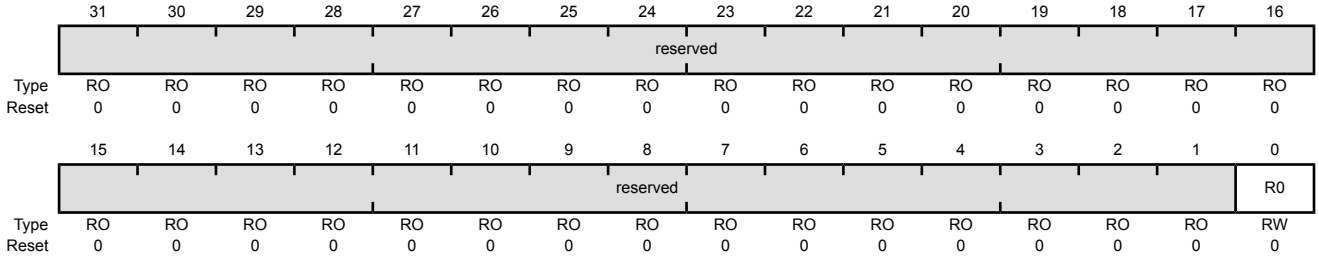
1. Software sets a bit (or bits) in the **SRQEI** register. While the **SRQEI** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRQEI** bit.

There may be latency from the clearing of the **SRQEI** bit to when the peripheral is ready for use. Software should check the corresponding **PRQEI** bit to verify that the QEI Module registers are ready to be accessed.

**Important:** This register should be used to reset the QEI modules.

Quadrature Encoder Interface Software Reset (SRQEI)

Base 0x400F.E000
Offset 0x544
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | QEI Module 0 Software Reset<br>Value Description<br>0 QEI module 0 is not reset.<br>1 QEI module 0 is reset. |

## Register 86: EEPROM Software Reset (SREEPROM), offset 0x558

The **SREEPROM** register provides software the capability to reset the available EEPROM module.

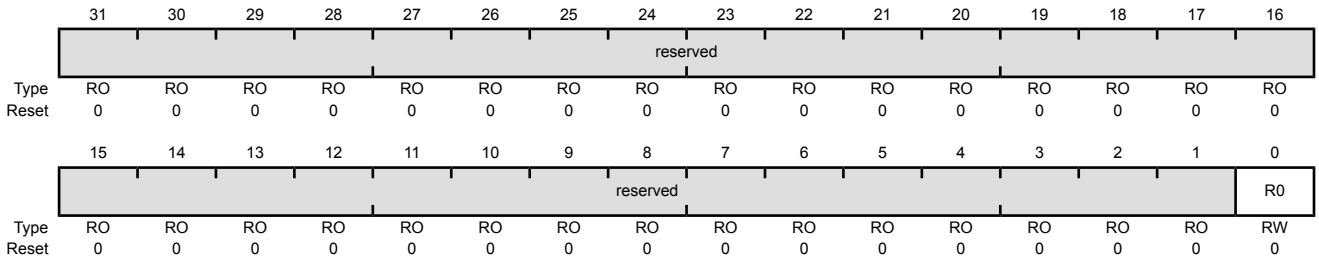
A peripheral is reset by software using a simple two-step process:

1. Software sets a bit (or bits) in the **SREEPROM** register. While the **SREEPROM** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SREEPROM** bit.

There may be latency from the clearing of the **SREEPROM** bit to when the peripheral is ready for use. Software should check the corresponding **PREEPROM** bit to verify that the EEPROM Module registers are ready to be accessed.

EEPROM Software Reset (SREEPROM)

Base 0x400F.E000
Offset 0x558
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | EEPROM Module 0 Software Reset |

| Value | Description |
|---|---|
| 0 | EEPROM module is not reset. |
| 1 | EEPROM module is reset. |

## Register 87: CRC and Cryptographic Modules Software Reset (SRCCM), offset 0x574

The **SRCCM** register provides software the capability to reset the CRC and Cryptographic Modules (AES, DES, and SHA/MD5).

A module is reset by software using a simple two-step process:

1. Software sets the bit in the **SRCCM** register. While the **SRCCM** bit is 1, the peripherals are held in reset.

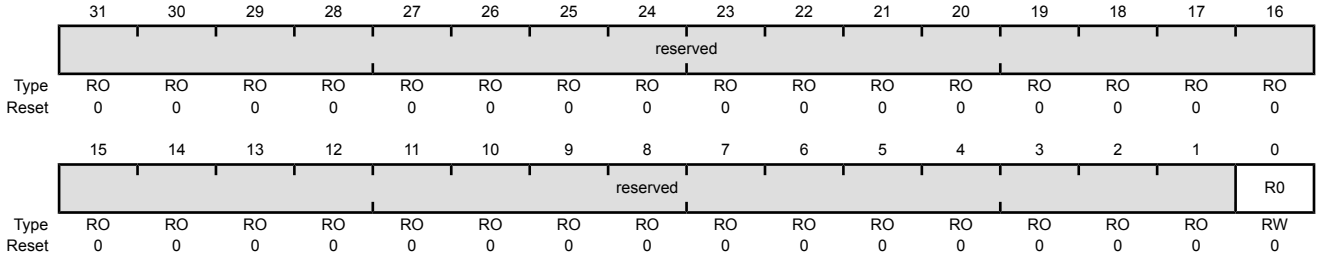
2. Software completes the reset process by clearing the **SRCCM** bit.

There may be latency from the clearing of the **SRCCM** bit to when the peripheral is ready for use. Software should check the corresponding **PRCCM** bit to verify that the CRC and Cryptographic Module (AES, DES, and SHA/MD5) registers are ready to be accessed.

**Important:** This register should be used to reset the CRC, AES, DES, and SHA/MD5 modules.

CRC and Cryptographic Modules Software Reset (SRCCM)

Base 0x400F.E000
Offset 0x574
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | CRC and Cryptographic Modules Software Reset<br>Value Description<br>0 The CRC, AES, DES, and SHA/MD5 modules are not reset.<br>1 The CRC, AES, DES, and SHA/MD5 modules are reset. |

## Register 88: LCD Controller Software Reset (SRLCD), offset 0x590

The **SRLCD** register provides software the capability to reset the available LCD module.

A peripheral is reset by software using a simple two-step process:

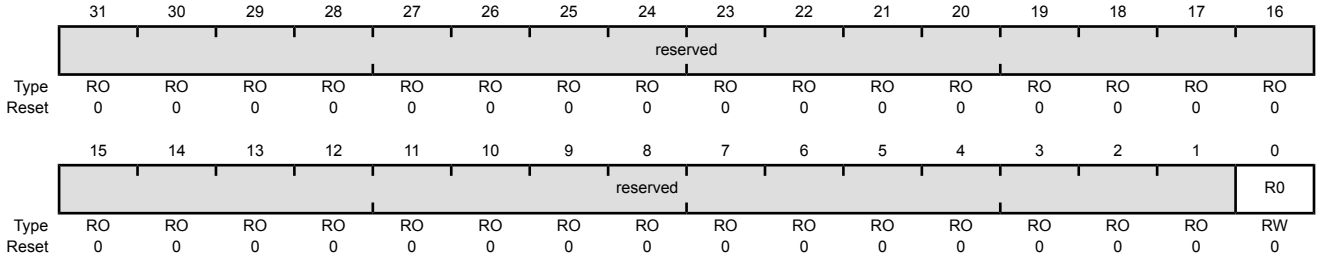
1. Software sets a bit (or bits) in the **SRLCD** register. While the **SRLCD** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SRLCD** bit.

There may be latency from the clearing of the **SRLCD** bit to when the peripheral is ready for use. Software should check the corresponding **PRLCD** bit to verify that the LCD Module registers are ready to be accessed.

**Important:** This register should be used to reset the LCD controller modules.

LCD Controller Software Reset (SRLCD)

Base 0x400F.E000
Offset 0x590
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | LCD Module 0 Software Reset<br>Value Description<br>0 LCD module 0 is not reset.<br>1 LCD module 0 is reset. |

## Register 89: 1-Wire Software Reset (SROWIRE), offset 0x598

The **SROWIRE** register provides software the capability to reset the available 1-Wire Module.

A peripheral is reset by software using a simple two-step process:

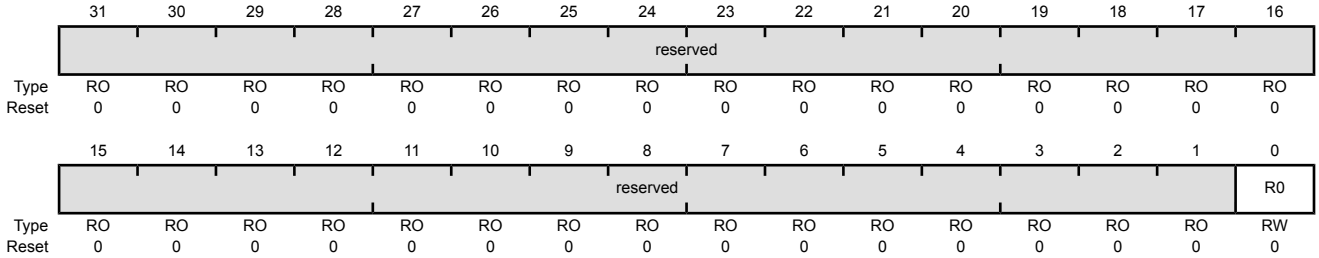
1. Software sets a bit (or bits) in the **SROWIRE** register. While the **SROWIRE** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SROWIRE** bit.

There may be latency from the clearing of the **SROWIRE** bit to when the peripheral is ready for use. Software should check the corresponding **PROWIRE** bit to verify that the 1-Wire Module registers are ready to be accessed.

**Important:** This register should be used to reset the 1-Wire controller modules.

1-Wire Software Reset (SROWIRE)

Base 0x400F.E000
Offset 0x598
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | 1-Wire Module Software Reset<br>Value Description<br>0 1-Wire module is not reset.<br>1 1-Wire module is reset. |

## Register 90: Ethernet MAC Software Reset (SREMAC), offset 0x59C

The **SREMAC** register provides software the capability to reset the available Ethernet MAC module.

A peripheral is reset by software using a simple two-step process:

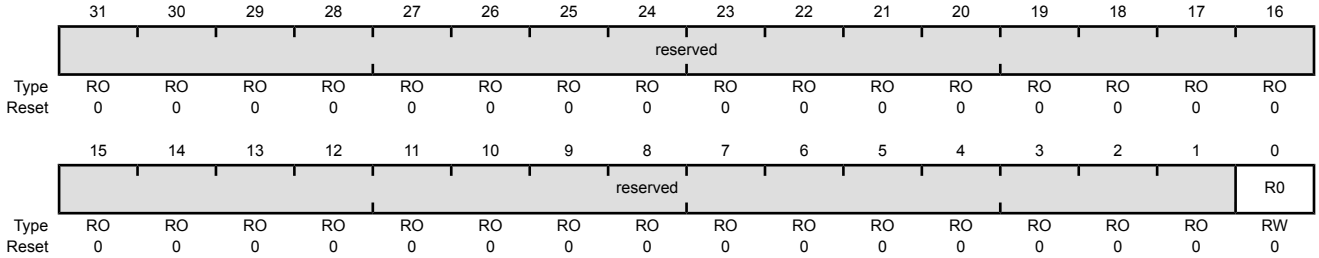
1. Software sets a bit (or bits) in the **SREMAC** register. While the **SREMAC** bit is 1, the peripheral is held in reset.
2. Software completes the reset process by clearing the **SREMAC** bit.

There may be latency from the clearing of the **SREMAC** bit to when the peripheral is ready for use. Software should check the corresponding **PREMAC** bit to verify that the Ethernet MAC Module registers are ready to be accessed.

**Important:** This register should be used to reset the Ethernet controller MAC module.

Ethernet MAC Software Reset (SREMAC)

Base 0x400F.E000
Offset 0x59C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | Ethernet Controller MAC Module 0 Software Reset<br>Value Description<br>0 Ethernet Controller MAC module 0 is not reset.<br>1 Ethernet Controller MAC module 0 is reset. |

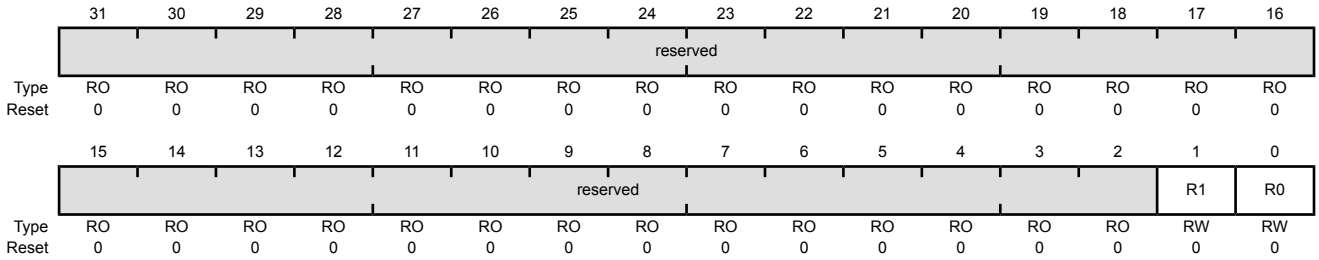
## Register 91: Watchdog Timer Run Mode Clock Gating Control (RCGCWD), offset 0x600

The **RCGCWD** register provides software the capability to enable and disable watchdog modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the watchdog modules

Watchdog Timer Run Mode Clock Gating Control (RCGCWD)

Base 0x400F.E000
Offset 0x600
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RW | 0 | Watchdog Timer 1 Run Mode Clock Gating Control<br><br>Value Description<br>0 Watchdog module 1 is disabled.<br>1 Enable and provide a clock to Watchdog module 1 in Run mode. |
| 0 | R0 | RW | 0 | Watchdog Timer 0 Run Mode Clock Gating Control<br><br>Value Description<br>0 Watchdog module 0 is disabled.<br>1 Enable and provide a clock to Watchdog module 0 in Run mode. |

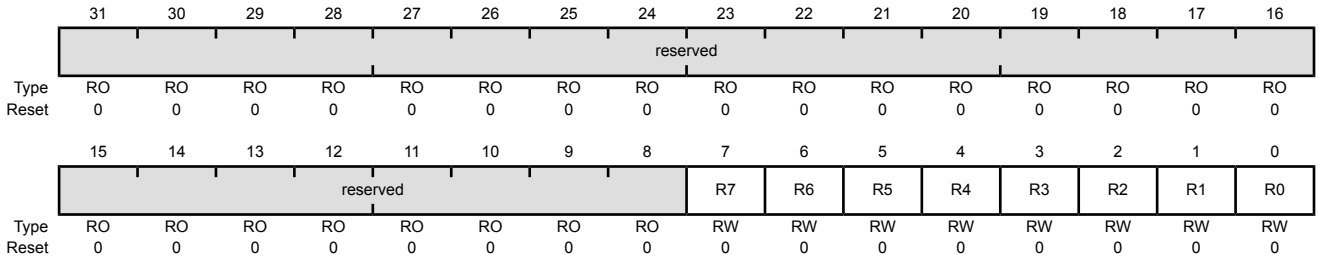
## Register 92: 16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER), offset 0x604

The **RCGCGPT32** register provides software the capability to enable and disable 16/32-bit timer modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the timer modules.

16/32-Bit General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER)

Base 0x400F.E000
Offset 0x604
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | R7 | RW | 0 | 16/32-Bit General-Purpose Timer 7 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 7 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 7 in Run mode. |
| 6 | R6 | RW | 0 | 16/32-Bit General-Purpose Timer 6 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 6 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 6 in Run mode. |
| 5 | R5 | RW | 0 | 16/32-Bit General-Purpose Timer 5 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 5 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 5 in Run mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | R4 | RW | 0 | 16/32-Bit General-Purpose Timer 4 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 4 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 4 in Run mode. |
| 3 | R3 | RW | 0 | 16/32-Bit General-Purpose Timer 3 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 3 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 3 in Run mode. |
| 2 | R2 | RW | 0 | 16/32-Bit General-Purpose Timer 2 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 2 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 2 in Run mode. |
| 1 | R1 | RW | 0 | 16/32-Bit General-Purpose Timer 1 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 1 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 1 in Run mode. |
| 0 | R0 | RW | 0 | 16/32-Bit General-Purpose Timer 0 Run Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 0 is disabled.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 0 in Run mode. |

## Register 93: General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO), offset 0x608

The **RCGCGPIO** register provides software the capability to enable and disable GPIO modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the GPIO modules.

General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

Base 0x400F.E000
Offset 0x608
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | R17 | R16 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | R17 | RW | 0 | GPIO Port T Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port T is disabled.<br>1 Enable and provide a clock to GPIO Port T in Run mode. |
| 16 | R16 | RW | 0 | GPIO Port S Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port S is disabled.<br>1 Enable and provide a clock to GPIO Port S in Run mode. |
| 15 | R15 | RW | 0 | GPIO Port R Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port R is disabled.<br>1 Enable and provide a clock to GPIO Port R in Run mode. |
| 14 | R14 | RW | 0 | GPIO Port Q Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port Q is disabled.<br>1 Enable and provide a clock to GPIO Port Q in Run mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | R13 | RW | 0 | GPIO Port P Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port P is disabled.<br>1 Enable and provide a clock to GPIO Port P in Run mode. |
| 12 | R12 | RW | 0 | GPIO Port N Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port N is disabled.<br>1 Enable and provide a clock to GPIO Port N in Run mode. |
| 11 | R11 | RW | 0 | GPIO Port M Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port M is disabled.<br>1 Enable and provide a clock to GPIO Port M in Run mode. |
| 10 | R10 | RW | 0 | GPIO Port L Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port L is disabled.<br>1 Enable and provide a clock to GPIO Port L in Run mode. |
| 9 | R9 | RW | 0 | GPIO Port K Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port K is disabled.<br>1 Enable and provide a clock to GPIO Port K in Run mode. |
| 8 | R8 | RW | 0 | GPIO Port J Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port J is disabled.<br>1 Enable and provide a clock to GPIO Port J in Run mode. |
| 7 | R7 | RW | 0 | GPIO Port H Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port H is disabled.<br>1 Enable and provide a clock to GPIO Port H in Run mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | R6 | RW | 0 | GPIO Port G Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port G is disabled.<br>1 Enable and provide a clock to GPIO Port G in Run mode. |
| 5 | R5 | RW | 0 | GPIO Port F Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port F is disabled.<br>1 Enable and provide a clock to GPIO Port F in Run mode. |
| 4 | R4 | RW | 0 | GPIO Port E Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port E is disabled.<br>1 Enable and provide a clock to GPIO Port E in Run mode. |
| 3 | R3 | RW | 0 | GPIO Port D Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port D is disabled.<br>1 Enable and provide a clock to GPIO Port D in Run mode. |
| 2 | R2 | RW | 0 | GPIO Port C Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port C is disabled.<br>1 Enable and provide a clock to GPIO Port C in Run mode. |
| 1 | R1 | RW | 0 | GPIO Port B Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port B is disabled.<br>1 Enable and provide a clock to GPIO Port B in Run mode. |
| 0 | R0 | RW | 0 | GPIO Port A Run Mode Clock Gating Control<br>Value Description<br>0 GPIO Port A is disabled.<br>1 Enable and provide a clock to GPIO Port A in Run mode. |

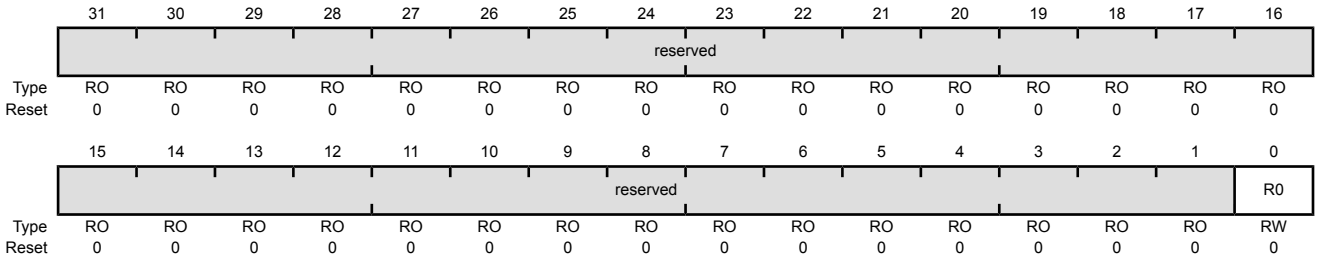
## Register 94: Micro Direct Memory Access Run Mode Clock Gating Control (RCGCDMA), offset 0x60C

The **RCGCDMA** register provides software the capability to enable and disable the µDMA module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the µDMA module.

Micro Direct Memory Access Run Mode Clock Gating Control (RCGCDMA)

Base 0x400F.E000
Offset 0x60C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | µDMA Module Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | µDMA module is disabled. |
| 1 | Enable and provide a clock to the µDMA module in Run mode. |

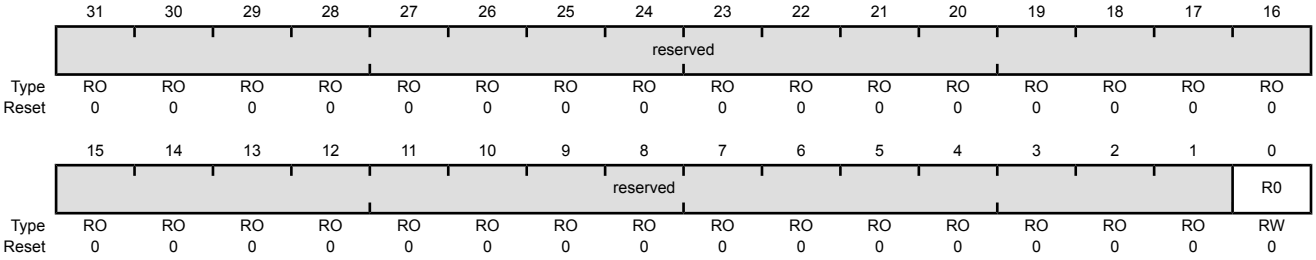
## Register 95: EPI Run Mode Clock Gating Control (RCGCEPI), offset 0x610

The **RCGCEPI** register provides software the capability to enable and disable the EPI module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the EPI module.

EPI Run Mode Clock Gating Control (RCGCEPI)

Base 0x400F.E000
Offset 0x610
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | EPI Module Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | EPI module is disabled. |
| 1 | Enable and provide a clock to the EPI module in Run mode. |

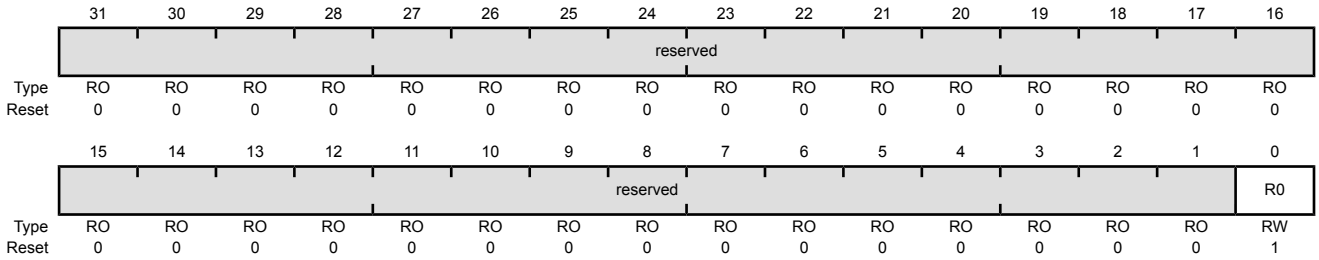
## Register 96: Hibernation Run Mode Clock Gating Control (RCGCHIB), offset 0x614

The **RCGCHIB** register provides software the capability to enable and disable the Hibernation module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the Hibernation module.

Hibernation Run Mode Clock Gating Control (RCGCHIB)

Base 0x400F.E000
Offset 0x614
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 1 | Hibernation Module Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Hibernation module is disabled. |
| 1 | Enable and provide a clock to the Hibernation module in Run mode. |

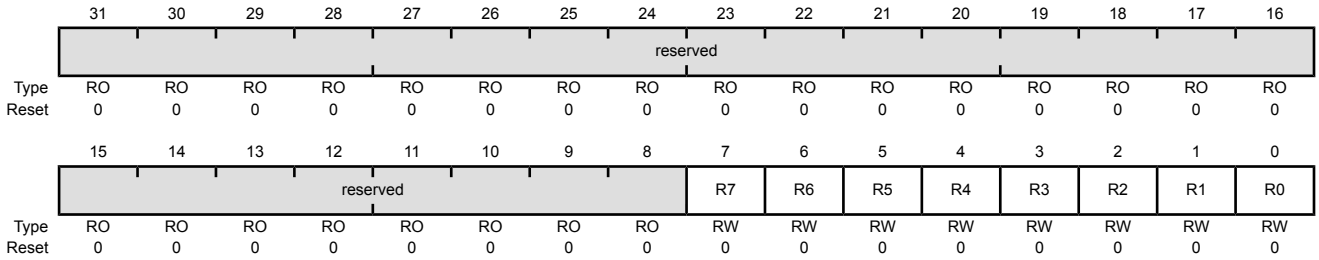
## Register 97: Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART), offset 0x618

The **RCGCUART** register provides software the capability to enable and disable the UART modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the UART modules.

Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART)

Base 0x400F.E000
Offset 0x618
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | R7 | RW | 0 | UART Module 7 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 7 is disabled.<br>1 Enable and provide a clock to UART module 7 in Run mode. |
| 6 | R6 | RW | 0 | UART Module 6 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 6 is disabled.<br>1 Enable and provide a clock to UART module 6 in Run mode. |
| 5 | R5 | RW | 0 | UART Module 5 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 5 is disabled.<br>1 Enable and provide a clock to UART module 5 in Run mode. |
| 4 | R4 | RW | 0 | UART Module 4 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 4 is disabled.<br>1 Enable and provide a clock to UART module 4 in Run mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | R3 | RW | 0 | UART Module 3 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 3 is disabled.<br>1 Enable and provide a clock to UART module 3 in Run mode. |
| 2 | R2 | RW | 0 | UART Module 2 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 2 is disabled.<br>1 Enable and provide a clock to UART module 2 in Run mode. |
| 1 | R1 | RW | 0 | UART Module 1 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 1 is disabled.<br>1 Enable and provide a clock to UART module 1 in Run mode. |
| 0 | R0 | RW | 0 | UART Module 0 Run Mode Clock Gating Control<br>Value Description<br>0 UART module 0 is disabled.<br>1 Enable and provide a clock to UART module 0 in Run mode. |

## Register 98: Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI), offset 0x61C

The **RCGCSSI** register provides software the capability to enable and disable the SSI modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the SSI modules.

Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI)

Base 0x400F.E000
Offset 0x61C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | R3 | R2 | R1 | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | R3 | RW | 0 | SSI Module 3 Run Mode Clock Gating Control<br>Value Description<br>0 SSI module 3 is disabled.<br>1 Enable and provide a clock to SSI module 3 in Run mode. |
| 2 | R2 | RW | 0 | SSI Module 2 Run Mode Clock Gating Control<br>Value Description<br>0 SSI module 2 is disabled.<br>1 Enable and provide a clock to SSI module 2 in Run mode. |
| 1 | R1 | RW | 0 | SSI Module 1 Run Mode Clock Gating Control<br>Value Description<br>0 SSI module 1 is disabled.<br>1 Enable and provide a clock to SSI module 1 in Run mode. |
| 0 | R0 | RW | 0 | SSI Module 0 Run Mode Clock Gating Control<br>Value Description<br>0 SSI module 0 is disabled.<br>1 Enable and provide a clock to SSI module 0 in Run mode. |

## Register 99: Inter-Integrated Circuit Run Mode Clock Gating Control (RCGCI2C), offset 0x620

The **RCGCI2C** register provides software the capability to enable and disable the $I^2C$ modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the $I^2C$ modules.

Inter-Integrated Circuit Run Mode Clock Gating Control (RCGCI2C)

Base 0x400F.E000
Offset 0x620
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | R9 | R8 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Type | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |
| 9 | R9 | RW | 0 | $I^2C$ Module 9 Run Mode Clock Gating Control<br>Value Description<br>0 $I^2C$ module 9 is disabled.<br>1 Enable and provide a clock to $I^2C$ module 9 in Run mode. |
| 8 | R8 | RW | 0 | $I^2C$ Module 8 Run Mode Clock Gating Control<br>Value Description<br>0 $I^2C$ module 8 is disabled.<br>1 Enable and provide a clock to $I^2C$ module 8 in Run mode. |
| 7 | R7 | RW | 0 | $I^2C$ Module 7 Run Mode Clock Gating Control<br>Value Description<br>0 $I^2C$ module 7 is disabled.<br>1 Enable and provide a clock to $I^2C$ module 7 in Run mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | R6 | RW | 0 | I²C Module 6 Run Mode Clock Gating Control<br>Value Description<br>0 I²C module 6 is disabled.<br>1 Enable and provide a clock to I²C module 6 in Run mode. |
| 5 | R5 | RW | 0 | I²C Module 5 Run Mode Clock Gating Control<br>Value Description<br>0 I²C module 5 is disabled.<br>1 Enable and provide a clock to I²C module 5 in Run mode. |
| 4 | R4 | RW | 0 | I²C Module 4 Run Mode Clock Gating Control<br>Value Description<br>0 I²C module 4 is disabled.<br>1 Enable and provide a clock to I²C module 4 in Run mode. |
| 3 | R3 | RW | 0 | I²C Module 3 Run Mode Clock Gating Control<br>Value Description<br>0 I²C module 3 is disabled.<br>1 Enable and provide a clock to I²C module 3 in Run mode. |
| 2 | R2 | RW | 0 | I²C Module 2 Run Mode Clock Gating Control<br>Value Description<br>0 I²C module 2 is disabled.<br>1 Enable and provide a clock to I²C module 2 in Run mode. |
| 1 | R1 | RW | 0 | I²C Module 1 Run Mode Clock Gating Control<br>Value Description<br>0 I²C module 1 is disabled.<br>1 Enable and provide a clock to I²C module 1 in Run mode. |
| 0 | R0 | RW | 0 | I²C Module 0 Run Mode Clock Gating Control<br>Value Description<br>0 I²C module 0 is disabled.<br>1 Enable and provide a clock to I²C module 0 in Run mode. |

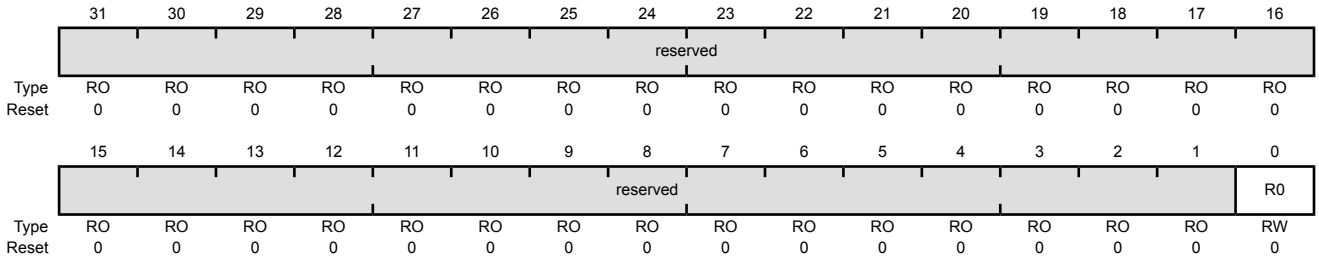
## Register 100: Universal Serial Bus Run Mode Clock Gating Control (RCGCUSB), offset 0x628

The **RCGCUSB** register provides software the capability to enable and disable the USB module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the USB module.

Universal Serial Bus Run Mode Clock Gating Control (RCGCUSB)

Base 0x400F.E000
Offset 0x628
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | USB Module Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | USB module is disabled. |
| 1 | Enable and provide a clock to the USB module in Run mode. |

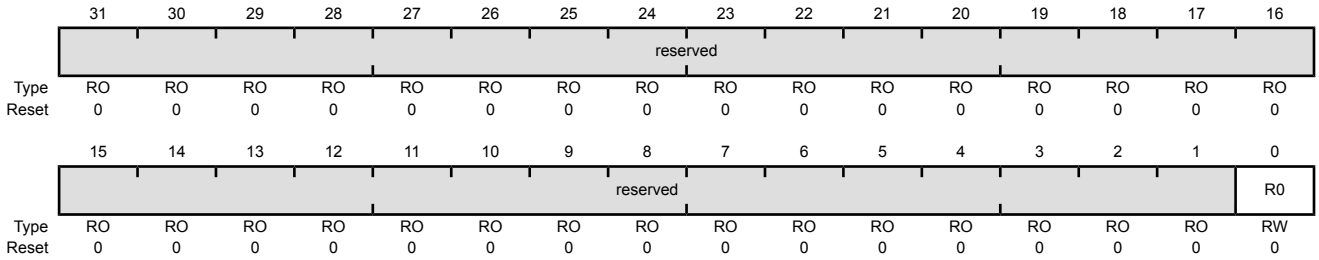
## Register 101: Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY), offset 0x630

The **RCGCEPHY** register provides software the capability to enable and disable the PHY module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the PHY module.

Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY)

Base 0x400F.E000
Offset 0x630
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | Ethernet PHY Module Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | PHY module is disabled. |
| 1 | Enable and provide a clock to the PHY module in Run mode. |

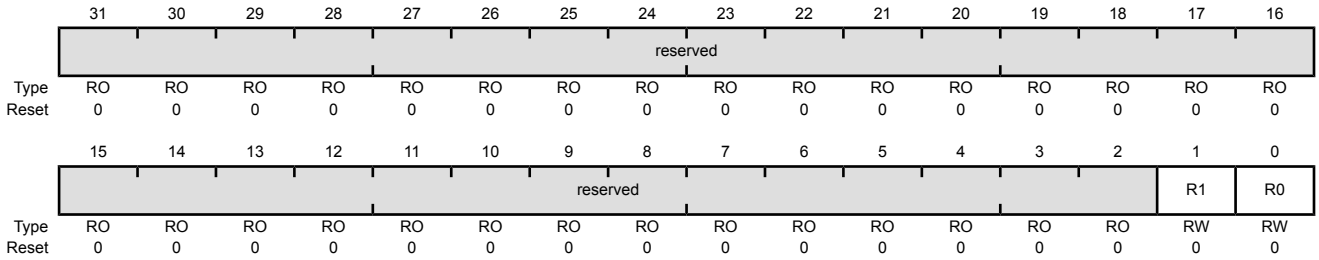
## Register 102: Controller Area Network Run Mode Clock Gating Control (RCGCCAN), offset 0x634

The **RCGCCAN** register provides software the capability to enable and disable the CAN modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the CAN modules.

Controller Area Network Run Mode Clock Gating Control (RCGCCAN)

Base 0x400F.E000
Offset 0x634
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RW | 0 | CAN Module 1 Run Mode Clock Gating Control<br>Value Description<br>0 CAN module 1 is disabled.<br>1 Enable and provide a clock to CAN module 1 in Run mode. |
| 0 | R0 | RW | 0 | CAN Module 0 Run Mode Clock Gating Control<br>Value Description<br>0 CAN module 0 is disabled.<br>1 Enable and provide a clock to CAN module 0 in Run mode. |

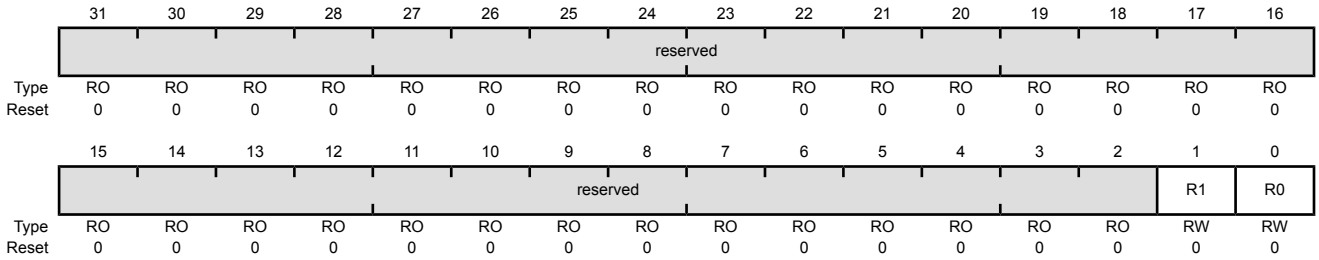
## Register 103: Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC), offset 0x638

The **RCGCADC** register provides software the capability to enable and disable the ADC modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the ADC modules.

Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)

Base 0x400F.E000
Offset 0x638
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RW | 0 | ADC Module 1 Run Mode Clock Gating Control<br>Value Description<br>0 ADC module 1 is disabled.<br>1 Enable and provide a clock to ADC module 1 in Run mode. |
| 0 | R0 | RW | 0 | ADC Module 0 Run Mode Clock Gating Control<br>Value Description<br>0 ADC module 0 is disabled.<br>1 Enable and provide a clock to ADC module 0 in Run mode. |

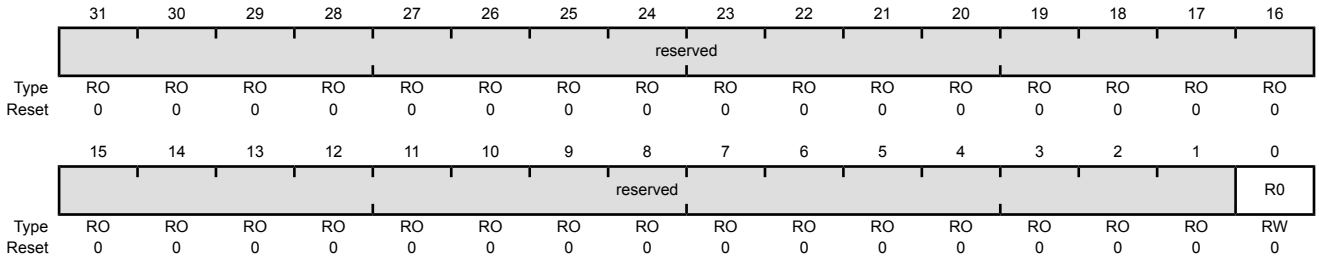
## Register 104: Analog Comparator Run Mode Clock Gating Control (RCGCACMP), offset 0x63C

The **RCGCACMP** register provides software the capability to enable and disable the analog comparator module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the analog comparator module.

Analog Comparator Run Mode Clock Gating Control (RCGCACMP)

Base 0x400F.E000
Offset 0x63C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | Analog Comparator Module 0 Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Analog comparator module is disabled. |
| 1 | Enable and provide a clock to the analog comparator module in Run mode. |

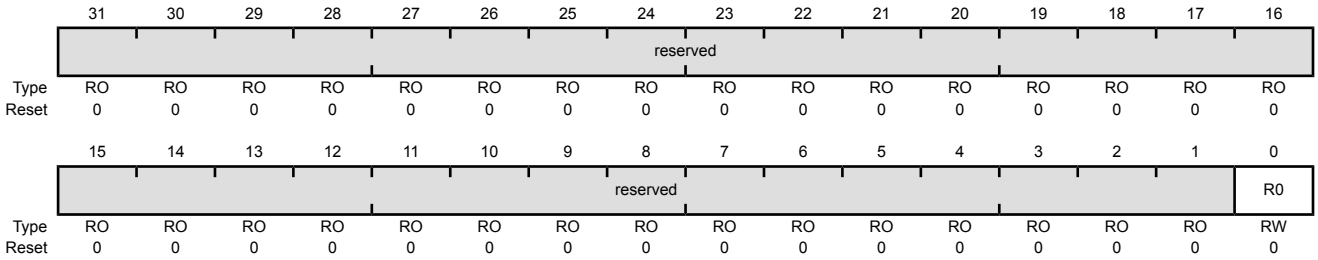
## Register 105: Pulse Width Modulator Run Mode Clock Gating Control (RCGCPWM), offset 0x640

The **RCGCPWM** register provides software the capability to enable and disable the PWM modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the PWM modules.

Pulse Width Modulator Run Mode Clock Gating Control (RCGCPWM)

Base 0x400F.E000
Offset 0x640
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | PWM Module 0 Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | PWM module 0 is disabled. |
| 1 | Enable and provide a clock to PWM module 0 in Run mode. |

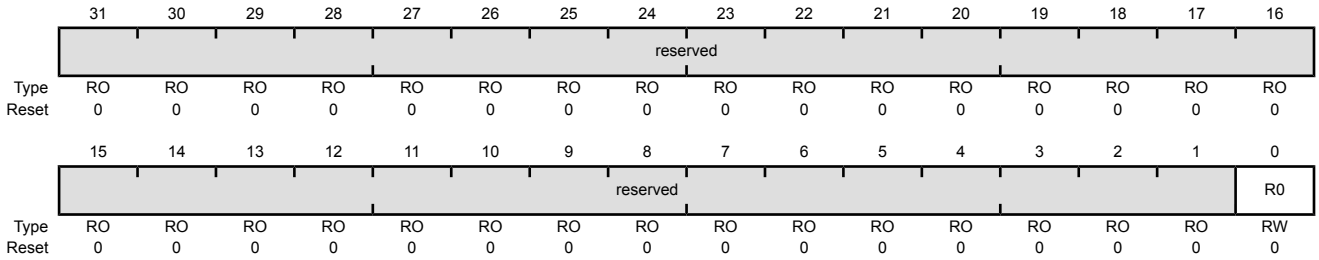
## Register 106: Quadrature Encoder Interface Run Mode Clock Gating Control (RCGCQEI), offset 0x644

The **RCGCQEI** register provides software the capability to enable and disable the QEI modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the QEI modules.

Quadrature Encoder Interface Run Mode Clock Gating Control (RCGCQEI)

Base 0x400F.E000
Offset 0x644
Type RW, reset 0x0000.0000



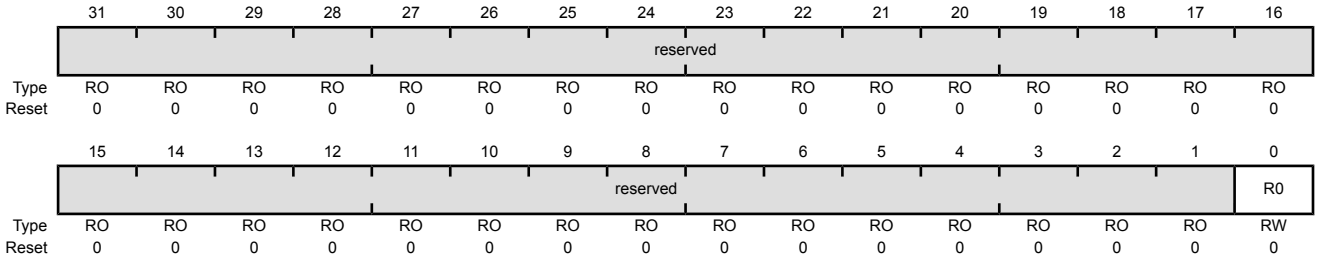

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | R0 | RW | 0 | QEI Module 0 Run Mode Clock Gating Control<br>Value Description<br>0 QEI module 0 is disabled.<br>1 Enable and provide a clock to QEI module 0 in Run mode. |

## Register 107: EEPROM Run Mode Clock Gating Control (RCGCEEPROM), offset 0x658

The **RCGCEEPROM** register provides software the capability to enable and disable the EEPROM module in Run mode. When enabled, the module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

EEPROM Run Mode Clock Gating Control (RCGCEEPROM)

Base 0x400F.E000
Offset 0x658
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | EEPROM Module 0 Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | EEPROM module is disabled. |
| 1 | Enable and provide a clock to the EEPROM module in Run mode. |

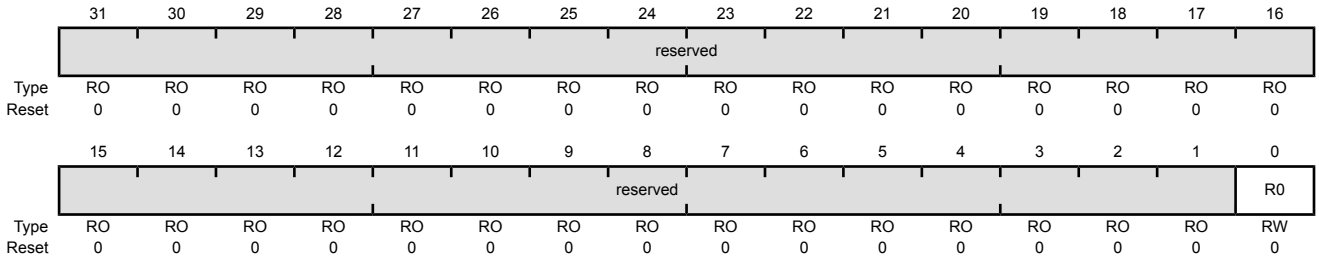
## Register 108: CRC and Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM), offset 0x674

The **RCGCCCM** register provides software the capability to enable and disable the CRC and Encryption Modules ( AES, DES, and SHA/MD5) in Run mode. When enabled, the modules are provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the CRC, AES, DES, and SHA/MD5 modules.

CRC and Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM)

Base 0x400F.E000
Offset 0x674
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | CRC and Cryptographic Modules Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | The CRC, AES, DES, and SHA/MD5 modules are disabled. |
| 1 | Enable and provide a clock to the CRC, AES, DES, and SHA/MD5 modules in Run mode. |

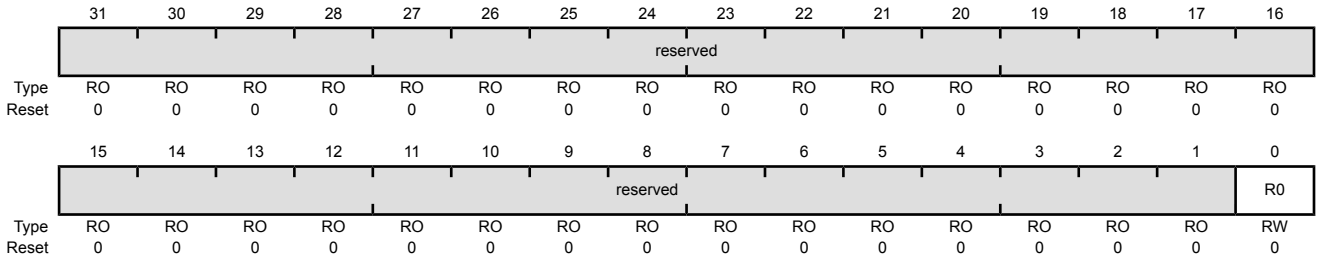
## Register 109: LCD Controller Run Mode Clock Gating Control (RCGCLCD), offset 0x690

The **RCGCLCD** register provides software the capability to enable and disable the LCD Controller module in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the LCD Controller module.

LCD Controller Run Mode Clock Gating Control (RCGCLCD)

Base 0x400F.E000
Offset 0x690
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | LCD Controller Module 0 Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | LCD Controller module 0 is disabled. |
| 1 | Enable and provide a clock to LCD Controller module 0 in Run mode. |

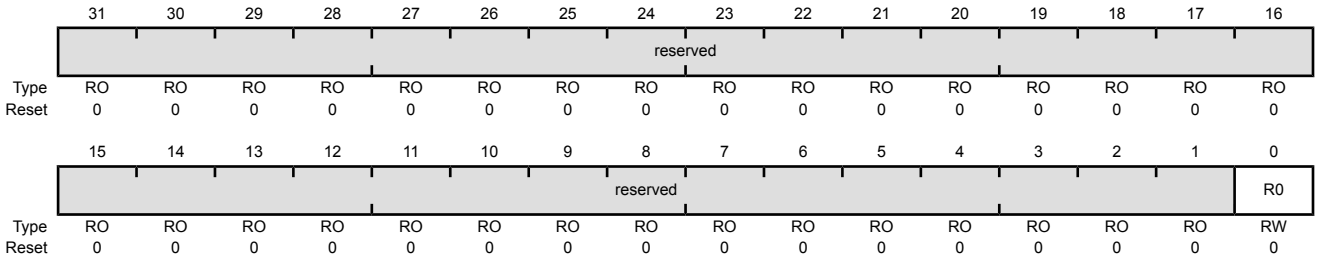
## Register 110: 1-Wire Run Mode Clock Gating Control (RCGCOWIRE), offset 0x698

The **RCGCOWIRE** register provides software the capability to enable and disable the 1-Wire module in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the 1-Wire module.

1-Wire Run Mode Clock Gating Control (RCGCOWIRE)

Base 0x400F.E000
Offset 0x698
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | 1-Wire Module 0 Run Mode Clock Gating Control<br>Value Description<br>0 1-Wire module 0 is disabled.<br>1 Enable and provide a clock to 1-Wire module 0 in Run mode. |

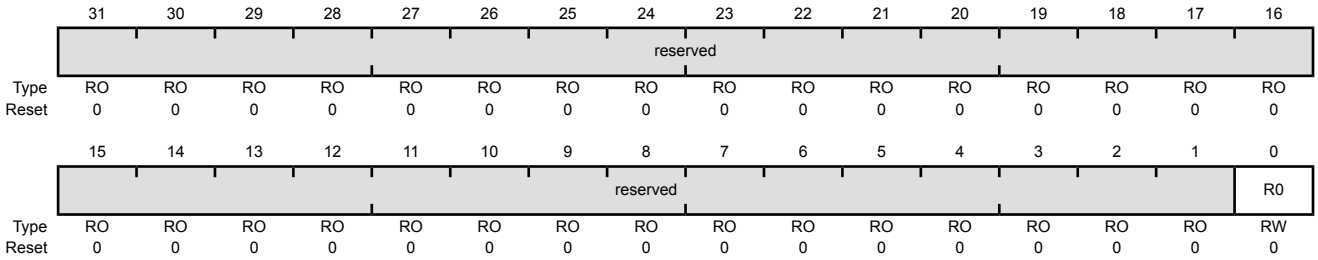
## Register 111: Ethernet MAC Run Mode Clock Gating Control (RCGCEMAC), offset 0x69C

The **RCGCEMAC** register provides software the capability to enable and disable the Ethernet MAC module in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

**Important:** This register should be used to control the clocking for the Ethernet Controller module.

Ethernet MAC Run Mode Clock Gating Control (RCGCEMAC)

Base 0x400F.E000
Offset 0x69C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RW | 0 | Ethernet MAC Module 0 Run Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Ethernet MAC module 0 is disabled. |
| 1 | Enable and provide a clock to Ethernet MAC module 0 in Run mode. |

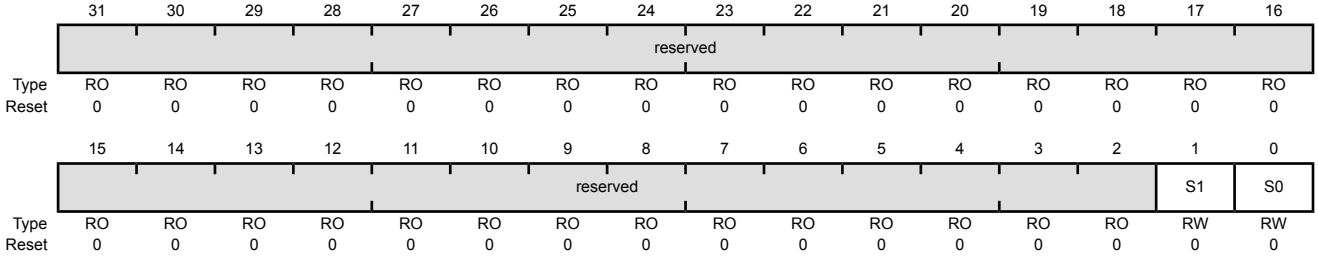
## Register 112: Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD), offset 0x700

The **SCGCWD** register provides software the capability to enable and disable watchdog modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the watchdog modules.

Watchdog Timer Sleep Mode Clock Gating Control (SCGCWD)

Base 0x400F.E000
Offset 0x700
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | S1 | RW | 0 | Watchdog Timer 1 Sleep Mode Clock Gating Control<br>Value Description<br>0 Watchdog module 1 is disabled in sleep mode.<br>1 Enable and provide a clock to Watchdog module 1 in sleep mode. |
| 0 | S0 | RW | 0 | Watchdog Timer 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 Watchdog module 0 is disabled in sleep mode.<br>1 Enable and provide a clock to Watchdog module 0 in sleep mode. |

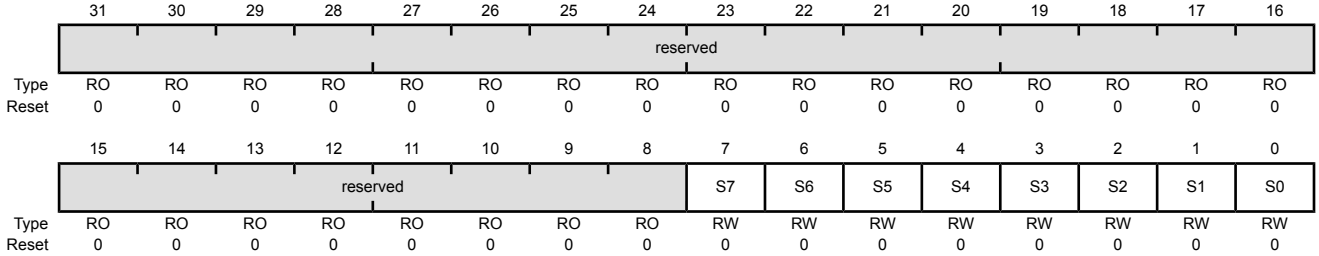
## Register 113: 16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control (SCGCTIMER), offset 0x704

The **SCGCGPT32** register provides software the capability to enable and disable 16/32-bit timer modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the timer modules.

16/32-Bit General-Purpose Timer Sleep Mode Clock Gating Control (SCGCTIMER)

Base 0x400F.E000
Offset 0x704
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | S7 | RW | 0 | 16/32-Bit General-Purpose Timer 7 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 7 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 7 in sleep mode. |
| 6 | S6 | RW | 0 | 16/32-Bit General-Purpose Timer 6 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 6 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 6 in sleep mode. |
| 5 | S5 | RW | 0 | 16/32-Bit General-Purpose Timer 5 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 5 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 5 in sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | S4 | RW | 0 | 16/32-Bit General-Purpose Timer 4 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 4 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 4 in sleep mode. |
| 3 | S3 | RW | 0 | 16/32-Bit General-Purpose Timer 3 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 3 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 3 in sleep mode. |
| 2 | S2 | RW | 0 | 16/32-Bit General-Purpose Timer 2 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 2 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 2 in sleep mode. |
| 1 | S1 | RW | 0 | 16/32-Bit General-Purpose Timer 1 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 1 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 1 in sleep mode. |
| 0 | S0 | RW | 0 | 16/32-Bit General-Purpose Timer 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 0 is disabled in sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 0 in sleep mode. |

## Register 114: General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO), offset 0x708

The **SCGCGPIO** register provides software the capability to enable and disable GPIO modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the GPIO modules.

General-Purpose Input/Output Sleep Mode Clock Gating Control (SCGCGPIO)

Base 0x400F.E000
Offset 0x708
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | S17 | S16 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S15 | S14 | S13 | S12 | S11 | S10 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | S17 | RW | 0 | GPIO Port T Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port T is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port T in sleep mode. |
| 16 | S16 | RW | 0 | GPIO Port S Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port S is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port S in sleep mode. |
| 15 | S15 | RW | 0 | GPIO Port R Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port R is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port R in sleep mode. |
| 14 | S14 | RW | 0 | GPIO Port Q Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port Q is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port Q in sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | S13 | RW | 0 | GPIO Port P Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port P is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port P in sleep mode. |
| 12 | S12 | RW | 0 | GPIO Port N Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port N is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port N in sleep mode. |
| 11 | S11 | RW | 0 | GPIO Port M Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port M is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port M in sleep mode. |
| 10 | S10 | RW | 0 | GPIO Port L Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port L is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port L in sleep mode. |
| 9 | S9 | RW | 0 | GPIO Port K Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port K is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port K in sleep mode. |
| 8 | S8 | RW | 0 | GPIO Port J Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port J is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port J in sleep mode. |
| 7 | S7 | RW | 0 | GPIO Port H Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port H is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port H in sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | S6 | RW | 0 | GPIO Port G Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port G is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port G in sleep mode. |
| 5 | S5 | RW | 0 | GPIO Port F Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port F is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port F in sleep mode. |
| 4 | S4 | RW | 0 | GPIO Port E Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port E is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port E in sleep mode. |
| 3 | S3 | RW | 0 | GPIO Port D Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port D is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port D in sleep mode. |
| 2 | S2 | RW | 0 | GPIO Port C Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port C is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port C in sleep mode. |
| 1 | S1 | RW | 0 | GPIO Port B Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port B is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port B in sleep mode. |
| 0 | S0 | RW | 0 | GPIO Port A Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port A is disabled in sleep mode.<br>1 Enable and provide a clock to GPIO Port A in sleep mode. |

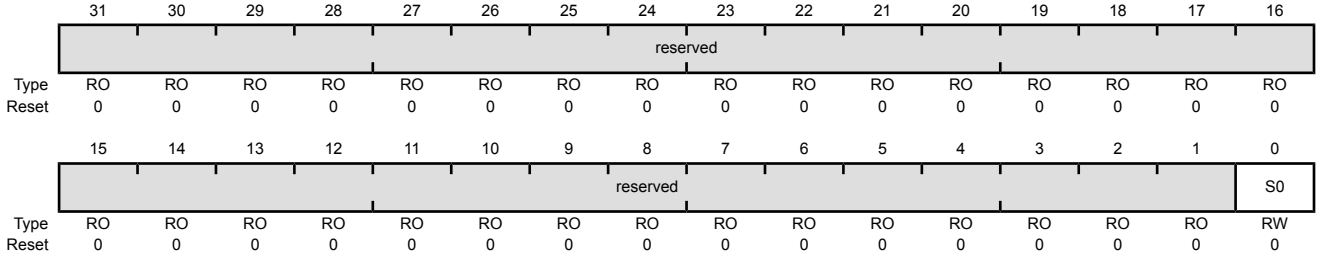
## Register 115: Micro Direct Memory Access Sleep Mode Clock Gating Control (SCGCDMA), offset 0x70C

The **SCGCDMA** register provides software the capability to enable and disable the µDMA module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the µDMA module.

Micro Direct Memory Access Sleep Mode Clock Gating Control (SCGCDMA)

Base 0x400F.E000
Offset 0x70C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | µDMA Module Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | µDMA module is disabled in sleep mode. |
| 1 | Enable and provide a clock to the µDMA module in sleep mode. |

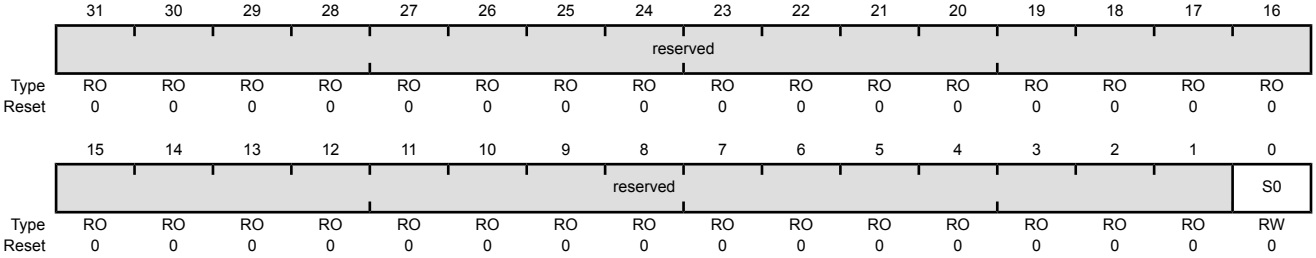
## Register 116: EPI Sleep Mode Clock Gating Control (SCGCEPI), offset 0x710

The **SCGCEPI** register provides software the capability to enable and disable the EPI module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the EPI module.

EPI Sleep Mode Clock Gating Control (SCGCEPI)

Base 0x400F.E000
Offset 0x710
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | EPI Module Sleep Mode Clock Gating Control<br>Value Description<br>0 EPI module is disabled in sleep mode.<br>1 Enable and provide a clock to the EPI module in sleep mode. |

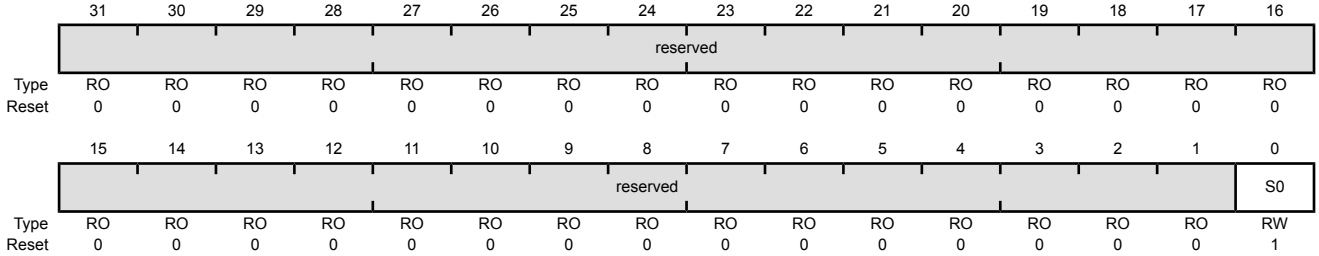
## Register 117: Hibernation Sleep Mode Clock Gating Control (SCGCHIB), offset 0x714

The **SCGCHIB** register provides software the capability to enable and disable the Hibernation module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the Hibernation module.

Hibernation Sleep Mode Clock Gating Control (SCGCHIB)

Base 0x400F.E000
Offset 0x714
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 1 | Hibernation Module Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Hibernation module is disabled in sleep mode. |
| 1 | Enable and provide a clock to the Hibernation module in sleep mode. |

## Register 118: Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control (SCGCUART), offset 0x718

The **SCGCUART** register provides software the capability to enable and disable the UART modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the UART modules.

Universal Asynchronous Receiver/Transmitter Sleep Mode Clock Gating Control (SCGCUART)

Base 0x400F.E000
Offset 0x718
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | S7 | RW | 0 | UART Module 7 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 7 is disabled in sleep mode.<br>1 Enable and provide a clock to UART module 7 in sleep mode. |
| 6 | S6 | RW | 0 | UART Module 6 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 6 is disabled in sleep mode.<br>1 Enable and provide a clock to UART module 6 in sleep mode. |
| 5 | S5 | RW | 0 | UART Module 5 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 5 is disabled in sleep mode.<br>1 Enable and provide a clock to UART module 5 in sleep mode. |
| 4 | S4 | RW | 0 | UART Module 4 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 4 is disabled.<br>1 Enable and provide a clock to UART module 4 in sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | S3 | RW | 0 | UART Module 3 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 3 is disabled in sleep mode.<br>1 Enable and provide a clock to UART module 3 in sleep mode. |
| 2 | S2 | RW | 0 | UART Module 2 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 2 is disabled in sleep mode.<br>1 Enable and provide a clock to UART module 2 in sleep mode. |
| 1 | S1 | RW | 0 | UART Module 1 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 1 is disabled in sleep mode.<br>1 Enable and provide a clock to UART module 1 in sleep mode. |
| 0 | S0 | RW | 0 | UART Module 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 0 is disabled.<br>1 Enable and provide a clock to UART module 0 in sleep mode. |

## Register 119: Synchronous Serial Interface Sleep Mode Clock Gating Control (SCGCSSI), offset 0x71C

The **SCGCSSI** register provides software the capability to enable and disable the SSI modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the SSI modules.

Synchronous Serial Interface Sleep Mode Clock Gating Control (SCGCSSI)

Base 0x400F.E000
Offset 0x71C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | S3 | S2 | S1 | S0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | S3 | RW | 0 | SSI Module 3 Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 3 is disabled in sleep mode.<br>1 Enable and provide a clock to SSI module 3 in sleep mode. |
| 2 | S2 | RW | 0 | SSI Module 2 Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 2 is disabled in sleep mode.<br>1 Enable and provide a clock to SSI module 2 in sleep mode. |
| 1 | S1 | RW | 0 | SSI Module 1 Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 1 is disabled in sleep mode.<br>1 Enable and provide a clock to SSI module 1 in sleep mode. |
| 0 | S0 | RW | 0 | SSI Module 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 0 is disabled in sleep mode.<br>1 Enable and provide a clock to SSI module 0 in sleep mode. |

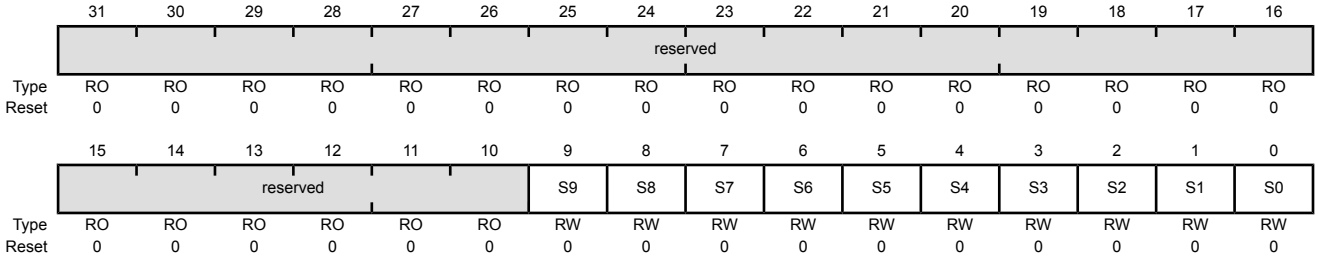
## Register 120: Inter-Integrated Circuit Sleep Mode Clock Gating Control (SCGCI2C), offset 0x720

The **SCGCI2C** register provides software the capability to enable and disable the I$^2$C modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the I$^2$C modules.

Inter-Integrated Circuit Sleep Mode Clock Gating Control (SCGCI2C)

Base 0x400F.E000
Offset 0x720
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |
| 9 | S9 | RW | 0 | I$^2$C Module 9 Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 9 is disabled in sleep mode.<br>1 Enable and provide a clock to I$^2$C module 9 in sleep mode. |
| 8 | S8 | RW | 0 | I$^2$C Module 8 Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 8 is disabled in sleep mode.<br>1 Enable and provide a clock to I$^2$C module 8 in sleep mode. |
| 7 | S7 | RW | 0 | I$^2$C Module 7 Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 7 is disabled in sleep mode.<br>1 Enable and provide a clock to I$^2$C module 7 in sleep mode. |
| 6 | S6 | RW | 0 | I$^2$C Module 6 Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 6 is disabled in sleep mode.<br>1 Enable and provide a clock to I$^2$C module 6 in sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | S5 | RW | 0 | I²C Module 5 Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 5 is disabled in sleep mode.<br>1 Enable and provide a clock to I²C module 5 in sleep mode. |
| 4 | S4 | RW | 0 | I²C Module 4 Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 4 is disabled in sleep mode.<br>1 Enable and provide a clock to I²C module 4 in sleep mode. |
| 3 | S3 | RW | 0 | I²C Module 3 Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 3 is disabled in sleep mode.<br>1 Enable and provide a clock to I²C module 3 in sleep mode. |
| 2 | S2 | RW | 0 | I²C Module 2 Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 2 is disabled in sleep mode.<br>1 Enable and provide a clock to I²C module 2 in sleep mode. |
| 1 | S1 | RW | 0 | I²C Module 1 Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 1 is disabled in sleep mode.<br>1 Enable and provide a clock to I²C module 1 in sleep mode. |
| 0 | S0 | RW | 0 | I²C Module 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 0 is disabled.<br>1 Enable and provide a clock to I²C module 0 in sleep mode. |

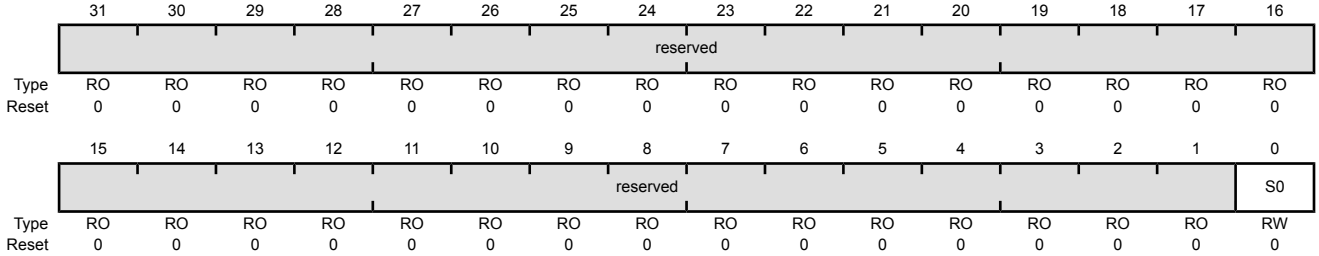
## Register 121: Universal Serial Bus Sleep Mode Clock Gating Control (SCGCUSB), offset 0x728

The **SCGCUSB** register provides software the capability to enable and disable the USB module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the USB module.

Universal Serial Bus Sleep Mode Clock Gating Control (SCGCUSB)

Base 0x400F.E000
Offset 0x728
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | USB Module Sleep Mode Clock Gating Control<br>Value Description<br>0 USB module is disabled in sleep mode.<br>1 Enable and provide a clock to the USB module in sleep mode. |

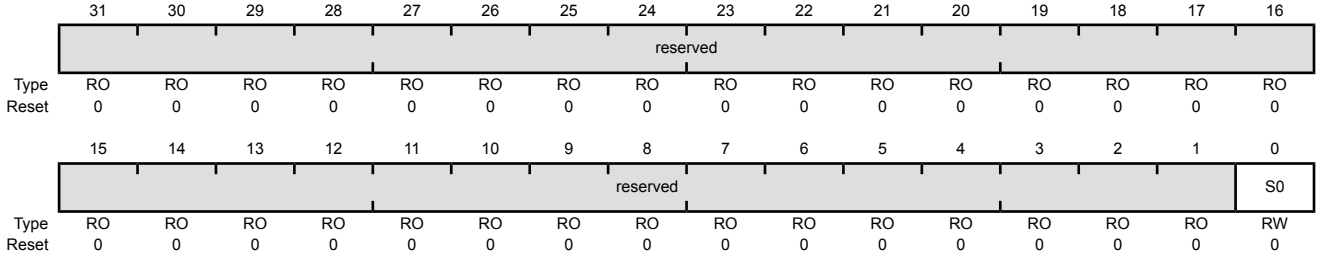
## Register 122: Ethernet PHY Sleep Mode Clock Gating Control (SCGCEPHY), offset 0x730

The **SCGCEPHY** register provides software the capability to enable and disable the PHY module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the PHY module.

Ethernet PHY Sleep Mode Clock Gating Control (SCGCEPHY)

Base 0x400F.E000
Offset 0x730
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | PHY Module Sleep Mode Clock Gating Control<br>Value Description<br>0 PHY module is disabled in sleep mode.<br>1 Enable and provide a clock to the PHY module in sleep mode. |

## Register 123: Controller Area Network Sleep Mode Clock Gating Control (SCGCCAN), offset 0x734

The **SCGCCAN** register provides software the capability to enable and disable the CAN modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the CAN modules.

Controller Area Network Sleep Mode Clock Gating Control (SCGCCAN)

Base 0x400F.E000
Offset 0x734
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | S1 | S0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | S1 | RW | 0 | CAN Module 1 Sleep Mode Clock Gating Control<br>Value Description<br>0 CAN module 1 is disabled in sleep mode.<br>1 Enable and provide a clock to CAN module 1 in sleep mode. |
| 0 | S0 | RW | 0 | CAN Module 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 CAN module 0 is disabled.<br>1 Enable and provide a clock to CAN module 0 in sleep mode. |

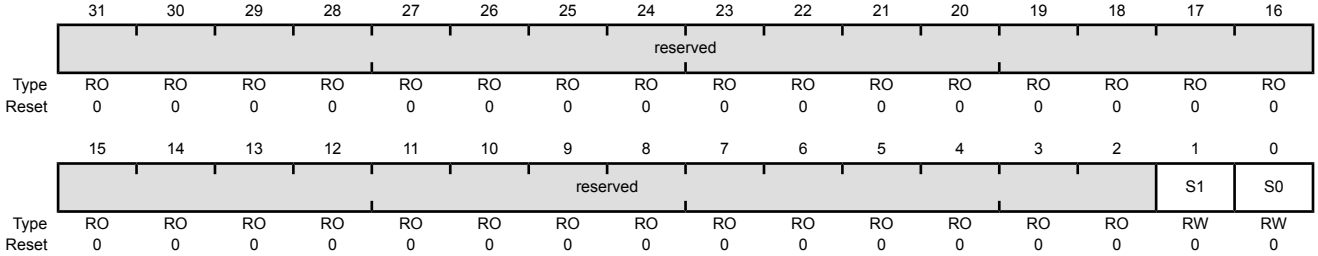
## Register 124: Analog-to-Digital Converter Sleep Mode Clock Gating Control (SCGCADC), offset 0x738

The **SCGCADC** register provides software the capability to enable and disable the ADC modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the ADC modules.

Analog-to-Digital Converter Sleep Mode Clock Gating Control (SCGCADC)

Base 0x400F.E000
Offset 0x738
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | S1 | RW | 0 | ADC Module 1 Sleep Mode Clock Gating Control<br>Value Description<br>0 ADC module 1 is disabled in sleep mode.<br>1 Enable and provide a clock to ADC module 1 in sleep mode. |
| 0 | S0 | RW | 0 | ADC Module 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 ADC module 0 is disabled in sleep mode.<br>1 Enable and provide a clock to ADC module 0 in sleep mode. |

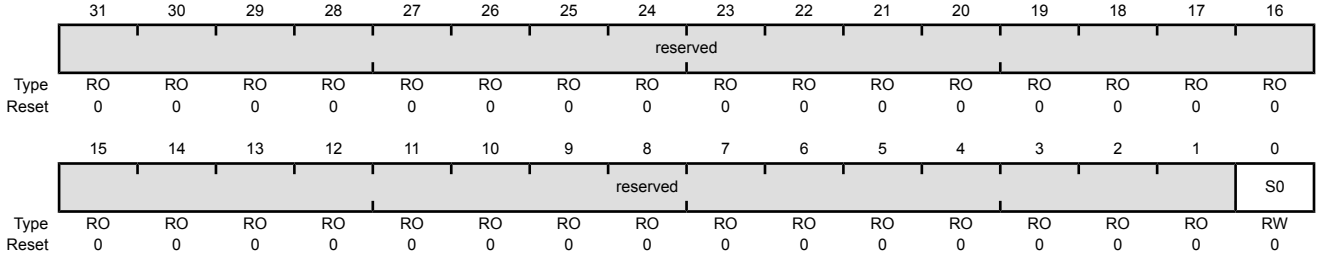
## Register 125: Analog Comparator Sleep Mode Clock Gating Control (SCGCACMP), offset 0x73C

The **SCGCACMP** register provides software the capability to enable and disable the analog comparator module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the analog comparator module.

Analog Comparator Sleep Mode Clock Gating Control (SCGCACMP)

Base 0x400F.E000
Offset 0x73C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | Analog Comparator Module 0 Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Analog comparator module is disabled in sleep mode. |
| 1 | Enable and provide a clock to the analog comparator module in sleep mode. |

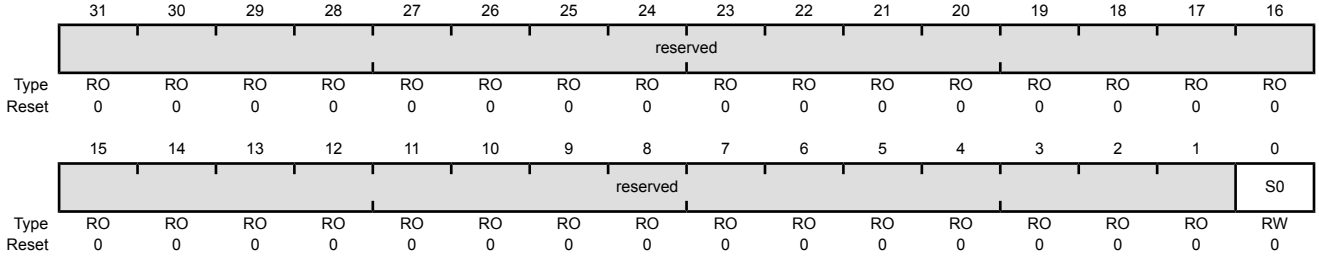
## Register 126: Pulse Width Modulator Sleep Mode Clock Gating Control (SCGCPWM), offset 0x740

The **SCGCPWM** register provides software the capability to enable and disable the PWM modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the PWM modules.

Pulse Width Modulator Sleep Mode Clock Gating Control (SCGCPWM)

Base 0x400F.E000
Offset 0x740
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | PWM Module 0 Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | PWM module 0 is disabled in sleep mode. |
| 1 | Enable and provide a clock to PWM module 0 in sleep mode. |

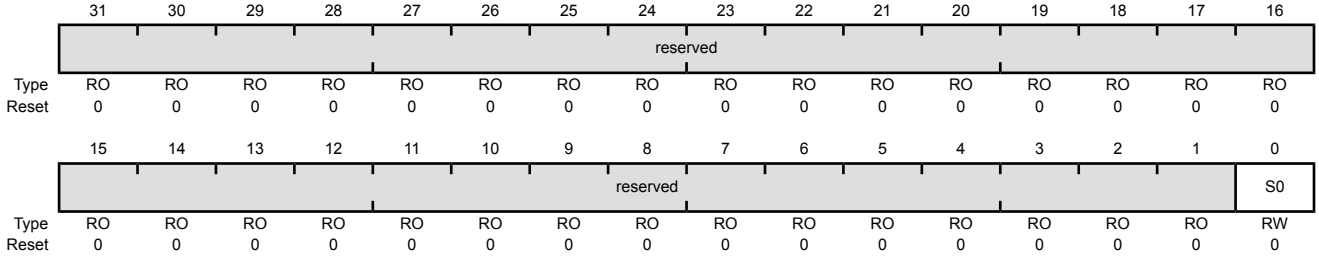
## Register 127: Quadrature Encoder Interface Sleep Mode Clock Gating Control (SCGCQEI), offset 0x744

The **SCGCQEI** register provides software the capability to enable and disable the QEI modules in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the QEI modules.

Quadrature Encoder Interface Sleep Mode Clock Gating Control (SCGCQEI)

Base 0x400F.E000
Offset 0x744
Type RW, reset 0x0000.0000



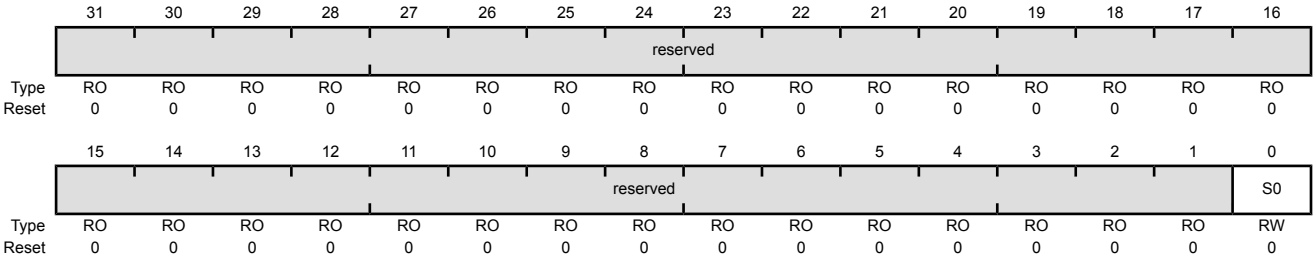

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | S0 | RW | 0 | QEI Module 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 QEI module 0 is disabled in sleep mode.<br>1 Enable and provide a clock to QEI module 0 in sleep mode. |

## Register 128: EEPROM Sleep Mode Clock Gating Control (SCGCEEPROM), offset 0x758

The **SCGCEEPROM** register provides software the capability to enable and disable the EEPROM module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

EEPROM Sleep Mode Clock Gating Control (SCGCEEPROM)

Base 0x400F.E000
Offset 0x758
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | EEPROM Module 0 Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | EEPROM module is disabled. |
| 1 | Enable and provide a clock to the EEPROM module in sleep mode. |

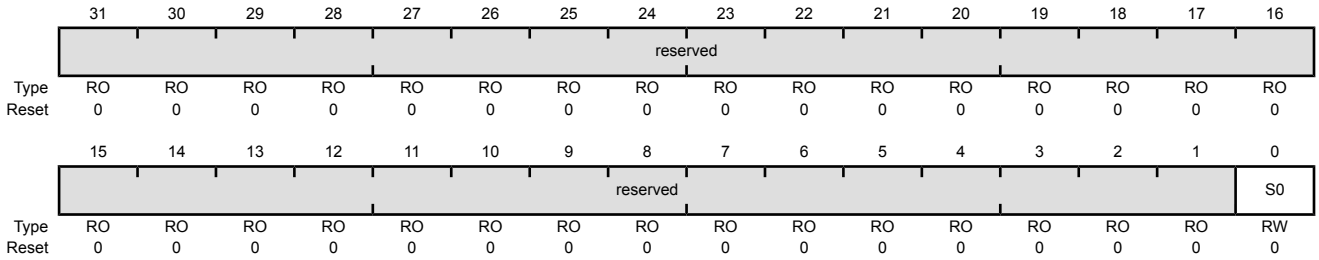
## Register 129: CRC and Cryptographic Modules Sleep Mode Clock Gating Control (SCGCCCM), offset 0x774

The **SCGCCCM** register provides software the capability to enable and disable the CRC and Encryption Control, AES, DES, and SHA/MD5 modules in sleep mode. When enabled, the modules are provided a clock . When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the CRC, AES, DES, and SHA/MD5 modules.

CRC and Cryptographic Modules Sleep Mode Clock Gating Control (SCGCCCM)

Base 0x400F.E000
Offset 0x774
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | CRC and Cryptographic Modules Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | The CRC, AES, DES, and SHA/MD5 modules are disabled in sleep mode. |
| 1 | Enable and provide a clock to the CRC, AES, DES, and SHA/MD5 modules in sleep mode. |

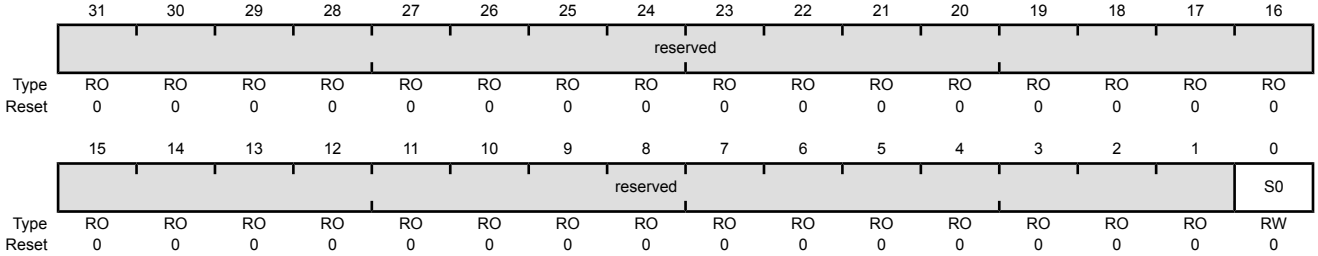
## Register 130: LCD Controller Sleep Mode Clock Gating Control (SCGCLCD), offset 0x790

The **SCGCLCD** register provides software the capability to enable and disable the LCD Controller module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the LCD Controller module.

LCD Controller Sleep Mode Clock Gating Control (SCGCLCD)

Base 0x400F.E000
Offset 0x790
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | LCD Controller Module 0 Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | LCD Controller module 0 is disabled in sleep mode. |
| 1 | Enable and provide a clock to LCD Controller module 0 in sleep mode. |

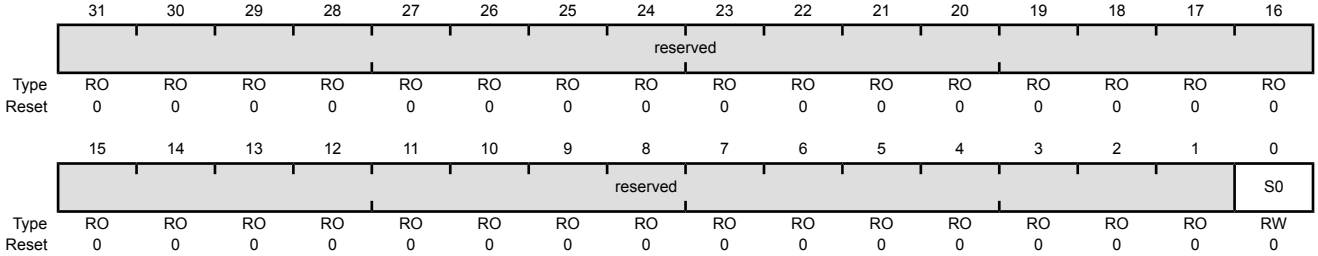
## Register 131: 1-Wire Sleep Mode Clock Gating Control (SCGCOWIRE), offset 0x798

The **SCGCOWIRE** register provides software the capability to enable and disable the 1-Wire module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the 1-Wire module.

1-Wire Sleep Mode Clock Gating Control (SCGCOWIRE)

Base 0x400F.E000
Offset 0x798
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | 1-Wire Module 0 Sleep Mode Clock Gating Control<br>Value Description<br>0 1-Wire module 0 is disabled in sleep mode.<br>1 Enable and provide a clock to 1-Wire module 0 in sleep mode. |

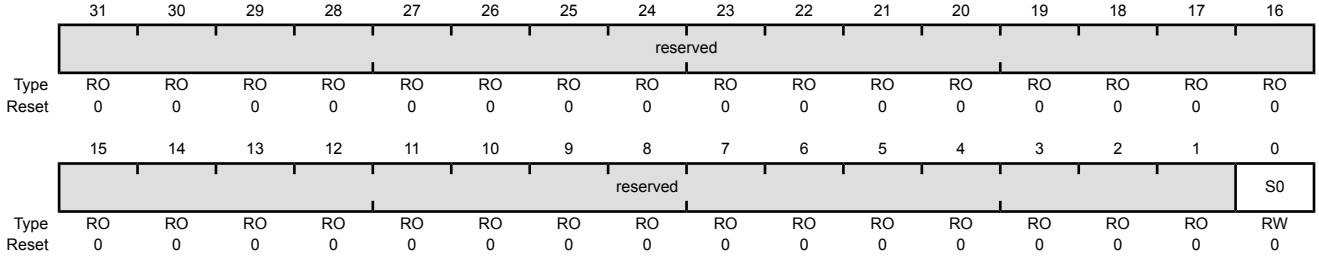
## Register 132: Ethernet MAC Sleep Mode Clock Gating Control (SCGCEMAC), offset 0x79C

The **SCGCEMAC** register provides software the capability to enable and disable the Ethernet MAC module in sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the Ethernet MAC module.

Ethernet MAC Sleep Mode Clock Gating Control (SCGCEMAC)

Base 0x400F.E000
Offset 0x79C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | S0 | RW | 0 | Ethernet MAC Module 0 Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Ethernet MAC module 0 is disabled in sleep mode. |
| 1 | Enable and provide a clock to Ethernet MAC module 0 in sleep mode. |

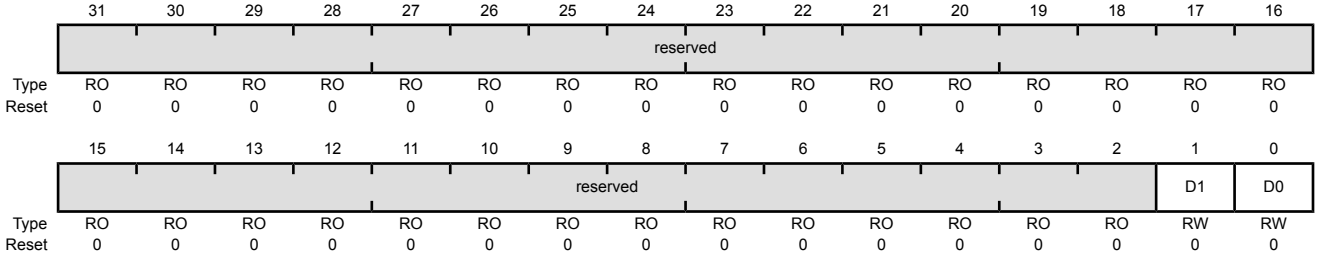
## Register 133: Watchdog Timer Deep-Sleep Mode Clock Gating Control (DCGCWD), offset 0x800

The **DCGCWD** register provides software the capability to enable and disable watchdog modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the watchdog modules.

Watchdog Timer Deep-Sleep Mode Clock Gating Control (DCGCWD)

Base 0x400F.E000
Offset 0x800
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | D1 | RW | 0 | Watchdog Timer 1 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 Watchdog module 1 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to Watchdog module 1 in deep-sleep mode. |
| 0 | D0 | RW | 0 | Watchdog Timer 0 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 Watchdog module 0 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to Watchdog module 0 in deep-sleep mode. |

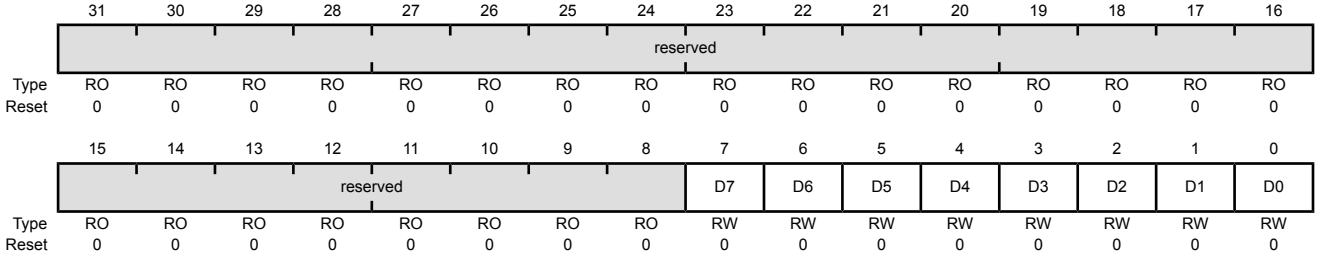
## Register 134: 16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCTIMER), offset 0x804

The **DCGCGPT32** register provides software the capability to enable and disable 16/32-bit timer modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the timer modules.

16/32-Bit General-Purpose Timer Deep-Sleep Mode Clock Gating Control (DCGCTIMER)

Base 0x400F.E000
Offset 0x804
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | D7 | RW | 0 | 16/32-Bit General-Purpose Timer 7 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 7 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 7 in deep-sleep mode. |
| 6 | D6 | RW | 0 | 16/32-Bit General-Purpose Timer 6 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 6 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 6 in deep-sleep mode. |
| 5 | D5 | RW | 0 | 16/32-Bit General-Purpose Timer 5 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 16/32-bit general-purpose timer module 5 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 5 in deep-sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | D4 | RW | 0 | 16/32-Bit General-Purpose Timer 4 Deep-Sleep Mode Clock Gating Control<br><br>Value Description<br>0 16/32-bit general-purpose timer module 4 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 4 in deep-sleep mode. |
| 3 | D3 | RW | 0 | 16/32-Bit General-Purpose Timer 3 Deep-Sleep Mode Clock Gating Control<br><br>Value Description<br>0 16/32-bit general-purpose timer module 3 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 3 in deep-sleep mode. |
| 2 | D2 | RW | 0 | 16/32-Bit General-Purpose Timer 2 Deep-Sleep Mode Clock Gating Control<br><br>Value Description<br>0 16/32-bit general-purpose timer module 2 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 2 in deep-sleep mode. |
| 1 | D1 | RW | 0 | 16/32-Bit General-Purpose Timer 1 Deep-Sleep Mode Clock Gating Control<br><br>Value Description<br>0 16/32-bit general-purpose timer module 1 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 1 in deep-sleep mode. |
| 0 | D0 | RW | 0 | 16/32-Bit General-Purpose Timer 0 Deep-Sleep Mode Clock Gating Control<br><br>Value Description<br>0 16/32-bit general-purpose timer module 0 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to 16/32-bit general-purpose timer module 0 in deep-sleep mode. |

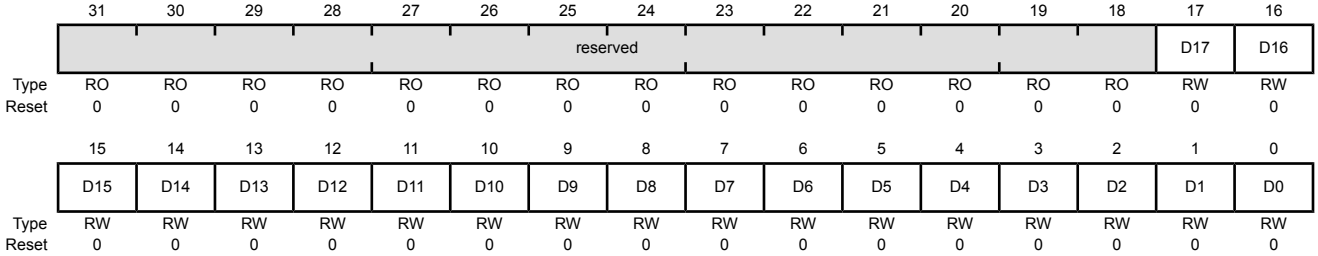
## Register 135: General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control (DCGCGPIO), offset 0x808

The **DCGCGPIO** register provides software the capability to enable and disable GPIO modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the GPIO modules.

General-Purpose Input/Output Deep-Sleep Mode Clock Gating Control (DCGCGPIO)

Base 0x400F.E000
Offset 0x808
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | D17 | RW | 0 | GPIO Port T Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port T is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port T in deep-sleep mode. |
| 16 | D16 | RW | 0 | GPIO Port S Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port S is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port S in deep-sleep mode. |
| 15 | D15 | RW | 0 | GPIO Port R Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port R is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port R in deep-sleep mode. |
| 14 | D14 | RW | 0 | GPIO Port Q Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port Q is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port Q in deep-sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | D13 | RW | 0 | GPIO Port P Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port P is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port P in deep-sleep mode. |
| 12 | D12 | RW | 0 | GPIO Port N Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port N is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port N in deep-sleep mode. |
| 11 | D11 | RW | 0 | GPIO Port M Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port M is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port M in deep-sleep mode. |
| 10 | D10 | RW | 0 | GPIO Port L Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port L is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port L in deep-sleep mode. |
| 9 | D9 | RW | 0 | GPIO Port K Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port K is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port K in deep-sleep mode. |
| 8 | D8 | RW | 0 | GPIO Port J Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port J is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port J in deep-sleep mode. |
| 7 | D7 | RW | 0 | GPIO Port H Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port H is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port H in deep-sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | D6 | RW | 0 | GPIO Port G Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port G is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port G in deep-sleep mode. |
| 5 | D5 | RW | 0 | GPIO Port F Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port F is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port F in deep-sleep mode. |
| 4 | D4 | RW | 0 | GPIO Port E Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port E is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port E in deep-sleep mode. |
| 3 | D3 | RW | 0 | GPIO Port D Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port D is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port D in deep-sleep mode. |
| 2 | D2 | RW | 0 | GPIO Port C Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port C is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port C in deep-sleep mode. |
| 1 | D1 | RW | 0 | GPIO Port B Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port B is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port B in deep-sleep mode. |
| 0 | D0 | RW | 0 | GPIO Port A Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 GPIO Port A is disabled in deep-sleep mode.<br>1 Enable and provide a clock to GPIO Port A in deep-sleep mode. |

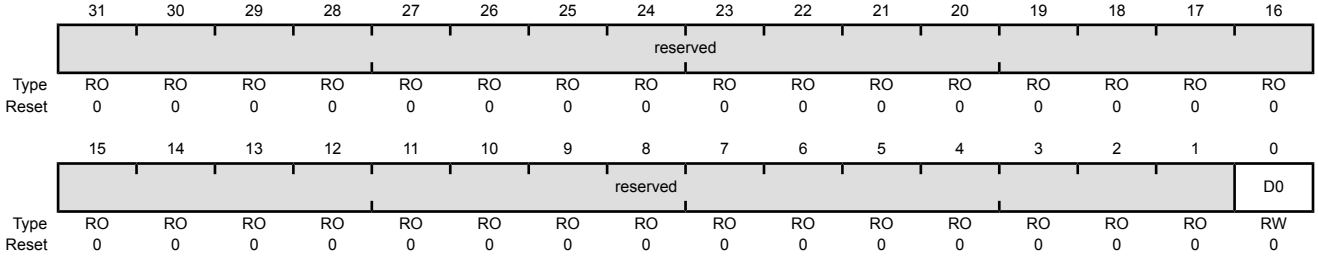
## Register 136: Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control (DCGCDMA), offset 0x80C

The **DCGCDMA** register provides software the capability to enable and disable the µDMA module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the µDMA module.

Micro Direct Memory Access Deep-Sleep Mode Clock Gating Control (DCGCDMA)

Base 0x400F.E000
Offset 0x80C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | µDMA Module Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | µDMA module is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the µDMA module in deep-sleep mode. |

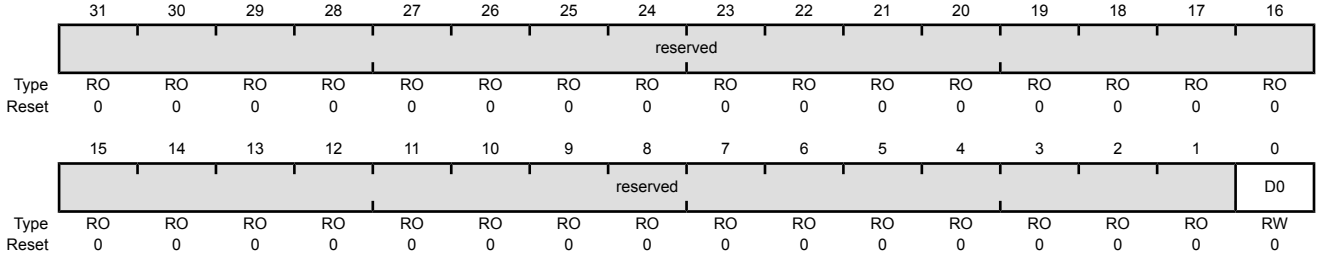
## Register 137: EPI Deep-Sleep Mode Clock Gating Control (DCGCEPI), offset 0x810

The **DCGCEPI** register provides software the capability to enable and disable the EPI module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the EPI module.

EPI Deep-Sleep Mode Clock Gating Control (DCGCEPI)

Base 0x400F.E000
Offset 0x810
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | EPI Module Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | EPI module is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the EPI module in deep-sleep mode. |

## Register 138: Hibernation Deep-Sleep Mode Clock Gating Control (DCGCHIB), offset 0x814

The **DCGCHIB** register provides software the capability to enable and disable the Hibernation module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the Hibernation module.

Hibernation Deep-Sleep Mode Clock Gating Control (DCGCHIB)

Base 0x400F.E000
Offset 0x814
Type RW, reset 0x0000.0001

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | D0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 1 | Hibernation Module Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Hibernation module is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the Hibernation module in deep-sleep mode. |

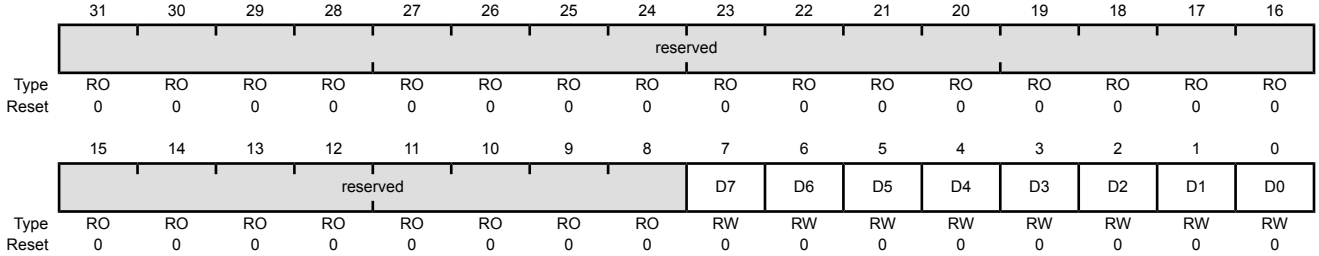
## Register 139: Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control (DCGCUART), offset 0x818

The **DCGCUART** register provides software the capability to enable and disable the UART modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the UART modules.

Universal Asynchronous Receiver/Transmitter Deep-Sleep Mode Clock Gating Control (DCGCUART)

Base 0x400F.E000
Offset 0x818
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | D7 | RW | 0 | UART Module 7 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 7 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 7 in deep-sleep mode. |
| 6 | D6 | RW | 0 | UART Module 6 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 6 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 6 in deep-sleep mode. |
| 5 | D5 | RW | 0 | UART Module 5 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 5 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 5 in deep-sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | D4 | RW | 0 | UART Module 4 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 4 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 4 in deep-sleep mode. |
| 3 | D3 | RW | 0 | UART Module 3 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 3 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 3 in deep-sleep mode. |
| 2 | D2 | RW | 0 | UART Module 2 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 2 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 2 in deep-sleep mode. |
| 1 | D1 | RW | 0 | UART Module 1 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 1 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 1 in deep-sleep mode. |
| 0 | D0 | RW | 0 | UART Module 0 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 UART module 0 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to UART module 0 in deep-sleep mode. |

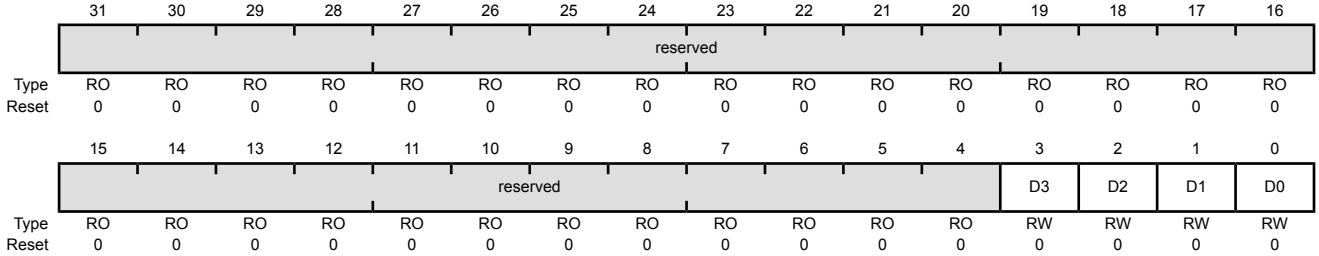
## Register 140: Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control (DCGCSSI), offset 0x81C

The **DCGCSSI** register provides software the capability to enable and disable the SSI modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the SSI modules.

Synchronous Serial Interface Deep-Sleep Mode Clock Gating Control (DCGCSSI)

Base 0x400F.E000
Offset 0x81C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | D3 | RW | 0 | SSI Module 3 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 3 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to SSI module 3 in deep-sleep mode. |
| 2 | D2 | RW | 0 | SSI Module 2 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 2 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to SSI module 2 in deep-sleep mode. |
| 1 | D1 | RW | 0 | SSI Module 1 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 1 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to SSI module 1 in deep-sleep mode. |
| 0 | D0 | RW | 0 | SSI Module 0 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 SSI module 0 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to SSI module 0 in deep-sleep mode. |

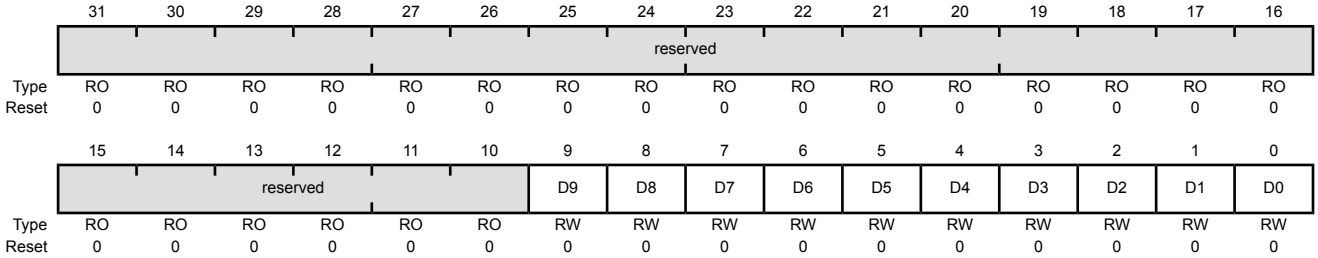
## Register 141: Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control (DCGCI2C), offset 0x820

The **DCGCI2C** register provides software the capability to enable and disable the I$^2$C modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the I$^2$C modules.

Inter-Integrated Circuit Deep-Sleep Mode Clock Gating Control (DCGCI2C)

Base 0x400F.E000
Offset 0x820
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |
| 9 | D9 | RW | 0 | I$^2$C Module 9 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 9 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I$^2$C module 9 in deep-sleep mode. |
| 8 | D8 | RW | 0 | I$^2$C Module 8 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 8 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I$^2$C module 8 in deep-sleep mode. |
| 7 | D7 | RW | 0 | I$^2$C Module 7 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 7 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I$^2$C module 7 in deep-sleep mode. |
| 6 | D6 | RW | 0 | I$^2$C Module 6 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I$^2$C module 6 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I$^2$C module 6 in deep-sleep mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | D5 | RW | 0 | I²C Module 5 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 5 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I²C module 5 in deep-sleep mode. |
| 4 | D4 | RW | 0 | I²C Module 4 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 4 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I²C module 4 in deep-sleep mode. |
| 3 | D3 | RW | 0 | I²C Module 3 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 3 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I²C module 3 in deep-sleep mode. |
| 2 | D2 | RW | 0 | I²C Module 2 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 2 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I²C module 2 in deep-sleep mode. |
| 1 | D1 | RW | 0 | I²C Module 1 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 1 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I²C module 1 in deep-sleep mode. |
| 0 | D0 | RW | 0 | I²C Module 0 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 I²C module 0 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to I²C module 0 in deep-sleep mode. |

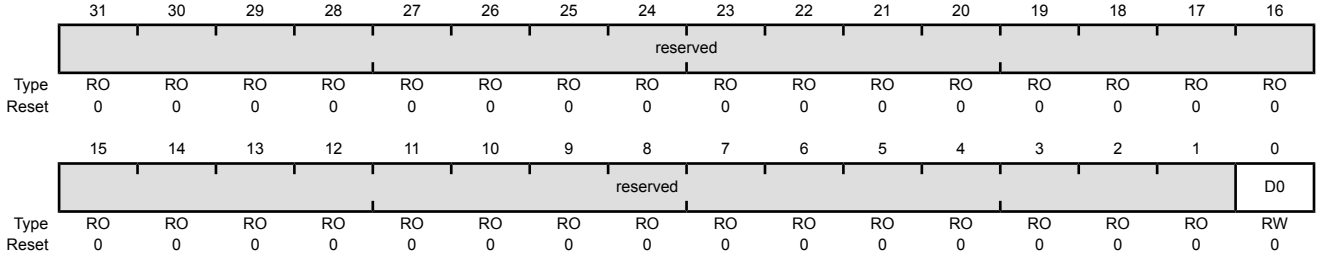
## Register 142: Universal Serial Bus Deep-Sleep Mode Clock Gating Control (DCGCUSB), offset 0x828

The **DCGCUSB** register provides software the capability to enable and disable the USB module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the USB module.

Universal Serial Bus Deep-Sleep Mode Clock Gating Control (DCGCUSB)

Base 0x400F.E000
Offset 0x828
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | USB Module Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | USB module is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the USB module in deep-sleep mode. |

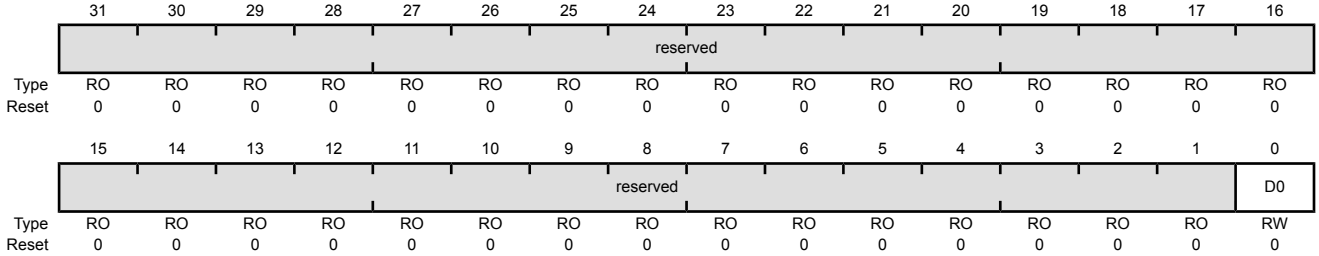
## Register 143: Ethernet PHY Deep-Sleep Mode Clock Gating Control (DCGCEPHY), offset 0x830

The **DCGCEPHY** register provides software the capability to enable and disable the PHY module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the PHY module.

Ethernet PHY Deep-Sleep Mode Clock Gating Control (DCGCEPHY)

Base 0x400F.E000
Offset 0x830
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | PHY Module Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | PHY module is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the PHY module in deep-sleep mode. |

## Register 144: Controller Area Network Deep-Sleep Mode Clock Gating Control (DCGCCAN), offset 0x834

The **DCGCCAN** register provides software the capability to enable and disable the CAN modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the CAN modules.

Controller Area Network Deep-Sleep Mode Clock Gating Control (DCGCCAN)

Base 0x400F.E000
Offset 0x834
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | D1 | D0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | D1 | RW | 0 | CAN Module 1 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 CAN module 1 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to CAN module 1 in deep-sleep mode. |
| 0 | D0 | RW | 0 | CAN Module 0 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 CAN module 0 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to CAN module 0 in deep-sleep mode. |

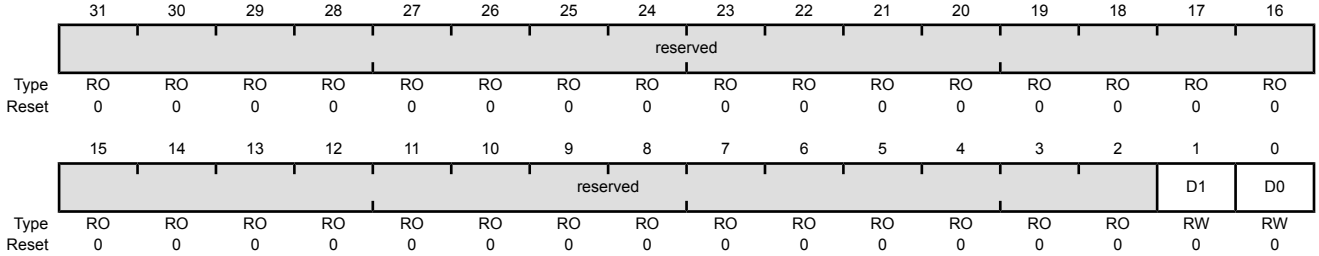
## Register 145: Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control (DCGCADC), offset 0x838

The **DCGCADC** register provides software the capability to enable and disable the ADC modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the ADC modules.

Analog-to-Digital Converter Deep-Sleep Mode Clock Gating Control (DCGCADC)

Base 0x400F.E000
Offset 0x838
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | D1 | RW | 0 | ADC Module 1 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 ADC module 1 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to ADC module 1 in deep-sleep mode. |
| 0 | D0 | RW | 0 | ADC Module 0 Deep-Sleep Mode Clock Gating Control<br>Value Description<br>0 ADC module 0 is disabled in deep-sleep mode.<br>1 Enable and provide a clock to ADC module 0 in deep-sleep mode. |

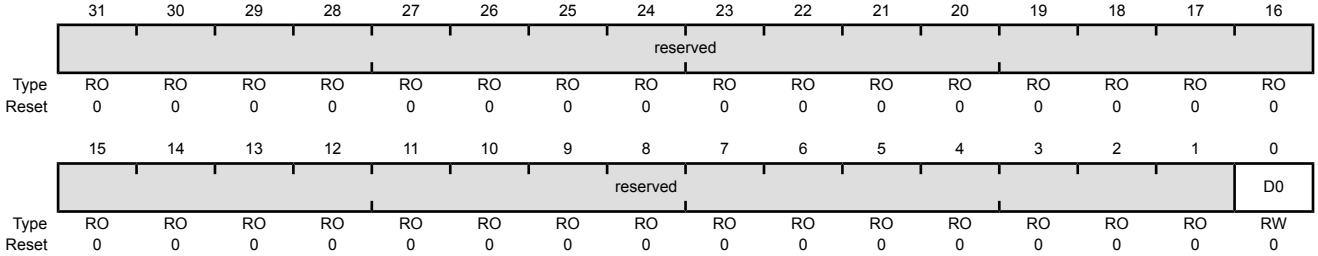
## Register 146: Analog Comparator Deep-Sleep Mode Clock Gating Control (DCGCACMP), offset 0x83C

The **DCGCACMP** register provides software the capability to enable and disable the analog comparator module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the analog comparator module.

Analog Comparator Deep-Sleep Mode Clock Gating Control (DCGCACMP)

Base 0x400F.E000
Offset 0x83C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | Analog Comparator Module 0 Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Analog comparator module is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the analog comparator module in deep-sleep mode. |

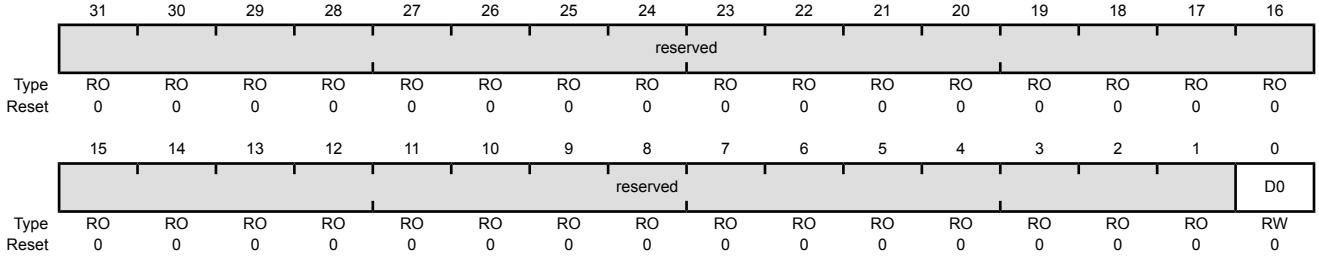
## Register 147: Pulse Width Modulator Deep-Sleep Mode Clock Gating Control (DCGCPWM), offset 0x840

The **DCGCPWM** register provides software the capability to enable and disable the PWM modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the PWM modules.

Pulse Width Modulator Deep-Sleep Mode Clock Gating Control (DCGCPWM)

Base 0x400F.E000
Offset 0x840
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | PWM Module 0 Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | PWM module 0 is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to PWM module 0 in deep-sleep mode. |

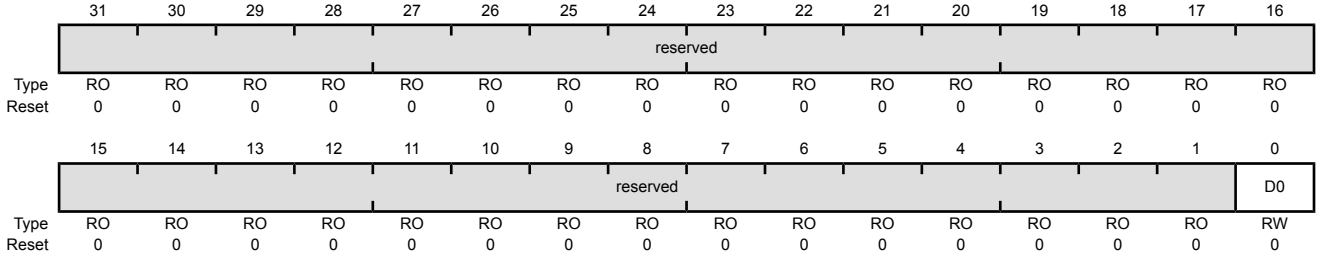
## Register 148: Quadrature Encoder Interface Deep-Sleep Mode Clock Gating Control (DCGCQEI), offset 0x844

The **DCGCQEI** register provides software the capability to enable and disable the QEI modules in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the QEI modules.

Quadrature Encoder Interface Deep-Sleep Mode Clock Gating Control (DCGCQEI)

Base 0x400F.E000
Offset 0x844
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | QEI Module 0 Deep-Sleep Mode Clock Gating Control |

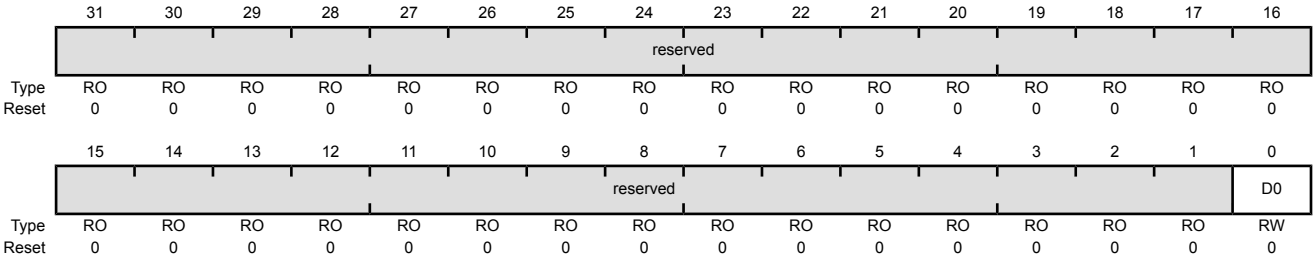
| Value | Description |
|---|---|
| 0 | QEI module 0 is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to QEI module 0 in deep-sleep mode. |

## Register 149: EEPROM Deep-Sleep Mode Clock Gating Control (DCGCEEPROM), offset 0x858

The **DCGCEEPROM** register provides software the capability to enable and disable the EEPROM module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

EEPROM Deep-Sleep Mode Clock Gating Control (DCGCEEPROM)

Base 0x400F.E000
Offset 0x858
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | EEPROM Module 0 Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | EEPROM module is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the EEPROM module in deep-sleep mode. |

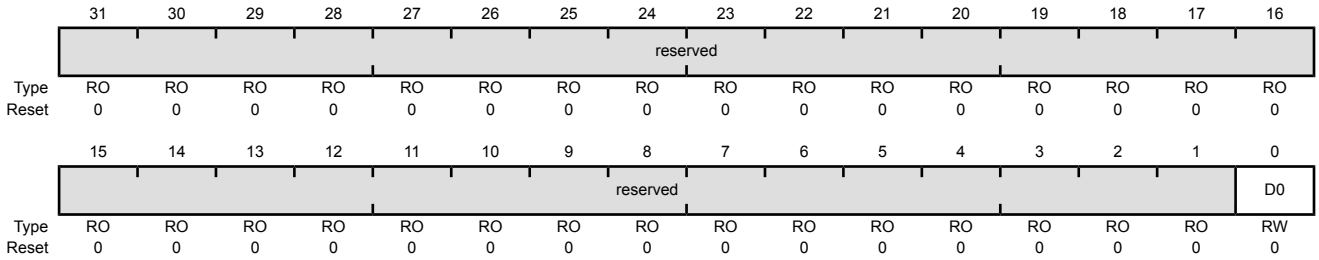
## Register 150: CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control (DCGCCCM), offset 0x874

The **DCGCCCM** register provides software the capability to enable and disable the CRC, AES, DES, and SHA/MD5 modules in deep-sleep mode. When enabled, the modules are provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the CRC, AES, DES, and SHA/MD modules.

CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control (DCGCCCM)

Base 0x400F.E000
Offset 0x874
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | The CRC, AES, DES, and SHA/MD5 modules are disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to the CRC, AES, DES, and SHA/MD5 modules in deep-sleep mode. |

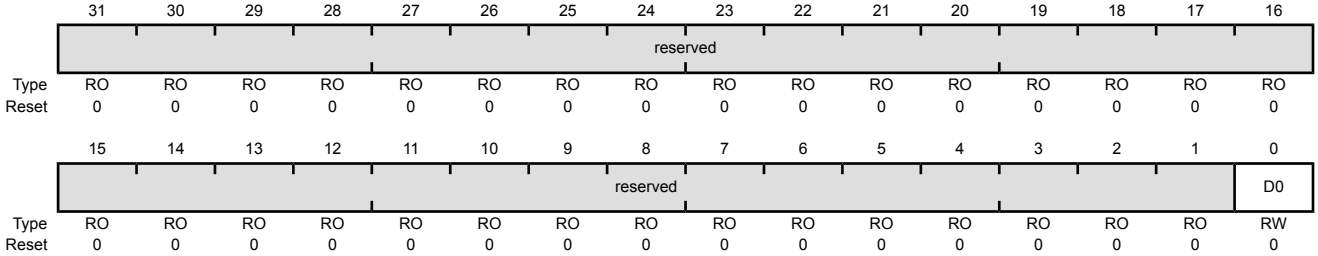
## Register 151: LCD Controller Deep-Sleep Mode Clock Gating Control (DCGCLCD), offset 0x890

The **DCGCLCD** register provides software the capability to enable and disable the LCD Controller module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the LCD Controller module.

LCD Controller Deep-Sleep Mode Clock Gating Control (DCGCLCD)

Base 0x400F.E000
Offset 0x890
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | LCD Controller Module 0 Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | LCD Controller module 0 is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to LCD Controller module 0 in deep-sleep mode. |

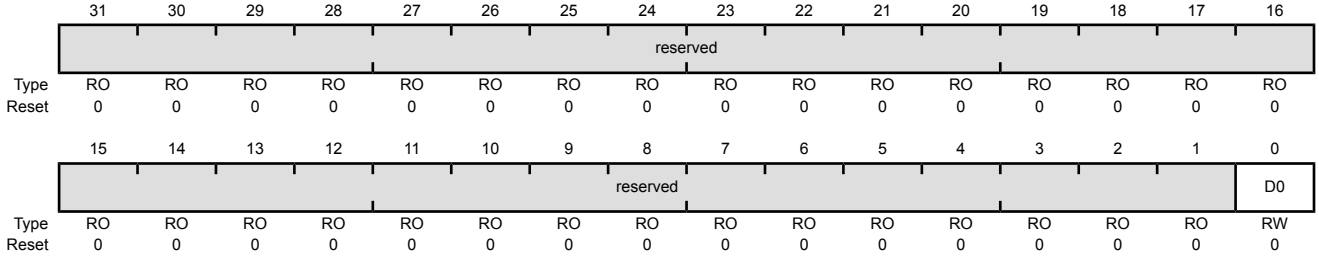
## Register 152: 1-Wire Deep-Sleep Mode Clock Gating Control (DCGCOWIRE), offset 0x898

The **DCGCOWIRE** register provides software the capability to enable and disable the 1-Wire module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the 1-Wire module.

1-Wire Deep-Sleep Mode Clock Gating Control (DCGCOWIRE)

Base 0x400F.E000
Offset 0x898
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | 1-Wire Module 0 Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | 1-Wire module 0 is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to 1-Wire module 0 in deep-sleep mode. |

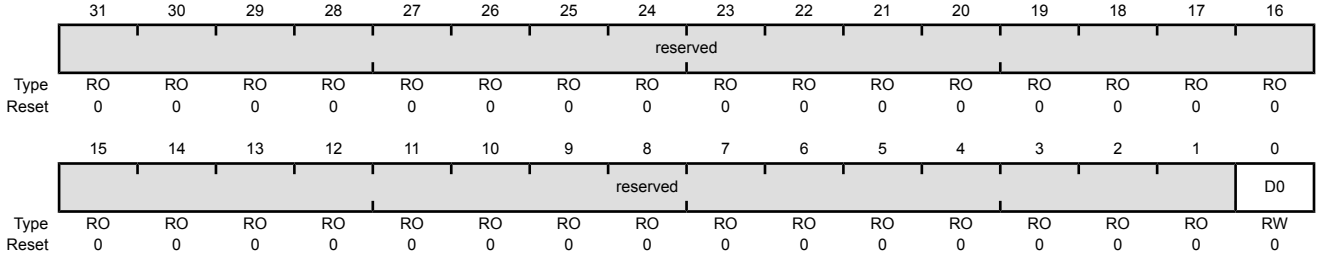
## Register 153: Ethernet MAC Deep-Sleep Mode Clock Gating Control (DCGCEMAC), offset 0x89C

The **DCGCEMAC** register provides software the capability to enable and disable the Ethernet MAC module in deep-sleep mode. When enabled, a module is provided a clock. When disabled, the clock is disabled to save power.

**Important:** This register should be used to control the clocking for the Ethernet MAC module.

Ethernet MAC Deep-Sleep Mode Clock Gating Control (DCGCEMAC)

Base 0x400F.E000
Offset 0x89C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | D0 | RW | 0 | Ethernet MAC Module 0 Deep-Sleep Mode Clock Gating Control |

| Value | Description |
|---|---|
| 0 | Ethernet MAC module 0 is disabled in deep-sleep mode. |
| 1 | Enable and provide a clock to Ethernet MAC module 0 in deep-sleep mode. |

## Register 154: Watchdog Timer Power Control (PCWD), offset 0x900

**Important:** The Watchdog Timer modules do not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCWD** register controls the power applied to the Watchdog Module module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCWD**, **SCGCWD** and **DCGCWD** registers. If the Rn, Sn, or Dn bit of the respective **RCGCWD**, **SCGCWD** and **DCGCWD** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCWD** register is.

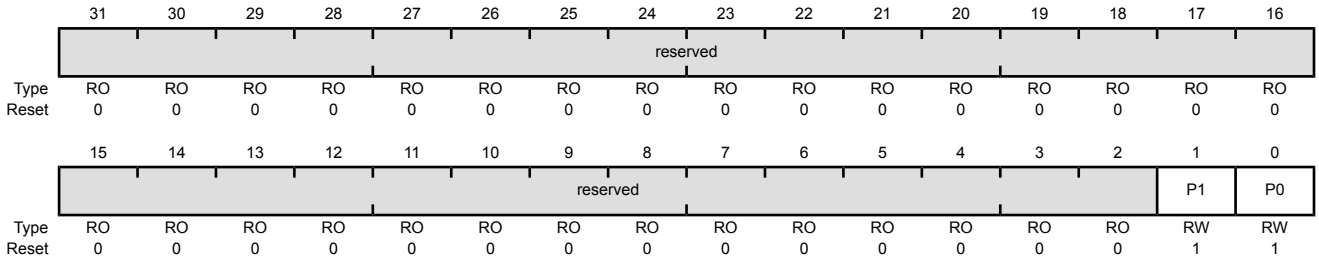
However, if the Rn, Sn, or Dn bit of the respective **RCGCWD**, **SCGCWD** and **DCGCWD** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCWD** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-16. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Watchdog Timer Power Control (PCWD)

Base 0x400F.E000
Offset 0x900
Type RW, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | P1 | RW | 1 | Watchdog Timer 1 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCWD**, **SCGCWD** or **DCGCWD** register is clear.<br>Value Description<br>0 Watchdog Timer 1 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Watchdog Timer 1 module is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | Watchdog Timer 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCWD**, **SCGCWD** or **DCGCWD** register is clear.<br>Value Description<br>0 Watchdog Timer 0 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Watchdog Timer 0 module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 155: 16/32-Bit General-Purpose Timer Power Control (PCTIMER), offset 0x904

**Important:** The Timer module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCTIMER** register controls the power applied to the Timer module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCTIMER**, **SCGCTIMER** and **DCGCTIMER** registers. If the Rn, Sn, or Dn bit of the respective **RCGCTIMER**, **SCGCTIMER** and **DCGCTIMER** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCTIMER** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCTIMER**, **SCGCTIMER** and **DCGCTIMER** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCTIMER** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-17. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

16/32-Bit General-Purpose Timer Power Control (PCTIMER)

Base 0x400F.E000
Offset 0x904
Type RW, reset 0x0000.00FF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | P7 | RW | 1 | General-Purpose Timer 7 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 7 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 7 module is powered, but does not receive a clock. In this case, the module is inactive. |
| 6 | P6 | RW | 1 | General-Purpose Timer 6 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 6 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 6 module is powered, but does not receive a clock. In this case, the module is inactive. |
| 5 | P5 | RW | 1 | General-Purpose Timer 5 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 5 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 5 module is powered, but does not receive a clock. In this case, the module is inactive. |
| 4 | P4 | RW | 1 | General-Purpose Timer 4 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 4 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 4 module is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | P3 | RW | 1 | General-Purpose Timer 3 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 3 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 3 module is powered, but does not receive a clock. In this case, the module is inactive. |
| 2 | P2 | RW | 1 | General-Purpose Timer 2 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 2 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 2 module is powered, but does not receive a clock. In this case, the module is inactive. |
| 1 | P1 | RW | 1 | General-Purpose Timer 1 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 1 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 1 module is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | General-Purpose Timer 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCTIMER**, **SCGCTIMER** or **DCGCTIMER** register is clear.<br>Value Description<br>0 Timer 0 module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Timer 0 module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 156: General-Purpose Input/Output Power Control (PCGPIO), offset 0x908

**Important:** The GPIO modules do not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCGPIO** register controls the power applied to the GPIO module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCGPIO**, **SCGCGPIO** and **DCGCGPIO** registers. If the Rn, Sn, or Dn bit of the respective **RCGCGPIO**, **SCGCGPIO** and **DCGCGPIO** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCGPIO** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCGPIO**, **SCGCGPIO** and **DCGCGPIO** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCGPIO** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-18. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

General-Purpose Input/Output Power Control (PCGPIO)

Base 0x400F.E000
Offset 0x908
Type RW, reset 0x0003.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | P17 | P16 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | P17 | RW | 1 | GPIO Port T Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port T is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port T is powered, but does not receive a clock. In this case, the module is inactive. |
| 16 | P16 | RW | 1 | GPIO Port S Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port S is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port S is powered, but does not receive a clock. In this case, the module is inactive. |
| 15 | P15 | RW | 1 | GPIO Port R Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port R is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port R is powered, but does not receive a clock. In this case, the module is inactive. |
| 14 | P14 | RW | 1 | GPIO Port Q Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port Q is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port Q is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | P13 | RW | 1 | GPIO Port P Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port P is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port P is powered, but does not receive a clock. In this case, the module is inactive. |
| 12 | P12 | RW | 1 | GPIO Port N Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port N is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port N is powered, but does not receive a clock. In this case, the module is inactive. |
| 11 | P11 | RW | 1 | GPIO Port M Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port M is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port M is powered, but does not receive a clock. In this case, the module is inactive. |
| 10 | P10 | RW | 1 | GPIO Port L Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port L is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port L is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | P9 | RW | 1 | GPIO Port K Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port K is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port K is powered, but does not receive a clock. In this case, the module is inactive. |
| 8 | P8 | RW | 1 | GPIO Port J Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port J is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port J is powered, but does not receive a clock. In this case, the module is inactive. |
| 7 | P7 | RW | 1 | GPIO Port H Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port H is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port H is powered, but does not receive a clock. In this case, the module is inactive. |
| 6 | P6 | RW | 1 | GPIO Port G Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port G is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port G is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | P5 | RW | 1 | GPIO Port F Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port F is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port F is powered, but does not receive a clock. In this case, the module is inactive. |
| 4 | P4 | RW | 1 | GPIO Port E Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port E is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port E is powered, but does not receive a clock. In this case, the module is inactive. |
| 3 | P3 | RW | 1 | GPIO Port D Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port D is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port D is powered, but does not receive a clock. In this case, the module is inactive. |
| 2 | P2 | RW | 1 | GPIO Port C Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port C is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port C is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | P1 | RW | 1 | GPIO Port B Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port B is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port B is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | GPIO Port A Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCGPIO**, **SCGCGPIO** or **DCGCGPIO** register is clear.<br>Value Description<br>0 GPIO Port A is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 GPIO Port A is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 157: Micro Direct Memory Access Power Control (PCDMA), offset 0x90C

**Important:** The µDMA module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCDMA** register controls the power applied to the DMA module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCDMA**, **SCGCDMA** and **DCGCDMA** registers. If the Rn, Sn, or Dn bit of the respective **RCGCDMA**, **SCGCDMA** and **DCGCDMA** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCDMA** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCDMA**, **SCGCDMA** and **DCGCDMA** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCDMA** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-19. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Micro Direct Memory Access Power Control (PCDMA)

Base 0x400F.E000
Offset 0x90C
Type RW, reset 0x0000.0001

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RW | 1 | µDMA Module Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCDMA**, **SCGCDMA** or **DCGCDMA** register is clear.<br>Value Description<br>0 The µDMA module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The µDMA module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 158: External Peripheral Interface Power Control (PCEPI), offset 0x910

**Important:** The EPI module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCEPI** register controls the power applied to the EPI module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCEPI**, **SCGCEPI** and **DCGCEPI** registers. If the Rn, Sn, or Dn bit of the respective **RCGCEPI**, **SCGCEPI** and **DCGCEPI** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCEPI** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCEPI**, **SCGCEPI** and **DCGCEPI** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCEPI** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-20. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

External Peripheral Interface Power Control (PCEPI)

Base 0x400F.E000
Offset 0x910
Type RW, reset 0x0000.0001

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RW | 1 | EPI Module Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCEPI**, **SCGCEPI** or **DCGCEPI** register is clear.<br>Value Description<br>0 The EPI module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The EPI module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 159: Hibernation Power Control (PCHIB), offset 0x914

**Important:** The Hibernation module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCHIB** register controls the power applied to the HIB module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCHIB**, **SCGCHIB** and **DCGCHIB** registers. If the Rn, Sn, or Dn bit of the respective **RCGCHIB**, **SCGCHIB** and **DCGCHIB** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCHIB** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCHIB**, **SCGCHIB** and **DCGCHIB** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCHIB** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-21. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Hibernation Power Control (PCHIB)

Base 0x400F.E000
Offset 0x914
Type RW, reset 0x0000.0001

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | P0 |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | Hibernation Module Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCHIB**, **SCGCHIB** or **DCGCHIB** register is clear.<br>Value Description<br>0 The HIB module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The HIB module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 160: Universal Asynchronous Receiver/Transmitter Power Control (PCUART), offset 0x918

**Important:** The UART module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCUART** register controls the power applied to the UART module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCUART**, **SCGCUART** and **DCGCUART** registers. If the Rn, Sn, or Dn bit of the respective **RCGCUART**, **SCGCUART** and **DCGCUART** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCUART** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCUART**, **SCGCUART** and **DCGCUART** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCUART** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-22. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Universal Asynchronous Receiver/Transmitter Power Control (PCUART)

Base 0x400F.E000
Offset 0x918
Type RW, reset 0x0000.00FF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | P7 | RW | 1 | UART Module 7 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 7 is not powered and does not receive a clock. In this case, the module's state is not retained. This configuration provides the lowest power consumption state.<br>1 The UART module 7 is powered, but does not receive a clock. In this case, the module is inactive. |
| 6 | P6 | RW | 1 | UART Module 6 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 6 is not powered and does not receive a clock. In this case, the module's state is not retained. This configuration provides the lowest power consumption state.<br>1 The UART module 6 is powered, but does not receive a clock. In this case, the module is inactive. |
| 5 | P5 | RW | 1 | UART Module 5 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 5 is not powered and does not receive a clock. In this case, the module's state is not retained. This configuration provides the lowest power consumption state.<br>1 The UART module 5 is powered, but does not receive a clock. In this case, the module is inactive. |
| 4 | P4 | RW | 1 | UART Module 4 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 4 is not powered and does not receive a clock. In this case, the module's state is not retained. This configuration provides the lowest power consumption state.<br>1 The UART module 4 is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | P3 | RW | 1 | UART Module 3 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 3 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The UART module 3 is powered, but does not receive a clock. In this case, the module is inactive. |
| 2 | P2 | RW | 1 | UART Module 2 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 2 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The UART module 2 is powered, but does not receive a clock. In this case, the module is inactive. |
| 1 | P1 | RW | 1 | UART Module 1 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 1 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The UART module 1 is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | UART Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUART**, **SCGCUART** or **DCGCUART** register is clear.<br>Value Description<br>0 The UART module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The UART module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 161: Synchronous Serial Interface Power Control (PCSSI), offset 0x91C

**Important:** The SSI module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCSSI** register controls the power applied to the SSI module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCSSI**, **SCGCSSI** and **DCGCSSI** registers. If the Rn, Sn, or Dn bit of the respective **RCGCSSI**, **SCGCSSI** and **DCGCSSI** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCSSI** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCSSI**, **SCGCSSI** and **DCGCSSI** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCSSI** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-23. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Synchronous Serial Interface Power Control (PCSSI)

Base 0x400F.E000
Offset 0x91C
Type RW, reset 0x0000.000F

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | P3 | P2 | P1 | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | P3 | RW | 1 | SSI Module 3 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCSSI**, **SCGCSSI** or **DCGCSSI** register is clear.<br>Value Description<br>0 The SSI module 3 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The SSI module 3 is powered, but does not receive a clock. In this case, the module is inactive. |
| 2 | P2 | RW | 1 | SSI Module 2 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCSSI**, **SCGCSSI** or **DCGCSSI** register is clear.<br>Value Description<br>0 The SSI module 2 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The SSI module 2 is powered, but does not receive a clock. In this case, the module is inactive. |
| 1 | P1 | RW | 1 | SSI Module 1 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCSSI**, **SCGCSSI** or **DCGCSSI** register is clear.<br>Value Description<br>0 The SSI module 1 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The SSI module 1 is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | SSI Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCSSI**, **SCGCSSI** or **DCGCSSI** register is clear.<br>Value Description<br>0 The SSI module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The SSI module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 162: Inter-Integrated Circuit Power Control (PCI2C), offset 0x920

**Important:** The I2C module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCI2C** register controls the power applied to the I2C module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCI2C**, **SCGCI2C** and **DCGCI2C** registers. If the Rn, Sn, or Dn bit of the respective **RCGCI2C**, **SCGCI2C** and **DCGCI2C** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCI2C** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCI2C**, **SCGCI2C** and **DCGCI2C** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCI2C** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-24. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Inter-Integrated Circuit Power Control (PCI2C)

Base 0x400F.E000
Offset 0x920
Type RW, reset 0x0000.03FF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | P9 | P8 | P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |
| Type | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | P9 | RW | 1 | I²C Module 9 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The I²C module 9 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The I²C module 9 is powered, but does not receive a clock. In this case, the module is inactive. |
| 8 | P8 | RW | 1 | I²C Module 8 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The I²C module 8 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The I²C module 8 is powered, but does not receive a clock. In this case, the module is inactive. |
| 7 | P7 | RW | 1 | I²C Module 7 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The I²C module 7 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The I²C module 7 is powered, but does not receive a clock. In this case, the module is inactive. |
| 6 | P6 | RW | 1 | I²C Module 6 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The I²C module 6 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The I²C module 6 is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | P5 | RW | 1 | $I^2C$ Module 5 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The $I^2C$ module 5 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The $I^2C$ module 5 is powered, but does not receive a clock. In this case, the module is inactive. |
| 4 | P4 | RW | 1 | $I^2C$ Module 4 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The $I^2C$ module 4 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The $I^2C$ module 4 is powered, but does not receive a clock. In this case, the module is inactive. |
| 3 | P3 | RW | 1 | $I^2C$ Module 3 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The $I^2C$ module 3 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The $I^2C$ module 3 is powered, but does not receive a clock. In this case, the module is inactive. |
| 2 | P2 | RW | 1 | $I^2C$ Module 2 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The $I^2C$ module 2 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The $I^2C$ module 2 is powered, but does not receive a clock. In this case, the module is inactive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | P1 | RW | 1 | $I^2C$ Module 1 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The $I^2C$ module 1 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The $I^2C$ module 1 is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | $I^2C$ Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCI2C**, **SCGCI2C** or **DCGCI2C** register is clear.<br>Value Description<br>0 The $I^2C$ module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The $I^2C$ module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 163: Universal Serial Bus Power Control (PCUSB), offset 0x928

The **PCUSB** register controls the power applied to the USB module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCUSB**, **SCGCUSB** and **DCGCUSB** registers. If the Rn, Sn, or Dn bit of the respective **RCGCUSB**, **SCGCUSB** and **DCGCUSB** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCUSB** register is.

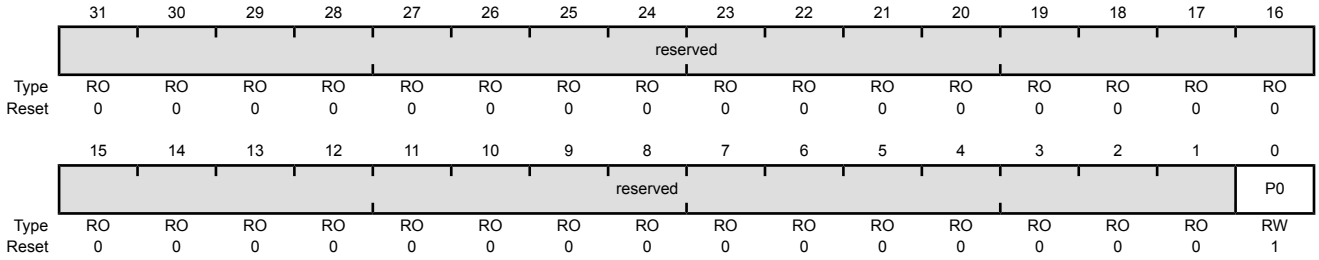
However, if the Rn, Sn, or Dn bit of the respective **RCGCUSB**, **SCGCUSB** and **DCGCUSB** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCUSB** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-25. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Universal Serial Bus Power Control (PCUSB)

Base 0x400F.E000
Offset 0x928
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | USB Module Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCUSB**, **SCGCUSB** or **DCGCUSB** register is clear.<br>Value Description<br>0 The USB module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The USB module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 164: Ethernet PHY Power Control (PCEPHY), offset 0x930

The **PCEPHY** register controls the power applied to the EEPROM module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCEPHY**, **SCGCEPHY** and **DCGCEPHY** registers. If the Rn, Sn, or Dn bit of the respective **RCGCEPHY**, **SCGCEPHY** and **DCGCEPHY** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCEPHY** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCEPHY**, **SCGCEPHY** and **DCGCEPHY** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCEPHY** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-26. Module Power Control**

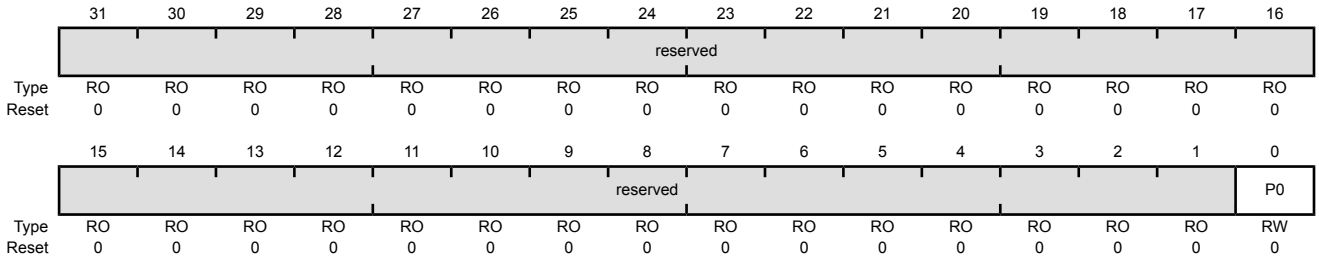
| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

**Note:** The Ethernet PHY module is not powered up at reset to prevent an automatic negotiation on power-up. To properly initialize the PHY, first inhibit the PHY from running on power up by setting the PHYHOLD bit in the **Ethernet Peripheral Configuration (EMACPC)** register and then set the P0 bit in the **PCEPHY** register. Once it is determined that the PHY is ready (by polling the R0 bit in the **Peripheral Ready (PREPHY)** register), the PHY can be programmed with its appropriate values.

**Note:** If the MOSC is chosen as the clock to the Ethernet PHY then software has to enable the MOSC before enabling the Ethernet PHY by setting the P0 bit in the **PCEPHY**.

Ethernet PHY Power Control (PCEPHY)

Base 0x400F.E000
Offset 0x930
Type RW, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RW | 0 | Ethernet PHY Module Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCEPHY**, **SCGCEPHY** or **DCGCEPHY** register is clear.<br>Value Description<br>0 The EPHY module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The EPHY module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 165: Controller Area Network Power Control (PCCAN), offset 0x934

The **PCCAN** register controls the power applied to the CAN module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCCAN**, **SCGCCAN** and **DCGCCAN** registers. If the Rn, Sn, or Dn bit of the respective **RCGCCAN**, **SCGCCAN** and **DCGCCAN** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCCAN** register is.

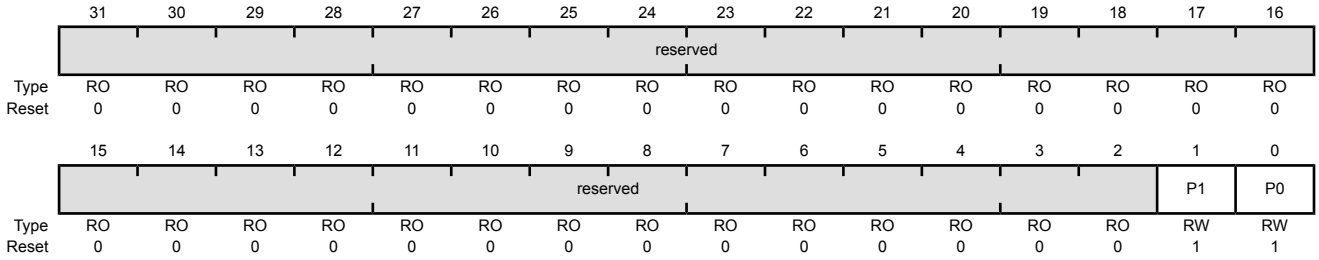
However, if the Rn, Sn, or Dn bit of the respective **RCGCCAN**, **SCGCCAN** and **DCGCCAN** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCCAN** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-27. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Controller Area Network Power Control (PCCAN)

Base 0x400F.E000
Offset 0x934
Type RW, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | P1 | RW | 1 | CAN Module 1 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCCAN**, **SCGCCAN** or **DCGCCAN** register is clear.<br>Value Description<br>0 The CAN module 1 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The CAN module 1 is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | CAN Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCCAN**, **SCGCCAN** or **DCGCCAN** register is clear.<br>Value Description<br>0 The CAN module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The CAN module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 166: Analog-to-Digital Converter Power Control (PCADC), offset 0x938

**Important:** The ADC module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCADC** register controls the power applied to the ADC module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCADC**, **SCGCADC** and **DCGCADC** registers. If the Rn, Sn, or Dn bit of the respective **RCGCADC**, **SCGCADC** and **DCGCADC** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCADC** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCADC**, **SCGCADC** and **DCGCADC** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCADC** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-28. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Analog-to-Digital Converter Power Control (PCADC)

Base 0x400F.E000
Offset 0x938
Type RW, reset 0x0000.0003

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | P1 | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | P1 | RW | 1 | ADC Module 1 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCADC**, **SCGCADC** or **DCGCADC** register is clear.<br>Value Description<br>0 The ADC module 1 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The ADC module 1 is powered, but does not receive a clock. In this case, the module is inactive. |
| 0 | P0 | RW | 1 | ADC Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCADC**, **SCGCADC** or **DCGCADC** register is clear.<br>Value Description<br>0 The ADC module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The ADC module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 167: Analog Comparator Power Control (PCACMP), offset 0x93C

**Important:** The ACMP module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCACMP** register controls the power applied to the ACMP module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCACMP**, **SCGCACMP** and **DCGCACMP** registers. If the Rn, Sn, or Dn bit of the respective **RCGCACMP**, **SCGCACMP** and **DCGCACMP** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCACMP** register is.

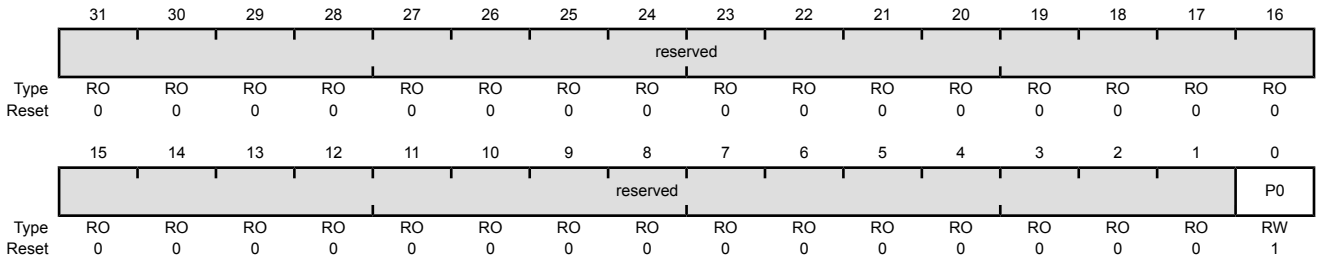
However, if the Rn, Sn, or Dn bit of the respective **RCGCACMP**, **SCGCACMP** and **DCGCACMP** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCACMP** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-29. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Analog Comparator Power Control (PCACMP)

Base 0x400F.E000
Offset 0x93C
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | Analog Comparator Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCACMP**, **SCGCACMP** or **DCGCACMP** register is clear.<br>Value Description<br>0 The Analog Comparator module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The Analog Comparator module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 168: Pulse Width Modulator Power Control (PCPWM), offset 0x940

**Important:** The PWM module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCPWM** register controls the power applied to the PWM module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCPWM**, **SCGCPWM** and **DCGCPWM** registers. If the Rn, Sn, or Dn bit of the respective **RCGCPWM**, **SCGCPWM** and **DCGCPWM** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCPWM** register is.

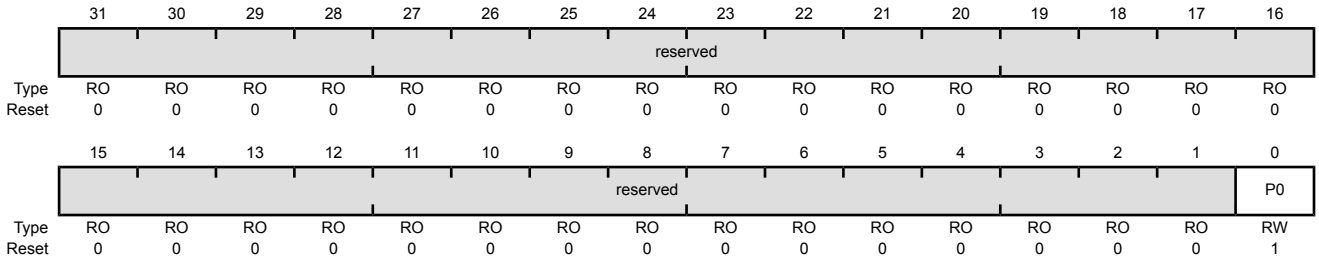
However, if the Rn, Sn, or Dn bit of the respective **RCGCPWM**, **SCGCPWM** and **DCGCPWM** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCPWM** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-30. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Pulse Width Modulator Power Control (PCPWM)

Base 0x400F.E000
Offset 0x940
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | PWM Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCPWM**, **SCGCPWM** or **DCGCPWM** register is clear.<br>Value Description<br>0 The PWM module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The PWM module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 169: Quadrature Encoder Interface Power Control (PCQEI), offset 0x944

**Important:** The QEI module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCQEI** register controls the power applied to the QEI module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCQEI**, **SCGCQEI** and **DCGCQEI** registers. If the Rn, Sn, or Dn bit of the respective **RCGCQEI**, **SCGCQEI** and **DCGCQEI** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCQEI** register is.

However, if the Rn, Sn, or Dn bit of the respective **RCGCQEI**, **SCGCQEI** and **DCGCQEI** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCQEI** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-31. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Quadrature Encoder Interface Power Control (PCQEI)

Base 0x400F.E000
Offset 0x944
Type RW, reset 0x0000.0001

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | P0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | P0 | RW | 1 | QEI Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCQEI**, **SCGCQEI** or **DCGCQEI** register is clear.<br>Value Description<br>0 QEI module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 QEI module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 170: EEPROM Power Control (PCEEPROM), offset 0x958

**Important:** The EEPROM module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCEEPROM** register controls the power applied to the EEPROM module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCEEPROM**, **SCGCEEPROM** and **DCGCEEPROM** registers. If the Rn, Sn, or Dn bit of the respective **RCGCEEPROM**, **SCGCEEPROM** and **DCGCEEPROM** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCEEPROM** register is.

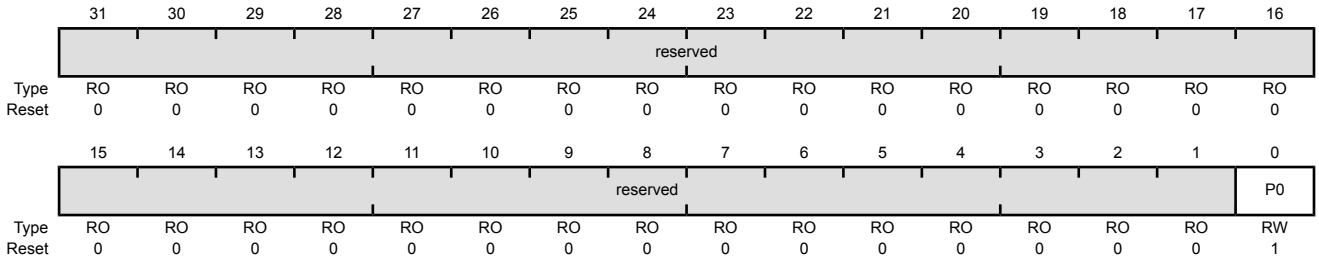
However, if the Rn, Sn, or Dn bit of the respective **RCGCEEPROM**, **SCGCEEPROM** and **DCGCEEPROM** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCEEPROM** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-32. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

EEPROM Power Control (PCEEPROM)

Base 0x400F.E000
Offset 0x958
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | EEPROM Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCEEPROM**, **SCGCEEPROM** or **DCGCEEPROM** register is clear.<br>Value Description<br>0 The EEPROM module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The EEPROM module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 171: CRC and Cryptographic Modules Power Control (PCCCM), offset 0x974

The **PCCCM** register controls the power applied to the CRC and AES, DES, and SHA/MD5 modules. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCCCM**, **SCGCCCM** and **DCGCCCM** registers. If the Rn, Sn, or Dn bit of the respective **RCGCCCM**, **SCGCCCM** and **DCGCCCM** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCCCM** register is.

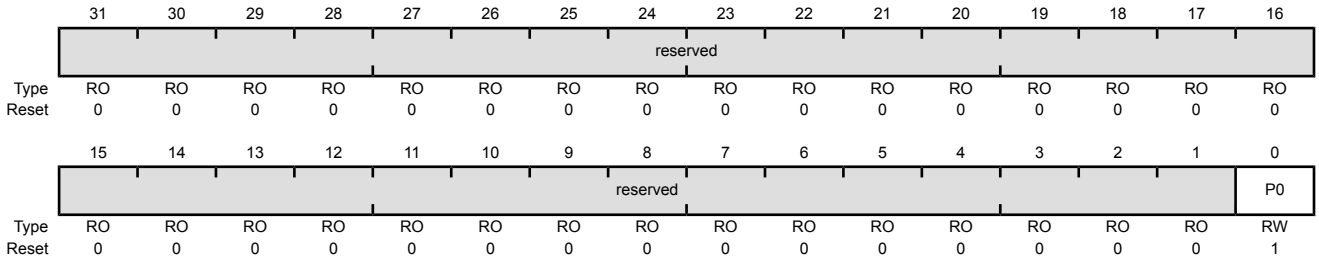
However, if the Rn, Sn, or Dn bit of the respective **RCGCCCM**, **SCGCCCM** and **DCGCCCM** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCCCM** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-33. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

CRC and Cryptographic Modules Power Control (PCCCM)

Base 0x400F.E000
Offset 0x974
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | CRC and Cryptographic Modules Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCCCM**, **SCGCCCM** or **DCGCCCM** register is clear.<br>Value Description<br>0 The CRC, AES, DES, and SHA/MD5 modules are not powered and do not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The CRC, AES, DES, and SHA/MD5 modules are powered, but do not receive a clock. In this case, the modules are inactive. |

## Register 172: LCD Controller Power Control (PCLCD), offset 0x990

The **PCLCD** register controls the power applied to the LCD module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCLCD**, **SCGCLCD** and **DCGCLCD** registers. If the Rn, Sn, or Dn bit of the respective **RCGCLCD**, **SCGCLCD** and **DCGCLCD** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCLCD** register is.

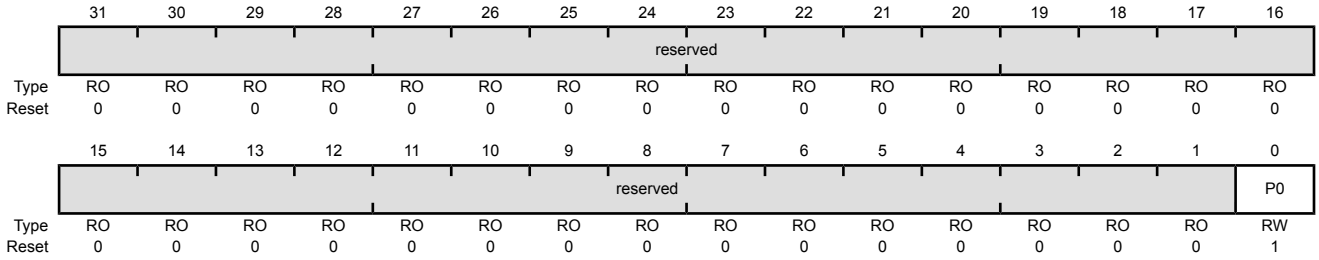
However, if the Rn, Sn, or Dn bit of the respective **RCGCLCD**, **SCGCLCD** and **DCGCLCD** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCLCD** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-34. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

LCD Controller Power Control (PCLCD)

Base 0x400F.E000
Offset 0x990
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | LCD Controller Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCLCD**, **SCGCLCD** or **DCGCLCD** register is clear.<br>Value Description<br>0 LCD module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 LCD module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 173: 1-Wire Power Control (PCOWIRE), offset 0x998

**Important:** The 1-Wire module does not currently provide the ability to respond to the power down request. Setting a bit in this register has no effect on power consumption. This register is defined for future software compatibility.

The **PCOWIRE** register controls the power applied to the 1-Wire module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCOWIRE**, **SCGCOWIRE** and **DCGCOWIRE** registers. If the Rn, Sn, or Dn bit of the respective **RCGCOWIRE**, **SCGCOWIRE** and **DCGCOWIRE** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCOWIRE** register is.

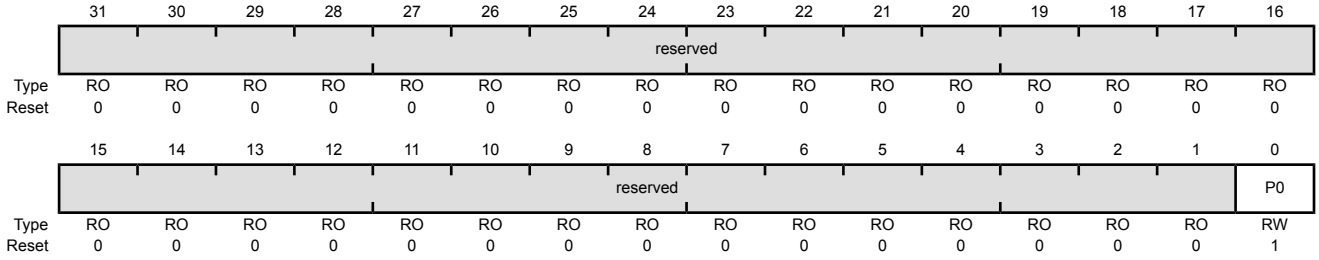
However, if the Rn, Sn, or Dn bit of the respective **RCGCOWIRE**, **SCGCOWIRE** and **DCGCOWIRE** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCOWIRE** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-35. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

1-Wire Power Control (PCOWIRE)

Base 0x400F.E000
Offset 0x998
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | 1-Wire Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCOWIRE**, **SCGCOWIRE** or **DCGCOWIRE** register is clear.<br>Value Description<br>0 The 1-Wire module is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 The 1-Wire module is powered, but does not receive a clock. In this case, the module is inactive. |

## Register 174: Ethernet MAC Power Control (PCEMAC), offset 0x99C

The **PCEMAC** register controls the power applied to the EMAC module. The function of this bit depends on the current state of the device (Run, Sleep or Deep-Sleep mode) and value of the corresponding bits in the **RCGCEMAC**, **SCGCEMAC** and **DCGCEMAC** registers. If the Rn, Sn, or Dn bit of the respective **RCGCEMAC**, **SCGCEMAC** and **DCGCEMAC** registers is 1 and the device is in that mode, the module is powered and receives a clock irrespective of what the corresponding Pn bit in the **PCEMAC** register is.

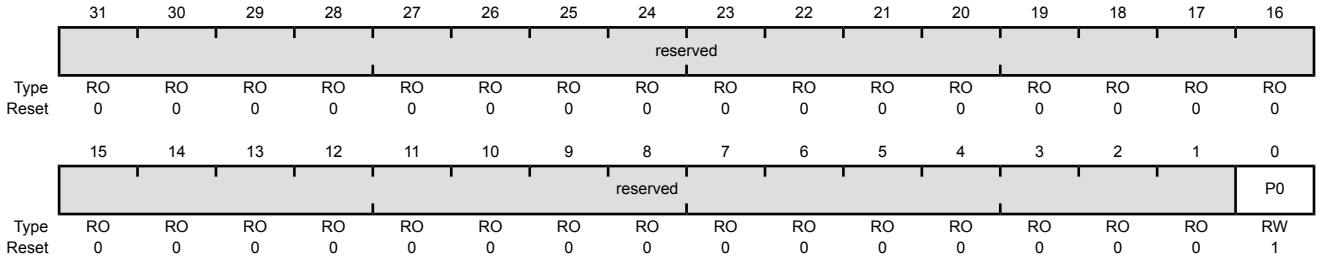
However, if the Rn, Sn, or Dn bit of the respective **RCGCEMAC**, **SCGCEMAC** and **DCGCEMAC** registers is 0 and the device is in that mode, then the module behaves differently depending on the value of the corresponding Pn bit in the **PCEMAC** register. In this case, when the Pn bit is clear the module is not powered and does not receive a clock. If the Pn bit is set, the module is powered but does not receive a clock. The table below details the differences.

**Table 5-36. Module Power Control**

| Rn, Sn or Dn Value in Respective RCGCx, SCGCx, or DCGCx Register | Pn | Description |
|---|---|---|
| 0 | 0 | Module is not powered and does not receive a clock. In this case, the peripheral's state is not retained.<br>This is the lowest power consumption state of any peripheral since it consumes no dynamic nor leakage current. Hardware should perform a peripheral reset if the active mode changes and the **RCGCx**, **SCGCx**, or **DCGCx** register is a 1 or the P0 bit is changed to a 1.<br>Software must re-initialize the peripheral when re-enabled due to the loss of state. |
| 0 | 1 | Module is powered, but does not receive a clock.<br>In this case, the peripheral is inactive. This is the second-lowest power consumption of any peripheral since it consumes only leakage current. |
| 1 | X | Module is powered and receives a clock. |

Ethernet MAC Power Control (PCEMAC)

Base 0x400F.E000
Offset 0x99C
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | P0 | RW | 1 | Ethernet MAC Module 0 Power Control<br>The Pn bit encodings are not applicable if the corresponding bit in the **RCGCEMAC**, **SCGCEMAC** or **DCGCEMAC** register is clear.<br>Value Description<br>0 Ethernet MAC Module 0 is not powered and does not receive a clock. In this case, the module's state is not retained.<br>This configuration provides the lowest power consumption state.<br>1 Ethernet MAC module 0 is powered, but does not receive a clock. In this case, the module is inactive. |

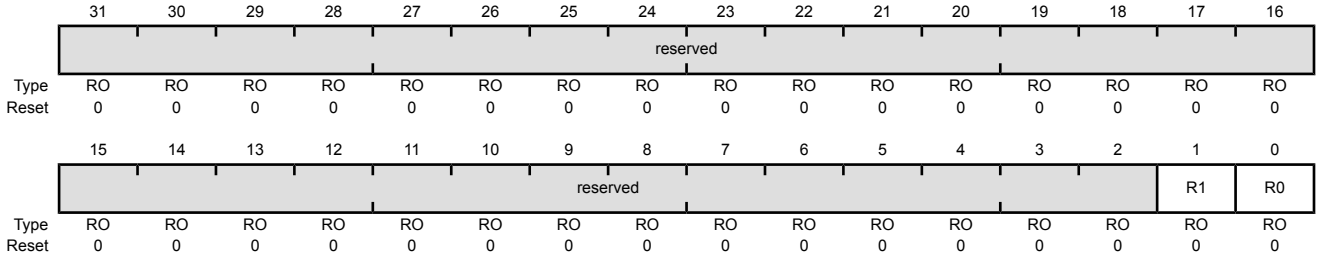
## Register 175: Watchdog Timer Peripheral Ready (PRWD), offset 0xA00

The **PRWD** register indicates whether the watchdog modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCWD** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCWD** bit is changed. A reset change is initiated if the corresponding **SRWD** bit is changed from 0 to 1.

The **PRWD** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Watchdog Timer Peripheral Ready (PRWD)

Base 0x400F.E000
Offset 0xA00
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RO | 0 | Watchdog Timer 1 Peripheral Ready<br>Value Description<br>0 Watchdog module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 Watchdog module 1 is ready for access. |
| 0 | R0 | RO | 0 | Watchdog Timer 0 Peripheral Ready<br>Value Description<br>0 Watchdog module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 Watchdog module 0 is ready for access. |

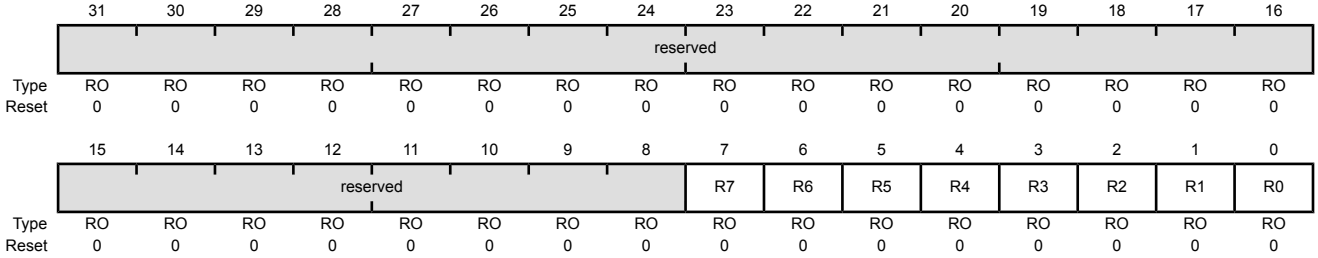
## Register 176: 16/32-Bit General-Purpose Timer Peripheral Ready (PRTIMER), offset 0xA04

The **PRGPT32** register indicates whether the timer modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCGPT32** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCGPT32** bit is changed. A reset change is initiated if the corresponding **SRGPT32** bit is changed from 0 to 1.

The **PRGPT32** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

16/32-Bit General-Purpose Timer Peripheral Ready (PRTIMER)

Base 0x400F.E000
Offset 0xA04
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | R7 | RO | 0 | 16/32-Bit General-Purpose Timer 7 Peripheral Ready<br>Value Description<br>0 16/32-bit timer module 7 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 7 is ready for access. |
| 6 | R6 | RO | 0 | 16/32-Bit General-Purpose Timer 6 Peripheral Ready<br>Value Description<br>0 16/32-bit timer module 6 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 6 is ready for access. |
| 5 | R5 | RO | 0 | 16/32-Bit General-Purpose Timer 5 Peripheral Ready<br>Value Description<br>0 16/32-bit timer module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 5 is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | R4 | RO | 0 | 16/32-Bit General-Purpose Timer 4 Peripheral Ready<br><br>Value Description<br>0 16/32-bit timer module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 4 is ready for access. |
| 3 | R3 | RO | 0 | 16/32-Bit General-Purpose Timer 3 Peripheral Ready<br><br>Value Description<br>0 16/32-bit timer module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 3 is ready for access. |
| 2 | R2 | RO | 0 | 16/32-Bit General-Purpose Timer 2 Peripheral Ready<br><br>Value Description<br>0 16/32-bit timer module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 2 is ready for access. |
| 1 | R1 | RO | 0 | 16/32-Bit General-Purpose Timer 1 Peripheral Ready<br><br>Value Description<br>0 16/32-bit timer module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 1 is ready for access. |
| 0 | R0 | RO | 0 | 16/32-Bit General-Purpose Timer 0 Peripheral Ready<br><br>Value Description<br>0 16/32-bit timer module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 16/32-bit timer module 0 is ready for access. |

## Register 177: General-Purpose Input/Output Peripheral Ready (PRGPIO), offset 0xA08

The **PRGPIO** register indicates whether the GPIO modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCGPIO** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCGPIO** bit is changed. A reset change is initiated if the corresponding **SRGPIO** bit is changed from 0 to 1.

The **PRGPIO** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

General-Purpose Input/Output Peripheral Ready (PRGPIO)

Base 0x400F.E000
Offset 0xA08
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | R17 | R16 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | R17 | RO | 0 | GPIO Port T Peripheral Ready<br>Value Description<br>0 GPIO Port T is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port T is ready for access. |
| 16 | R16 | RO | 0 | GPIO Port S Peripheral Ready<br>Value Description<br>0 GPIO Port S is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port S is ready for access. |
| 15 | R15 | RO | 0 | GPIO Port R Peripheral Ready<br>Value Description<br>0 GPIO Port R is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port R is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 14 | R14 | RO | 0 | GPIO Port Q Peripheral Ready<br><br>Value Description<br>0 GPIO Port Q is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port Q is ready for access. |
| 13 | R13 | RO | 0 | GPIO Port P Peripheral Ready<br><br>Value Description<br>0 GPIO Port P is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port P is ready for access. |
| 12 | R12 | RO | 0 | GPIO Port N Peripheral Ready<br><br>Value Description<br>0 GPIO Port N is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port N is ready for access. |
| 11 | R11 | RO | 0 | GPIO Port M Peripheral Ready<br><br>Value Description<br>0 GPIO Port M is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port M is ready for access. |
| 10 | R10 | RO | 0 | GPIO Port L Peripheral Ready<br><br>Value Description<br>0 GPIO Port L is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port L is ready for access. |
| 9 | R9 | RO | 0 | GPIO Port K Peripheral Ready<br><br>Value Description<br>0 GPIO Port K is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port K is ready for access. |
| 8 | R8 | RO | 0 | GPIO Port J Peripheral Ready<br><br>Value Description<br>0 GPIO Port J is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port J is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | R7 | RO | 0 | GPIO Port H Peripheral Ready<br><br>Value Description<br>0 GPIO Port H is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port H is ready for access. |
| 6 | R6 | RO | 0 | GPIO Port G Peripheral Ready<br><br>Value Description<br>0 GPIO Port G is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port G is ready for access. |
| 5 | R5 | RO | 0 | GPIO Port F Peripheral Ready<br><br>Value Description<br>0 GPIO Port F is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port F is ready for access. |
| 4 | R4 | RO | 0 | GPIO Port E Peripheral Ready<br><br>Value Description<br>0 GPIO Port E is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port E is ready for access. |
| 3 | R3 | RO | 0 | GPIO Port D Peripheral Ready<br><br>Value Description<br>0 GPIO Port D is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port D is ready for access. |
| 2 | R2 | RO | 0 | GPIO Port C Peripheral Ready<br><br>Value Description<br>0 GPIO Port C is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port C is ready for access. |
| 1 | R1 | RO | 0 | GPIO Port B Peripheral Ready<br><br>Value Description<br>0 GPIO Port B is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 GPIO Port B is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | R0 | RO | 0 | GPIO Port A Peripheral Ready |

| Value | Description |
|---|---|
| 0 | GPIO Port A is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | GPIO Port A is ready for access. |

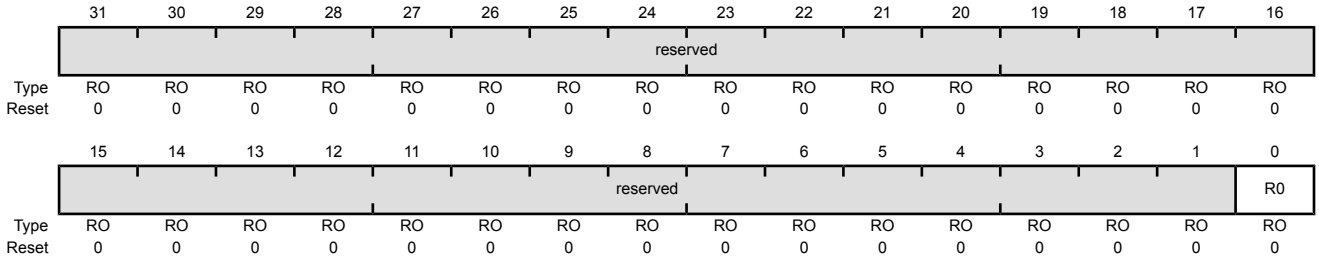
## Register 178: Micro Direct Memory Access Peripheral Ready (PRDMA), offset 0xA0C

The **PRDMA** register indicates whether the µDMA module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCDMA** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCDMA** bit is changed. A reset change is initiated if the corresponding **SRDMA** bit is changed from 0 to 1.

The **PRDMA** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Micro Direct Memory Access Peripheral Ready (PRDMA)

Base 0x400F.E000
Offset 0xA0C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | µDMA Module Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The µDMA module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The µDMA module is ready for access. |

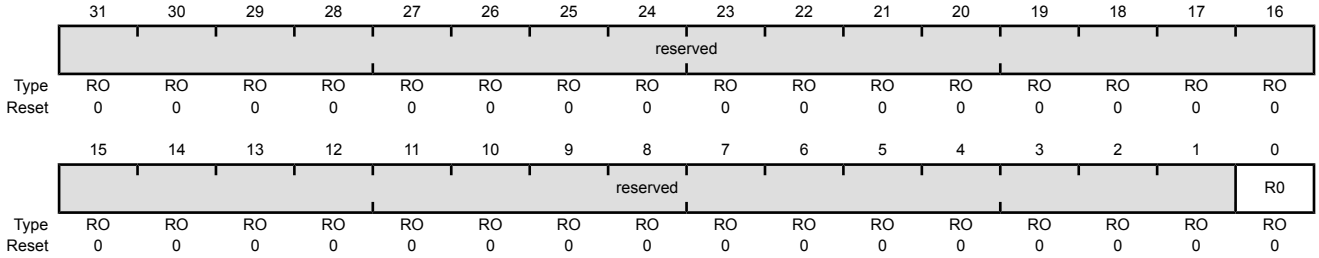
## Register 179: EPI Peripheral Ready (PREPI), offset 0xA10

The **PREPI** register indicates whether the EPI module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCEPI** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCEPI** bit is changed. A reset change is initiated if the corresponding **SREPI** bit is changed from 0 to 1.

The **PREPI** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

EPI Peripheral Ready (PREPI)

Base 0x400F.E000
Offset 0xA10
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | EPI Module Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The EPI module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The EPI module is ready for access. |

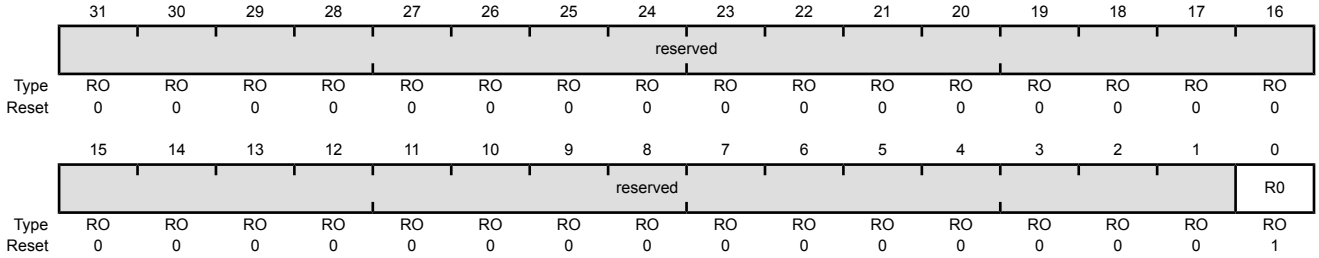
## Register 180: Hibernation Peripheral Ready (PRHIB), offset 0xA14

The **PRHIB** register indicates whether the Hibernation module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCHIB** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCHIB** bit is changed. A reset change is initiated if the corresponding **SRHIB** bit is changed from 0 to 1.

The **PRHIB** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Hibernation Peripheral Ready (PRHIB)

Base 0x400F.E000
Offset 0xA14
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 1 | Hibernation Module Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The Hibernation module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The Hibernation module is ready for access. |

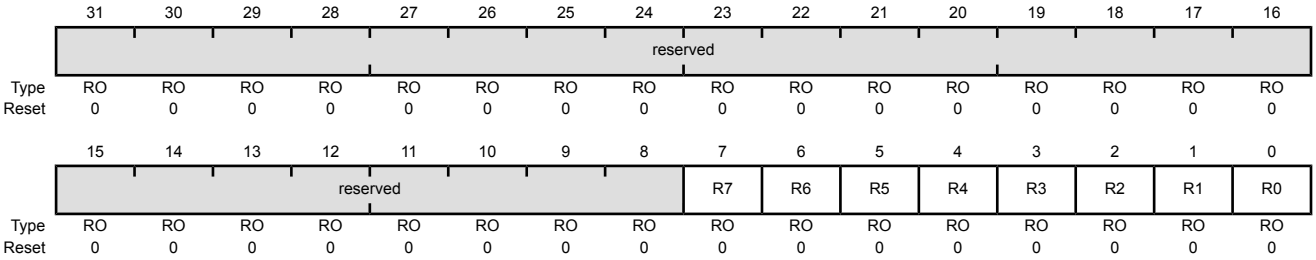
## Register 181: Universal Asynchronous Receiver/Transmitter Peripheral Ready (PRUART), offset 0xA18

The **PRUART** register indicates whether the UART modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCUART** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCUART** bit is changed. A reset change is initiated if the corresponding **SRUART** bit is changed from 0 to 1.

The **PRUART** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Universal Asynchronous Receiver/Transmitter Peripheral Ready (PRUART)

Base 0x400F.E000
Offset 0xA18
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | R7 | RO | 0 | UART Module 7 Peripheral Ready<br>Value Description<br>0 UART module 7 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 7 is ready for access. |
| 6 | R6 | RO | 0 | UART Module 6 Peripheral Ready<br>Value Description<br>0 UART module 6 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 6 is ready for access. |
| 5 | R5 | RO | 0 | UART Module 5 Peripheral Ready<br>Value Description<br>0 UART module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 5 is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | R4 | RO | 0 | UART Module 4 Peripheral Ready<br><br>Value Description<br>0 UART module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 4 is ready for access. |
| 3 | R3 | RO | 0 | UART Module 3 Peripheral Ready<br><br>Value Description<br>0 UART module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 3 is ready for access. |
| 2 | R2 | RO | 0 | UART Module 2 Peripheral Ready<br><br>Value Description<br>0 UART module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 2 is ready for access. |
| 1 | R1 | RO | 0 | UART Module 1 Peripheral Ready<br><br>Value Description<br>0 UART module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 1 is ready for access. |
| 0 | R0 | RO | 0 | UART Module 0 Peripheral Ready<br><br>Value Description<br>0 UART module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 UART module 0 is ready for access. |

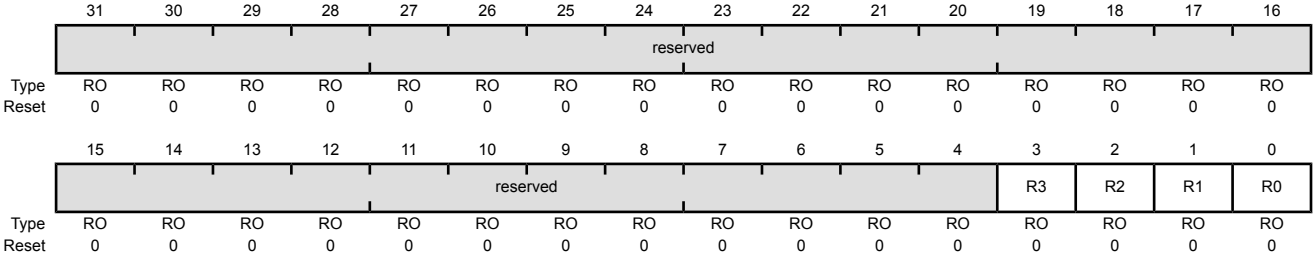
## Register 182: Synchronous Serial Interface Peripheral Ready (PRSSI), offset 0xA1C

The **PRSSI** register indicates whether the SSI modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCSSI** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCSSI** bit is changed. A reset change is initiated if the corresponding **SRSSI** bit is changed from 0 to 1.

The **PRSSI** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Synchronous Serial Interface Peripheral Ready (PRSSI)

Base 0x400F.E000
Offset 0xA1C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | R3 | RO | 0 | SSI Module 3 Peripheral Ready<br>Value Description<br>0 SSI module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 SSI module 3 is ready for access. |
| 2 | R2 | RO | 0 | SSI Module 2 Peripheral Ready<br>Value Description<br>0 SSI module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 SSI module 2 is ready for access. |
| 1 | R1 | RO | 0 | SSI Module 1 Peripheral Ready<br>Value Description<br>0 SSI module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 SSI module 1 is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | R0 | RO | 0 | SSI Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | SSI module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | SSI module 0 is ready for access. |

## Register 183: Inter-Integrated Circuit Peripheral Ready (PRI2C), offset 0xA20

The **PRI2C** register indicates whether the $I^2C$ modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCI2C** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCI2C** bit is changed. A reset change is initiated if the corresponding **SRI2C** bit is changed from 0 to 1.

The **PRI2C** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Inter-Integrated Circuit Peripheral Ready (PRI2C)

Base 0x400F.E000
Offset 0xA20
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | R9 | R8 | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0 | 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. |
| 9 | R9 | RO | 0 | $I^2C$ Module 9 Peripheral Ready<br>Value Description<br>0 $I^2C$ module 9 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 $I^2C$ module 9 is ready for access. |
| 8 | R8 | RO | 0 | $I^2C$ Module 8 Peripheral Ready<br>Value Description<br>0 $I^2C$ module 8 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 $I^2C$ module 8 is ready for access. |
| 7 | R7 | RO | 0 | $I^2C$ Module 7 Peripheral Ready<br>Value Description<br>0 $I^2C$ module 7 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 $I^2C$ module 7 is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | R6 | RO | 0 | I²C Module 6 Peripheral Ready<br>Value Description<br>0 I²C module 6 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 I²C module 6 is ready for access. |
| 5 | R5 | RO | 0 | I²C Module 5 Peripheral Ready<br>Value Description<br>0 I²C module 5 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 I²C module 5 is ready for access. |
| 4 | R4 | RO | 0 | I²C Module 4 Peripheral Ready<br>Value Description<br>0 I²C module 4 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 I²C module 4 is ready for access. |
| 3 | R3 | RO | 0 | I²C Module 3 Peripheral Ready<br>Value Description<br>0 I²C module 3 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 I²C module 3 is ready for access. |
| 2 | R2 | RO | 0 | I²C Module 2 Peripheral Ready<br>Value Description<br>0 I²C module 2 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 I²C module 2 is ready for access. |
| 1 | R1 | RO | 0 | I²C Module 1 Peripheral Ready<br>Value Description<br>0 I²C module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 I²C module 1 is ready for access. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | R0 | RO | 0 | $I^2C$ Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | $I^2C$ module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | $I^2C$ module 0 is ready for access. |

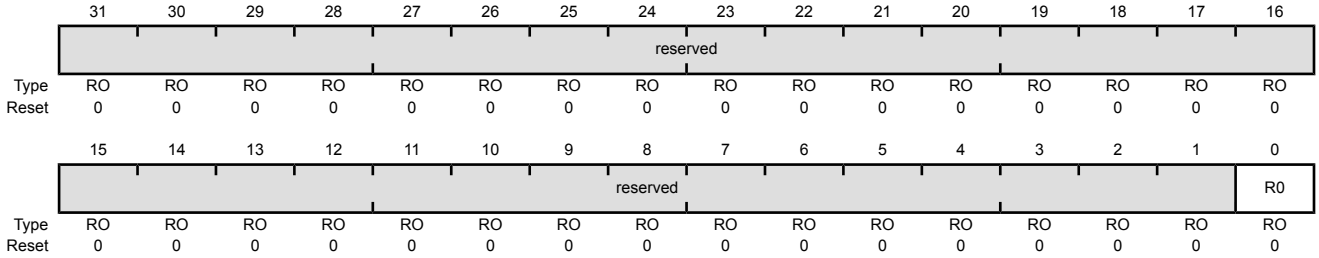
## Register 184: Universal Serial Bus Peripheral Ready (PRUSB), offset 0xA28

The **PRUSB** register indicates whether the USB module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCUSB** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCUSB** bit is changed. A reset change is initiated if the corresponding **SRUSB** bit is changed from 0 to 1.

The **PRUSB** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Universal Serial Bus Peripheral Ready (PRUSB)

Base 0x400F.E000
Offset 0xA28
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | USB Module Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The USB module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The USB module is ready for access. |

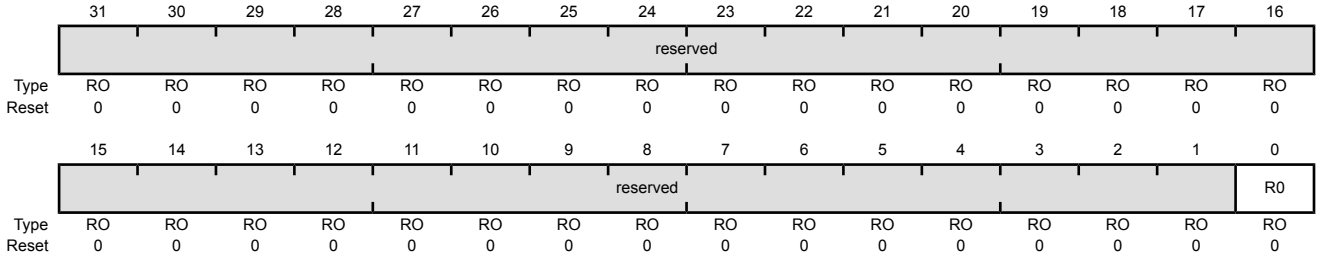
## Register 185: Ethernet PHY Peripheral Ready (PREPHY), offset 0xA30

The **PREPHY** register indicates whether the Ethernet PHY module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCEPHY** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCEPHY** bit is changed. A reset change is initiated if the corresponding **SREPHY** bit is changed from 0 to 1.

The **PREPHY** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Ethernet PHY Peripheral Ready (PREPHY)

Base 0x400F.E000
Offset 0xA30
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | Ethernet PHY Module Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The Ethernet PHY module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The Ethernet PHY module is ready for access. |

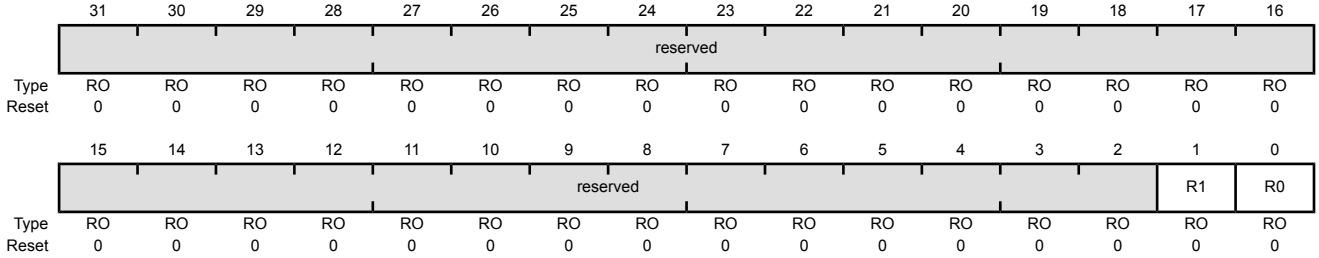
## Register 186: Controller Area Network Peripheral Ready (PRCAN), offset 0xA34

The **PRCAN** register indicates whether the CAN modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCCAN** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCCAN** bit is changed. A reset change is initiated if the corresponding **SRCAN** bit is changed from 0 to 1.

The **PRCAN** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Controller Area Network Peripheral Ready (PRCAN)

Base 0x400F.E000
Offset 0xA34
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RO | 0 | CAN Module 1 Peripheral Ready<br>Value Description<br>0 CAN module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 CAN module 1 is ready for access. |
| 0 | R0 | RO | 0 | CAN Module 0 Peripheral Ready<br>Value Description<br>0 CAN module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 CAN module 0 is ready for access. |

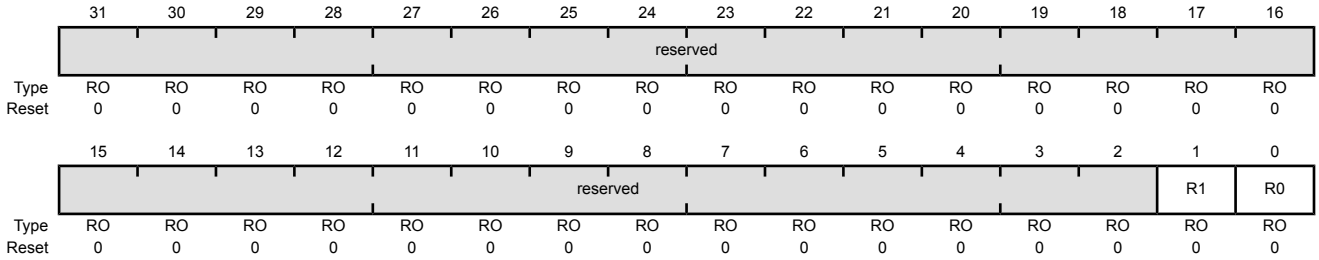
## Register 187: Analog-to-Digital Converter Peripheral Ready (PRADC), offset 0xA38

The **PRADC** register indicates whether the ADC modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCADC** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCADC** bit is changed. A reset change is initiated if the corresponding **SRADC** bit is changed from 0 to 1.

The **PRADC** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Analog-to-Digital Converter Peripheral Ready (PRADC)

Base 0x400F.E000
Offset 0xA38
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | 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. |
| 1 | R1 | RO | 0 | ADC Module 1 Peripheral Ready<br>Value Description<br>0 ADC module 1 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 ADC module 1 is ready for access. |
| 0 | R0 | RO | 0 | ADC Module 0 Peripheral Ready<br>Value Description<br>0 ADC module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 ADC module 0 is ready for access. |

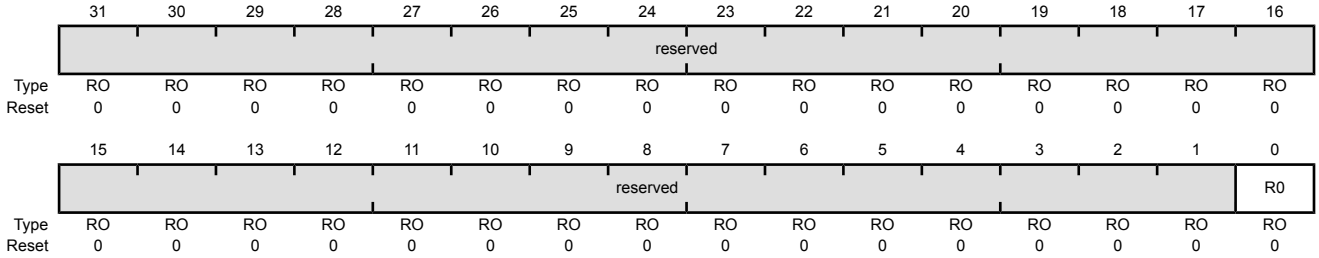
## Register 188: Analog Comparator Peripheral Ready (PRACMP), offset 0xA3C

The **PRACMP** register indicates whether the analog comparator module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCACMP** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCACMP** bit is changed. A reset change is initiated if the corresponding **SRACMP** bit is changed from 0 to 1.

The **PRACMP** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Analog Comparator Peripheral Ready (PRACMP)

Base 0x400F.E000
Offset 0xA3C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | Analog Comparator Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The analog comparator module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The analog comparator module is ready for access. |

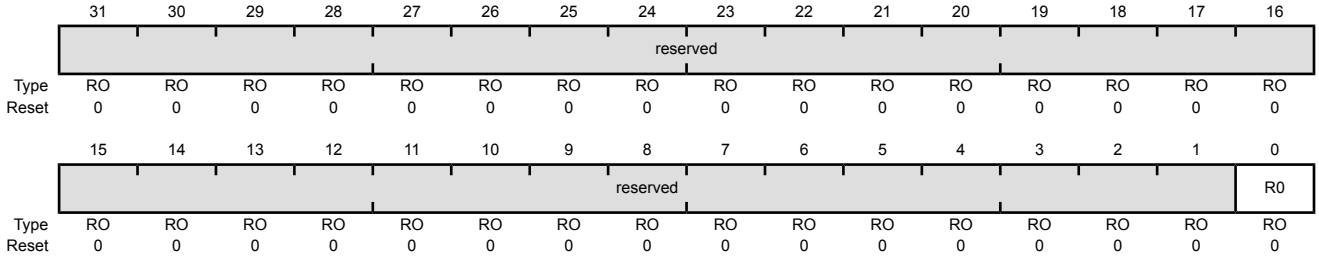
## Register 189: Pulse Width Modulator Peripheral Ready (PRPWM), offset 0xA40

The **PRPWM** register indicates whether the PWM modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCPWM** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCPWM** bit is changed. A reset change is initiated if the corresponding **SRPWM** bit is changed from 0 to 1.

The **PRPWM** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Pulse Width Modulator Peripheral Ready (PRPWM)

Base 0x400F.E000
Offset 0xA40
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | PWM Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | PWM module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | PWM module 0 is ready for access. |

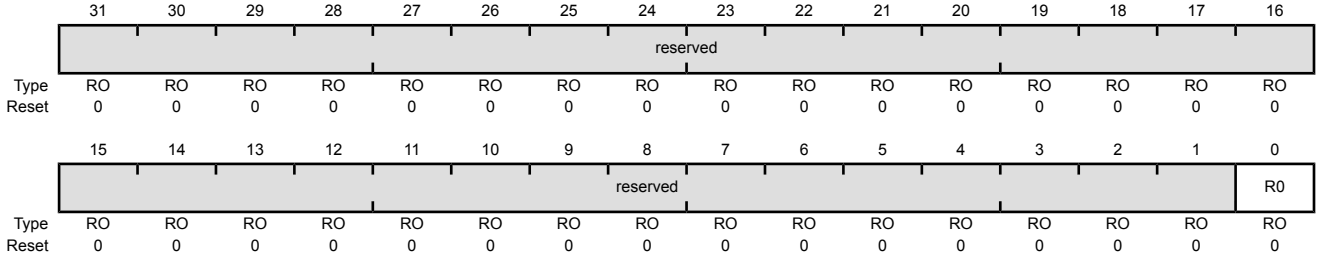
## Register 190: Quadrature Encoder Interface Peripheral Ready (PRQEI), offset 0xA44

The **PRQEI** register indicates whether the QEI modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCQEI** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCQEI** bit is changed. A reset change is initiated if the corresponding **SRQEI** bit is changed from 0 to 1.

The **PRQEI** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Quadrature Encoder Interface Peripheral Ready (PRQEI)

Base 0x400F.E000
Offset 0xA44
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | QEI Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | QEI module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | QEI module 0 is ready for access. |

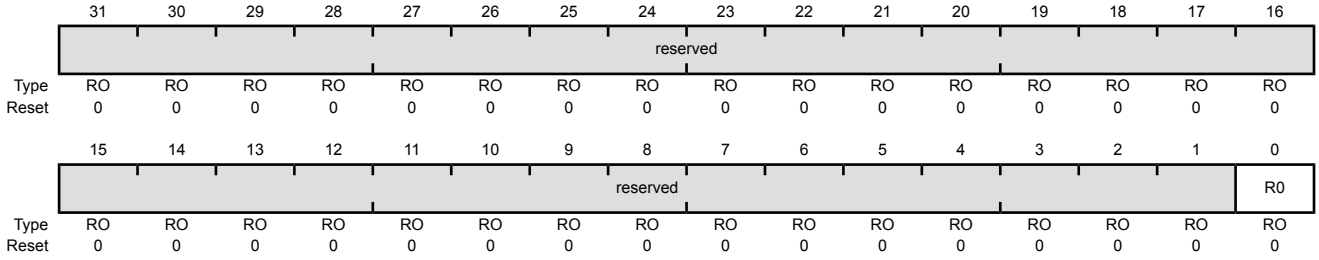
## Register 191: EEPROM Peripheral Ready (PREEPROM), offset 0xA58

The **PREEPROM** register indicates whether the EEPROM module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCEEPROM** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCEEPROM** bit is changed. A reset change is initiated if the corresponding **SREEPROM** bit is changed from 0 to 1.

The **PREEPROM** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

EEPROM Peripheral Ready (PREEPROM)

Base 0x400F.E000
Offset 0xA58
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | EEPROM Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The EEPROM module is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The EEPROM module is ready for access. |

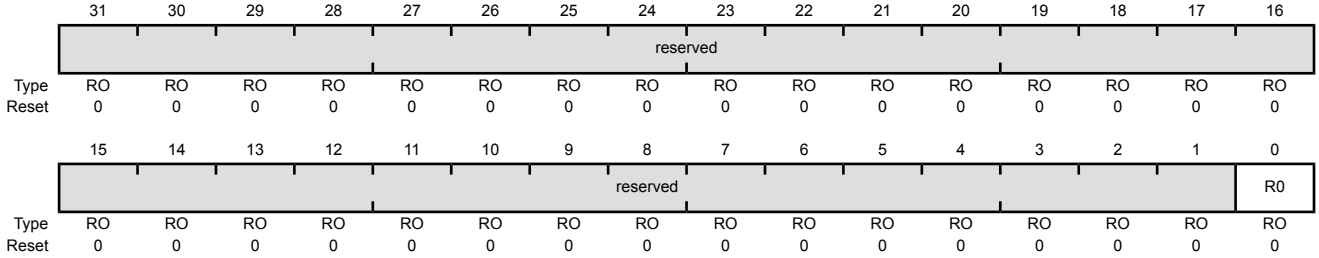
## Register 192: CRC and Cryptographic Modules Peripheral Ready (PRCCM), offset 0xA74

The **PRCCM** register indicates whether the CRC and Cryptographic Modules( AES, DES, and SHA/MD5) are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCCCM** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCCCM** bit is changed. A reset change is initiated if the corresponding **SRCCM** bit is changed from 0 to 1.

The **PRCCM** bit is cleared on any of the above events and is not set again until the modules are completely powered, enabled, and internally reset.

CRC and Cryptographic Modules Peripheral Ready (PRCCM)

Base 0x400F.E000
Offset 0xA74
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | CRC and Cryptographic Modules Peripheral Ready |

| Value | Description |
|---|---|
| 0 | The CRC, AES, DES, and SHA/MD modules are not ready for access. They are unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | The CRC, AES, DES, and SHA/MD modules are ready for access. |

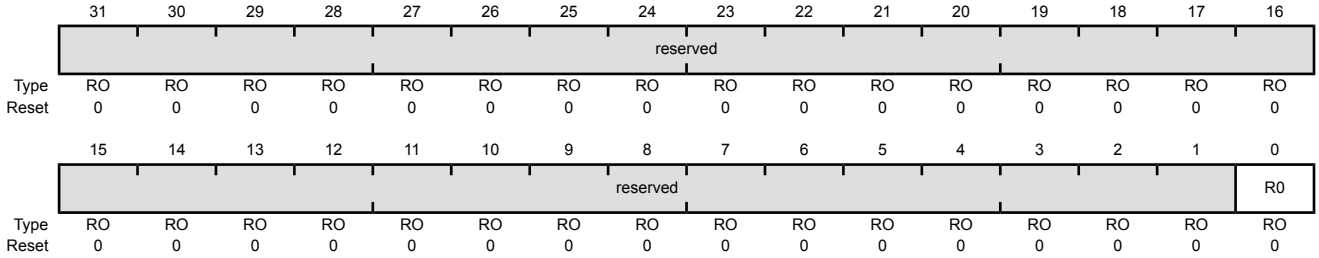
## Register 193: LCD Controller Peripheral Ready (PRLCD), offset 0xA90

The **PRLCD** register indicates whether the LCD Controller modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCLCD** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCLCD** bit is changed. A reset change is initiated if the corresponding **SRLCD** bit is changed from 0 to 1.

The **PRLCD** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

LCD Controller Peripheral Ready (PRLCD)

Base 0x400F.E000
Offset 0xA90
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | LCD Controller Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | LCD Controller module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | LCD Controller module 0 is ready for access. |

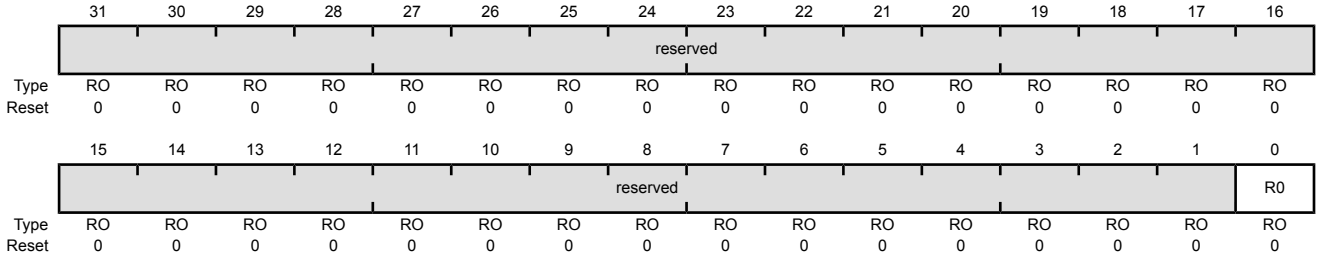
## Register 194: 1-Wire Peripheral Ready (PROWIRE), offset 0xA98

The **PROWIRE** register indicates whether the 1-Wire modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCOWIRE** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCOWIRE** bit is changed. A reset change is initiated if the corresponding **SROWIRE** bit is changed from 0 to 1.

The **PROWIRE** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

1-Wire Peripheral Ready (PROWIRE)

Base 0x400F.E000
Offset 0xA98
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | 1-Wire Module 0 Peripheral Ready<br><br>Value Description<br>0 HIM module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.<br>1 1-Wire module 0 is ready for access. |

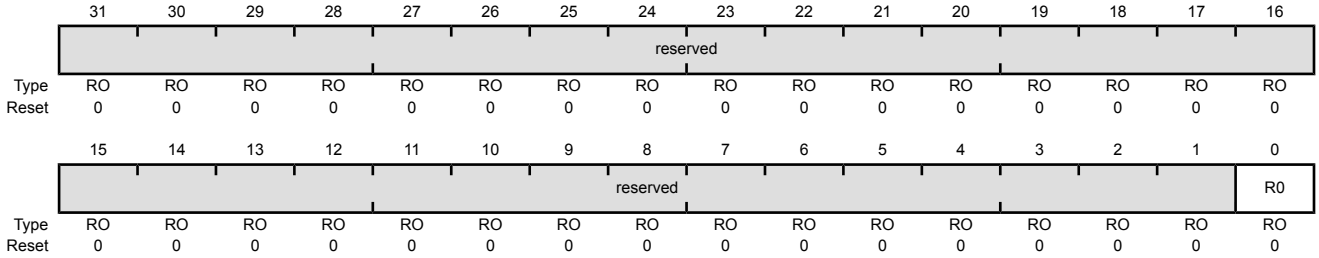
## Register 195: Ethernet MAC Peripheral Ready (PREMAC), offset 0xA9C

The **PREMAC** register indicates whether the Ethernet module is ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A power change is initiated if the corresponding **PCEMAC** bit is changed from 0 to 1. A Run mode clocking change is initiated if the corresponding **RCGCEMAC** bit is changed. A reset change is initiated if the corresponding **SREMAC** bit is changed from 0 to 1.

The **PREMAC** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

Ethernet MAC Peripheral Ready (PREMAC)

Base 0x400F.E000
Offset 0xA9C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | R0 | RO | 0 | Ethernet MAC Module 0 Peripheral Ready |

| Value | Description |
|---|---|
| 0 | Ethernet MAC module 0 is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence. |
| 1 | Ethernet MAC module 0 is ready for access. |

### Register 196: Unique ID 0 (UNIQUEID0), offset 0xF20

### Register 197: Unique ID 1 (UNIQUEID1), offset 0xF24

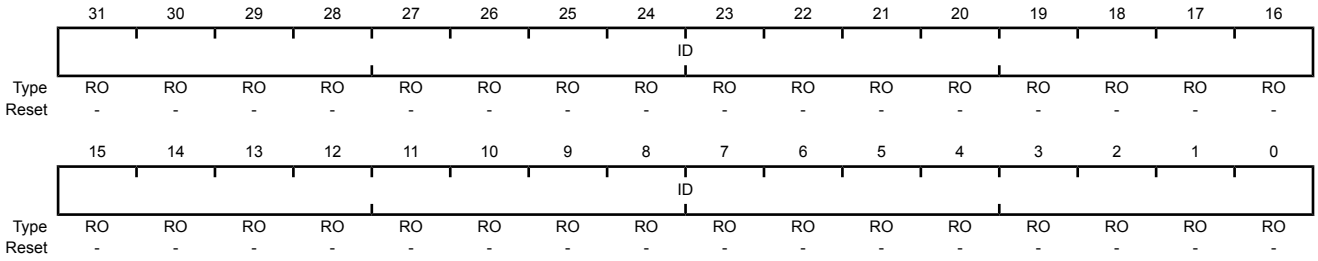
### Register 198: Unique ID 2 (UNIQUEID2), offset 0xF28

### Register 199: Unique ID 3 (UNIQUEID3), offset 0xF2C

These registers contain a unique 128-bit identifier that cannot be modified by the user. This value is unique to each individual die but is not a random value. This unique device identifier can be used to initiate secure boot processes or as a serial number for USB or other end applications.

Unique ID n (UNIQUEIDn)

Base 0x400F.E000
Offset 0xF20
Type RO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ID | RO | - | Unique ID<br>The result of registers 0-3 concatenated defines the unique 128-bit device identifier. |

## 5.6 Cryptographic System Control Register Description (CCM Offset)

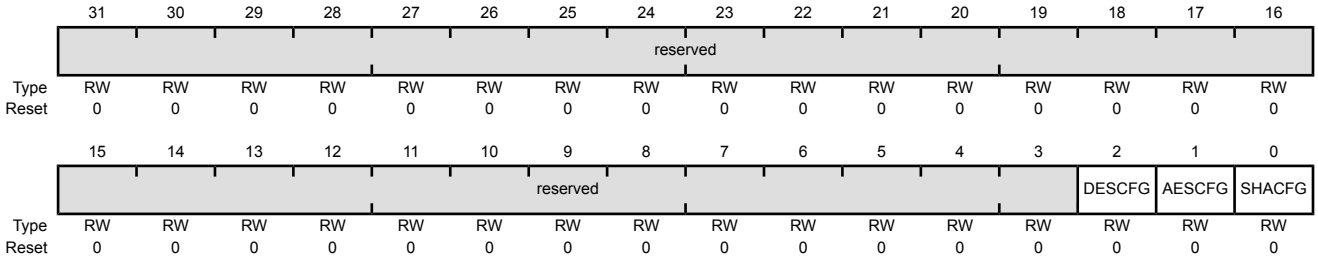
This section lists and describes the system control registers for the CRC and Cryptographic (AES, DES, SHA) Modules. Registers in this section are relative to the CCM base address of 0x4403.0000.

## Register 200: Cryptographic Modules Clock Gating Request (CCMCGREQ), offset 0x204

The **Cryptographic Modules Clock Gating Request (CCMCGREQ)** register is written to enable or disable clock gating in the AES, DES, and SHA/MD5 Module.

Cryptographic Modules Clock Gating Request (CCMCGREQ)

Base 0x400F.E000
Offset 0x204
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RW | 0x00000 | 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. |
| 2 | DESCFG | RW | 0 | DES Clock Gating Request<br><br>Value Description<br>0 Request to un-gate clock gating<br>1 Request to gate the clock. |
| 1 | AESCFG | RW | 0 | AES Clock Gating Request<br><br>Value Description<br>0 Request to un-gate clock gating<br>1 Request to gate the clock. |
| 0 | SHACFG | RW | 0 | SHA/MD5 Clock Gating Request<br><br>Value Description<br>0 Request to un-gate clock gating<br>1 Request to gate the clock. |

# 6 Processor Support and Exception Module

This module is an AHB peripheral that handles system-level Cortex-M4 FPU exceptions. For functions with registers mapped into this aperture, if the function is not available on a device, then all writes to the associated registers are ignored and reads return zeros.

## 6.1 Functional Description

The System Exception module provides control and status of the system-level interrupts. All the interrupt events are ORed together before being sent to the interrupt controller, so the System Exception module 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 by reading the **System Exception Masked Interrupt Status (SYSEXCMIS)** register. The interrupt events that can trigger a controller-level interrupt are defined in the **System Exception Interrupt Mask (SYSEXCIM)** register by setting the corresponding interrupt mask bits. If interrupts are not used, the raw interrupt status is always visible via the **System Exception Raw Interrupt Status (SYSEXCRIS)** register. Interrupts are always cleared (for both the **SYSEXCMIS** and **SYSEXCRIS** registers) by writing a 1 to the corresponding bit in the **System Exception Interrupt Clear (SYSEXCIC)** register.

## 6.2 Register Map

Table 6-1 on page 555 lists the System Exception module registers. The offset listed is a hexadecimal increment to the register's address, relative to the System Exception base address of 0x400F.9000.

**Note:** Spaces in the System Exception register space that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

**Table 6-1. System Exception Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | SYSEXCRIS | RO | 0x0000.0000 | System Exception Raw Interrupt Status | 556 |
| 0x004 | SYSEXCIM | RW | 0x0000.0000 | System Exception Interrupt Mask | 558 |
| 0x008 | SYSEXCMIS | RO | 0x0000.0000 | System Exception Masked Interrupt Status | 560 |
| 0x00C | SYSEXCIC | W1C | 0x0000.0000 | System Exception Interrupt Clear | 562 |

## 6.3 Register Descriptions

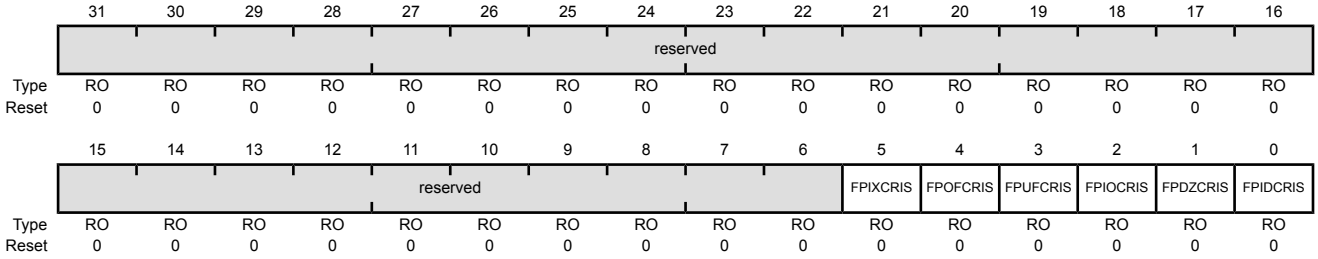
All addresses given are relative to the System Exception base address of 0x400F.9000.

## Register 1: System Exception Raw Interrupt Status (SYSEXCRIS), offset 0x000

The **SYSEXCRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

System Exception Raw Interrupt Status (SYSEXCRIS)

Base 0x400F.9000
Offset 0x000
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.00 | 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 | FPIXCRIS | RO | 0 | Floating-Point Inexact Exception Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A floating-point inexact exception has occurred.<br><br>This bit is cleared by writing a 1 to the `IXCIC` bit in the **SYSEXCIC** register. |
| 4 | FPOFCRIS | RO | 0 | Floating-Point Overflow Exception Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A floating-point overflow exception has occurred.<br><br>This bit is cleared by writing a 1 to the `OFCIC` bit in the **SYSEXCIC** register. |
| 3 | FPUFCRIS | RO | 0 | Floating-Point Underflow Exception Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A floating-point underflow exception has occurred.<br><br>This bit is cleared by writing a 1 to the `UFCIC` bit in the **SYSEXCIC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | FPIOCRIS | RO | 0 | Floating-Point Invalid Operation Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A floating-point invalid operation exception has occurred.<br><br>This bit is cleared by writing a 1 to the IOCIC bit in the **SYSEXCIC** register. |
| 1 | FPDZCRIS | RO | 0 | Floating-Point Divide By 0 Exception Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A floating-point divide by 0 exception has occurred.<br><br>This bit is cleared by writing a 1 to the DZCIC bit in the **SYSEXCIC** register. |
| 0 | FPIDCRIS | RO | 0 | Floating-Point Input Denormal Exception Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A floating-point input denormal exception has occurred.<br><br>This bit is cleared by writing a 1 to the IDCIC bit in the **SYSEXCIC** register. |

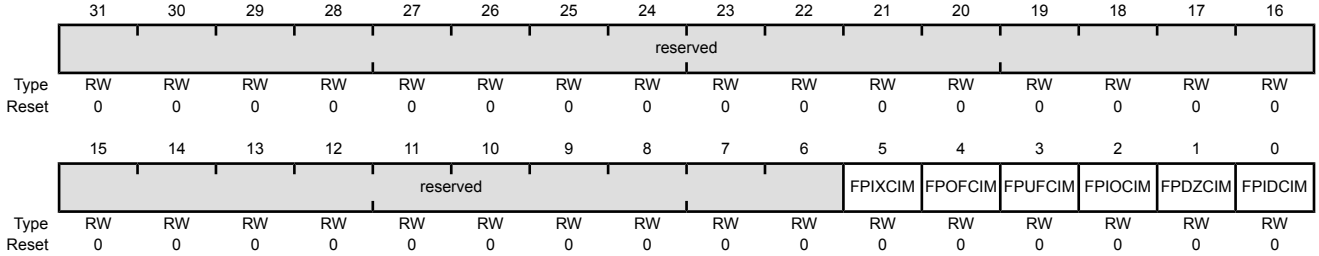
## Register 2: System Exception Interrupt Mask (SYSEXCIM), offset 0x004

The **SYSEXCIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

System Exception Interrupt Mask (SYSEXCIM)

Base 0x400F.9000
Offset 0x004
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RW | 0x0000.00 | 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 | FPIXCIM | RW | 0 | Floating-Point Inexact Exception Interrupt Mask<br>Value Description<br>0 The FPIXCRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the FPISCRIS bit in the **SYSEXCRIS** register is set. |
| 4 | FPOFCIM | RW | 0 | Floating-Point Overflow Exception Interrupt Mask<br>Value Description<br>0 The FPOFCIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the FPOFCRIS bit in the **SYSEXCRIS** register is set. |
| 3 | FPUFCIM | RW | 0 | Floating-Point Underflow Exception Interrupt Mask<br>Value Description<br>0 The FPUFCRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the FPUFCRIS bit in the **SYSEXCRIS** register is set. |

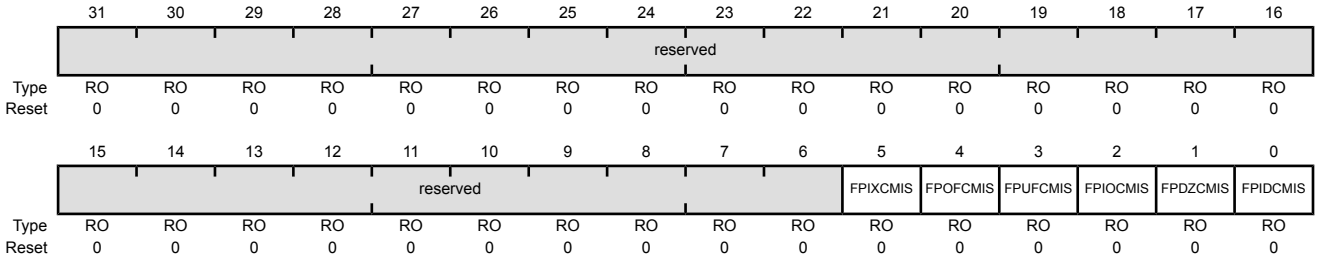
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | FPIOCIM | RW | 0 | Floating-Point Invalid Operation Interrupt Mask<br><br>Value Description<br>0 The FPIOCRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the FPIOCRIS bit in the **SYSEXCRIS** register is set. |
| 1 | FPDZCIM | RW | 0 | Floating-Point Divide By 0 Exception Interrupt Mask<br><br>Value Description<br>0 The FPDZCRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the FPDZCRIS bit in the **SYSEXCRIS** register is set. |
| 0 | FPIDCIM | RW | 0 | Floating-Point Input Denormal Exception Interrupt Mask<br><br>Value Description<br>0 The FPIDCRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the FPIDCRIS bit in the **SYSEXCRIS** register is set. |

## Register 3: System Exception Masked Interrupt Status (SYSEXCMIS), offset 0x008

The **SYSEXCMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

System Exception Masked Interrupt Status (SYSEXCMIS)

Base 0x400F.9000
Offset 0x008
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.00 | 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 | FPIXCMIS | RO | 0 | Floating-Point Inexact Exception Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to an inexact exception.<br><br>This bit is cleared by writing a 1 to the FPIXCIC bit in the **SYSEXCIC** register. |
| 4 | FPOFCMIS | RO | 0 | Floating-Point Overflow Exception Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to an overflow exception.<br><br>This bit is cleared by writing a 1 to the FPOFCIC bit in the **SYSEXCIC** register. |
| 3 | FPUFCMIS | RO | 0 | Floating-Point Underflow Exception Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to an underflow exception.<br><br>This bit is cleared by writing a 1 to the FPUFCIC bit in the **SYSEXCIC** register. |

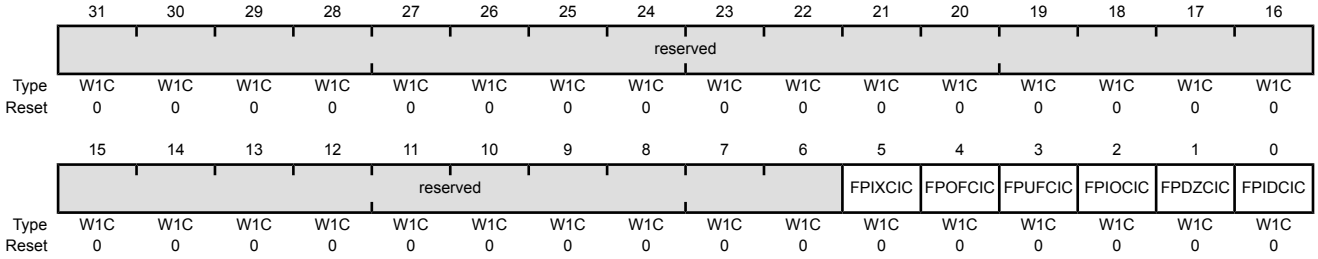
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | FPIOCMIS | RO | 0 | Floating-Point Invalid Operation Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to an invalid operation.<br><br>This bit is cleared by writing a 1 to the FPIOCIC bit in the **SYSEXCIC** register. |
| 1 | FPDZCMIS | RO | 0 | Floating-Point Divide By 0 Exception Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a divide by 0 exception.<br><br>This bit is cleared by writing a 1 to the FPDZCIC bit in the **SYSEXCIC** register. |
| 0 | FPIDCMIS | RO | 0 | Floating-Point Input Denormal Exception Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to an input denormal exception.<br><br>This bit is cleared by writing a 1 to the FPIDCIC bit in the **SYSEXCIC** register. |

## Register 4: System Exception Interrupt Clear (SYSEXCIC), offset 0x00C

The **SYSEXCIC** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

System Exception Interrupt Clear (SYSEXCIC)

Base 0x400F.9000
Offset 0x00C
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | W1C | 0x0000.00 | 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 | FPIXCIC | W1C | 0 | Floating-Point Inexact Exception Interrupt Clear<br>Writing a 1 to this bit clears the FPIXCRIS bit in the **SYSEXCRIS** register and the FPIXCMIS bit in the **SYSEXCMIS** register. |
| 4 | FPOFCIC | W1C | 0 | Floating-Point Overflow Exception Interrupt Clear<br>Writing a 1 to this bit clears the FPOFCRIS bit in the **SYSEXCRIS** register and the FPOFCMIS bit in the **SYSEXCMIS** register. |
| 3 | FPUFCIC | W1C | 0 | Floating-Point Underflow Exception Interrupt Clear<br>Writing a 1 to this bit clears the FPUFCRIS bit in the **SYSEXCRIS** register and the FPUFCMIS bit in the **SYSEXCMIS** register. |
| 2 | FPIOCIC | W1C | 0 | Floating-Point Invalid Operation Interrupt Clear<br>Writing a 1 to this bit clears the FPIOCRIS bit in the **SYSEXCRIS** register and the FPIOCMIS bit in the **SYSEXCMIS** register. |
| 1 | FPDZCIC | W1C | 0 | Floating-Point Divide By 0 Exception Interrupt Clear<br>Writing a 1 to this bit clears the FPDZCRIS bit in the **SYSEXCRIS** register and the FPDZCMIS bit in the **SYSEXCMIS** register. |
| 0 | FPIDCIC | W1C | 0 | Floating-Point Input Denormal Exception Interrupt Clear<br>Writing a 1 to this bit clears the FPIDCRIS bit in the **SYSEXCRIS** register and the FPIDCMIS bit in the **SYSEXCMIS** register. |

# 7 Hibernation Module

The Hibernation Module manages removal and restoration of power to provide a means for reducing system power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal or at a certain time using the built-in Real-Time Clock (RTC). The Hibernation module can be independently supplied from an external battery or an auxiliary power supply.

The Hibernation also integrates a tamper module which provides mechanisms to detect, respond to, and log system tampering events. The Tamper module is designed to be low power and operate from either a battery or the MCU I/O voltage supply.

The Hibernation module has the following features:

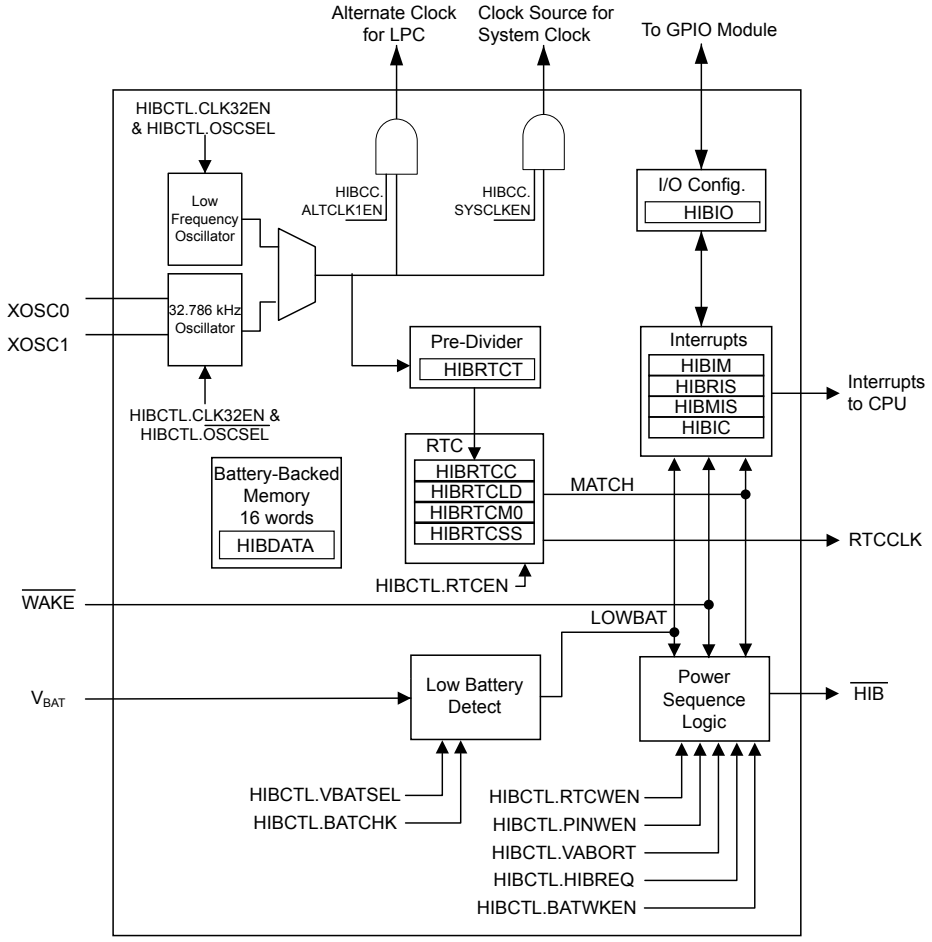
- 32-bit real-time seconds counter (RTC) with 1/32,768 second resolution and a 15-bit sub-seconds counter
  - 32-bit RTC seconds match register and a 15-bit sub seconds match for timed wake-up and interrupt generation with 1/32,768 second resolution
  - RTC predivider trim for making fine adjustments to the clock rate
- Hardware Calendar Function
  - Year, Month, Day, Day of Week, Hours, Minutes, Seconds
  - Four-year leap compensation
  - 24-hour or AM/PM configuration
- Two mechanisms for power control
  - System power control using discrete external regulator
  - On-chip power control using internal switches under register control
- $V_{DD}$ supplies power when valid, even if $V_{BAT} > V_{DD}$
- Dedicated pin for waking using an external signal
- Capability to configure external reset ($\overline{RST}$) pin and/or up to four GPIO port pins as wake source, with programmable wake level
- Tamper Functionality
  - Support for four tamper inputs
  - Configurable level, weak pull-up, and glitch filter
  - Configurable tamper event response
  - Logging of up to four tamper events
  - Optional BBRAM erase on tamper detection
  - Tamper wake from hibernate capability

  - Hibernation clock input failure detect with a switch to the internal oscillator on detection

- RTC operational and hibernation memory valid as long as $V_{DD}$ or $V_{BAT}$ is valid
- Low-battery detection, signaling, and interrupt generation, with optional wake on low battery
- GPIO pin state can be retained during hibernation
- Clock source from an internal low frequency oscillator (HIB LFIOSC) or a 32.768-kHz external crystal or oscillator
- Sixteen 32-bit words of battery-backed memory to save state during hibernation
- Programmable interrupts for:
  - RTC match
  - External wake
  - Low battery

## 7.1 Block Diagram

**Figure 7-1. Hibernation Module Block Diagram**




**Note:** References to alternate clock to LPC only apply to devices which have LPC.

## 7.2 Signal Description

The following table lists the external signals of the Hibernation module and describes the function of each.

The RTCCLK and TMPR[3:0] signals are alternate functions for a GPIO signal and defaults to be a GPIO signal at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the RTCCLK and TMPR[3:0] signals. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign each signal to its specified GPIO port pin. In addition, the AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the proper HIB function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

The remaining signals have a fixed pin assignment and function.

**Note:** In addition to the Hibernation signals that are part of the Hibernation Module, GPIO pins K[7:4] can be configured as external wake sources. Refer to “Waking from Hibernate” on page 578 for more information.

**Note:** Port pins PM[7:4] operate as Fast GPIO pads but support only 2-, 4-, 6-, and 8-mA drive capability. 10- and 12-mA drive are not supported. All standard GPIO register controls, except for the **GPIODR12R** register, apply to these port pins. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 772 and “Recommended GPIO Operating Characteristics” on page 2097 for more information.

**Table 7-1. Hibernate Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| GNDX | R18 | fixed | - | Power | GND for the Hibernation oscillator. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| HIB | M17 | fixed | O | TTL | An output that indicates the processor is in Hibernate mode. |
| RTCCLK | M1<br>W16<br>C12 | PC5 (7)<br>PK7 (5)<br>PP3 (7) | O | TTL | Buffered version of the Hibernation module's 32.768-kHz clock. This signal is not output when the part is in Hibernate mode and before being configured after power-on reset. |
| TMPR0 | N18 | PM7 | I/O | TTL | Tamper signal 0. |
| TMPR1 | N19 | PM6 | I/O | TTL | Tamper signal 1. |
| TMPR2 | G15 | PM5 | I/O | TTL | Tamper signal 2. |
| TMPR3 | M18 | PM4 | I/O | TTL | Tamper signal 3. |
| VBAT | P19 | fixed | - | Power | Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply. |
| WAKE | U18 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| XOSC0 | T18 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a crystal or a 32.768-kHz oscillator for the Hibernation module RTC. |
| XOSC1 | T19 | fixed | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |

## 7.3 Functional Description

The Hibernation module provides two mechanisms for power control:

- The first mechanism uses internal switches to control power to the Cortex-M4F as well as to most analog and digital functions while retaining I/O pin power (VDD3ON mode).
- The second mechanism controls the power to the microcontroller with a control signal (HIB) that signals an external voltage regulator to turn on or off.

The Hibernation module power source is supplied by $V_{DD}$ as long as it is within a valid range, even if $V_{BAT}$>$V_{DD}$. The Hibernation module also has an independent clock source to maintain a real-time

clock (RTC) when the system clock is powered down. Hibernate mode can be entered through one of two ways:

- The user initiates hibernation by setting the HIBREQ bit in the **Hibernation Control (HIBCTL)** register
- Power is arbitrarily removed from $V_{DD}$ while a valid $V_{BAT}$ is applied

Once in hibernation, the module signals an external voltage regulator to turn the power back on when an external pin ($\overline{\text{WAKE}}$, $\overline{\text{RST}}$ or a wake-enabled GPIO pin) is asserted or when the internal RTC reaches a certain value. The Hibernation module can also detect when the battery voltage is low and optionally prevent hibernation or wake from hibernation when the battery voltage falls below a certain threshold. Note that multiple wake sources can be configured at the same time to generate a wake signal such that any of them can wake the module.

When waking from hibernation, the $\overline{\text{HIB}}$ signal is deasserted. The return of $V_{DD}$ causes a POR to be executed. The time from when the $\overline{\text{WAKE}}$ signal is asserted to when code begins execution is equal to the wake-up time ($t_{WAKE_TO_HIB}$) plus the power-on reset time ($T_{POR}$).

### 7.3.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, hibernation registers must be written only with a timing gap between accesses. The delay time is $t_{HIB_REG_ACCESS}$, therefore software must guarantee that this delay is inserted between back-to-back writes to Hibernation registers or between a write followed by a read. The WC interrupt in the **HIBMIS** register can be used to notify the application when the Hibernation modules registers can be accessed. Alternatively, software may make use of the WRC bit in the **Hibernation Control (HIBCTL)** register to ensure that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **HIBCTL** for WRC=1 prior to accessing any hibernation register.

Back-to-back reads from Hibernation module registers have no timing restrictions. Reads are performed at the full peripheral clock rate.

### 7.3.2 Hibernation Clock Source

The HIB module can be clocked by one of three different clock sources:

- A 32.768-kHz oscillator
- An external 32.768-kHz clock source
- An internal low frequency oscillator (HIB LFIOSC)

Table 7-2 on page 568 summarizes the encodings for the bits in the **HIBCTL** register that are required for each clock source to be enabled. Note that CLK32EN must be set for any Hibernation clock source to be valid. The Hibernation module is not enabled until the CLK32EN bit is set. The HIB clock source is the source of the RTC Oscillator (RTCOSC), which can be selected as the system clock source by programming a 0x4 in the OSCSRC field of the **Run and Sleep Mode Configuration (RSCLKCFG)** register in the System Control Module. Please refer to “System Control” on page 231 for more information.

**Table 7-2. HIB Clock Source Configurations**

| HIB Clock Source | CLK32EN | OSCSEL | OSCBYP |
|---|---|---|---|
| 32.768 kHz Oscillator | 1 | 0 | 0 |
| External 32.768-kHz Clock Source | 1 | 0 | 1 |
| Low-frequency internal oscillator (HIB LFIOSC)[a] | 1 | 1 | 0 |

a. The frequency can have wide variations; refer to "Hibernation Clock Source Specifications" on page 2114 for more details.

To use an external crystal, a 32.768-kHz crystal is connected to the XOSC0 and XOSC1 pins. Alternatively, a 32.768-kHz oscillator can be connected to the XOSC0 pin, leaving XOSC1 unconnected. Care must be taken that the voltage amplitude of the 32.768-kHz oscillator is less than $V_{BAT}$, otherwise, the Hibernation module may draw power from the oscillator and not $V_{BAT}$ during hibernation. See Figure 7-2 on page 569 and Figure 7-3 on page 569.

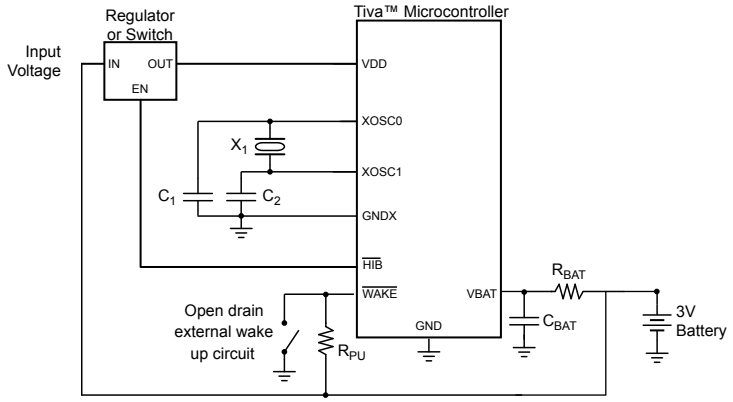
Alternatively, a low frequency oscillator source (HIB LFIOSC) present in the Hibernation module can be a clock source. (The frequency can have wide variations; refer to "Hibernation Clock Source Specifications" on page 2114 for more details.) The intent of this source is to provide an internal low power clock source to enable the use of the asynchronous pin wakes and memory storage without the requirement of an external crystal. To enable the HIB LFIOSC to be the clock source for the Hibernation module, both the OSCSEL bit and the CLK32EN bit in the **Hibernation Control (HIBCTL)** register must be set.

**Note:** The HIB low-frequency oscillator (HIB LFIOSC) has a wide frequency variation, therefore the RTC is not accurate when using this clock source. It is not recommended to use the HIB LFIOSC as an RTC clock source.

The Hibernation module is enabled by setting the CLK32EN bit of the **HIBCTL** register. The CLK32EN bit must be set before accessing any other Hibernation module register. The type of clock source used for the HIB module is selected by setting the OSCSEL and OSCBYP bit of the **HIBCTL** register. If the internal low frequency precision oscillator is used as the clock source, the OSCSEL bit should be set to a 1 at the same time the CLK32EN bit is set. If a crystal is used for the clock source, the software must leave a delay of $t_{HIBOSC_START}$ after writing to the CLK32EN bit and before any other accesses to the Hibernation module registers. The delay allows the crystal to power up and stabilize. If an external oscillator is used for the clock source, no delay is needed. When using an external clock source, the OSCBYP bit in the **HIBCTL** register should be set. When using a crystal clock source, the GNDX pin should be connected to digital ground along with the crystal load capacitors, as shown in Figure 7-2 on page 569. When using an external clock source, the GNDX pin should be connected to digital ground.

**Note:** In the figures below the parameters $R_{BAT}$ and $C_{BAT}$ have recommended values of 51Ω ±5% and 0.1µF ±5%, respectively. See "Hibernation Module" on page 2122 for more information.

**Figure 7-2. Using a Crystal as the Hibernation Clock Source with a Single Battery Source**




**Note:** Some devices may not supply the GNDX signal. If GNDX is absent, the crystal load capacitors can be tied to GND externally. See "Signal Tables" on page 2028 for pins specific to your device.

$X_1$ = Crystal frequency is $f_{XOSC_XTAL}$.

$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

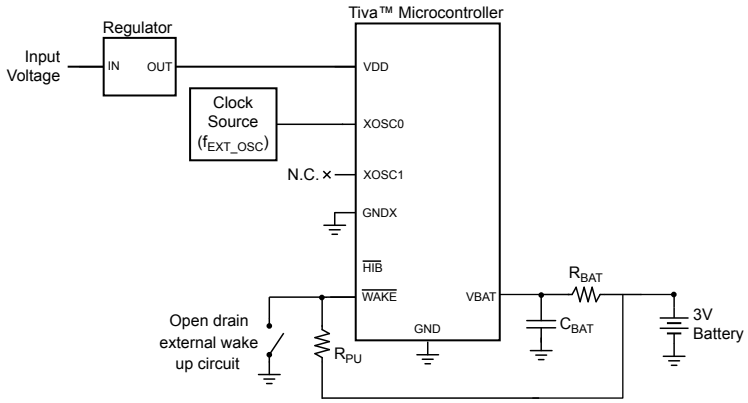
$R_{PU}$ = Pull-up resistor is 200 kΩ

$R_{BAT}$ = 51Ω ±5%

$C_{BAT}$ = 0.1µF ±20%

See "Hibernation Clock Source Specifications" on page 2114 for specific parameter values.

**Figure 7-3. Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode**




**Note:** Some devices may not supply a GNDX signal. See "Signal Tables" on page 2028 for pins specific to your device.

$R_{PU}$ = Pull-up resistor is 1 MΩ

$R_{BAT}$ = 51Ω ±5%

$C_{BAT}$ = 0.1µF ±20%

#### 7.3.2.1 Hibernate Clock Output RTCOSC

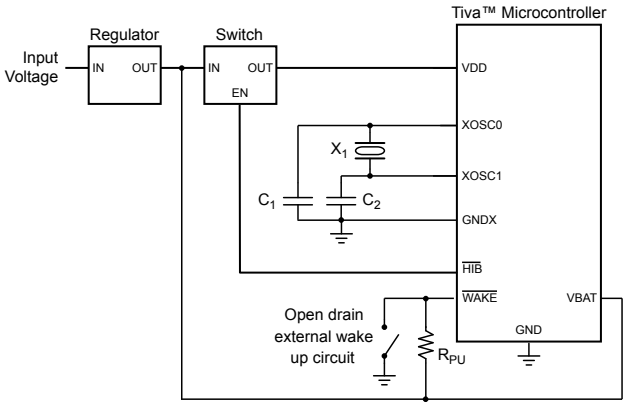
The clock source that is configured as the HIB clock has the option of becoming an internal output, RTCOSC, and being selected as the clock source for the system clock. To enable RTCOSC as a system clock source, the SYSCLKEN bit must be set in the **Hibernate Clock Control (HIBCC)** register.

### 7.3.3 System Implementation

Several different system configurations are possible when using the Hibernation module:

- Using a single battery source, where the battery provides both $V_{DD}$ and $V_{BAT}$, as shown in Figure 7-2 on page 569.
- Using the VDD3ON mode, where $V_{DD}$ continues to be powered in hibernation, allowing the GPIO pins to retain their states, as shown in Figure 7-3 on page 569. In this mode, $V_{DDC}$ is powered off internally. In VDD3ON mode, the RETCLR bit in the **HIBCTL** register must be set so that after power is reapplied, GPIO retention is held until software clears the bit. GPIO retention is released when software writes a 0 to the RETCLR bit.
- Using separate sources for $V_{DD}$ and $V_{BAT}$. In this mode, additional circuitry is required for system start-up without a battery or with a depleted battery.
- Using a regulator to provide both $V_{DD}$ and $V_{BAT}$ with a switch enabled by $\overline{\text{HIB}}$ to remove $V_{DD}$ during hibernation as shown in Figure 7-4 on page 570.

**Figure 7-4. Using a Regulator for Both $V_{DD}$ and $V_{BAT}$**




**Note:** Some devices may not supply a GNDX signal. See "Signal Tables" on page 2028 for pins specific to your device.

Adding external capacitance to the $V_{BAT}$ supply reduces the accuracy of the low-battery measurement and should be avoided if possible. The diagrams referenced in this section only show the connection to the Hibernation pins and not to the full system.

If the application does not require the use of the Hibernation module, refer to "Connections for Unused Signals" on page 2093. In this situation, the HIB bit in the **Hibernation Run Mode Clock Gating Control (RCGCHIB)** register must be cleared, disabling the system clock to the Hibernation module and Hibernation module registers are not accessible.

### 7.3.4 Battery Management

**Important:** System-level factors may affect the accuracy of the low-battery detect circuit. The designer should consider battery type, discharge characteristics, and a test load during battery voltage measurements.

The Hibernation module can be independently powered by a battery or an auxiliary power source using the VBAT pin. The module can monitor the voltage level of the battery and detect when the voltage drops below $V_{LOWBAT}$. The voltage threshold can be between 1.9 V and 2.5 V and is configured using the VBATSEL field in the **HIBCTL** register. The module can also be configured so that it does not go into Hibernate mode if the battery voltage drops below this threshold. In addition, battery voltage is monitored while in hibernation, and the microcontroller can be configured to wake from hibernation if the battery voltage goes below the threshold using the BATWKEN bit in the **HIBCTL** register.

The Hibernation module is designed to detect a low-battery condition and set the LOWBAT bit of the **Hibernation Raw Interrupt Status (HIBRIS)** register when this occurs. If the VABORT bit in the **HIBCTL** register is also set, then the module is prevented from entering Hibernate mode when a low-battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see “Interrupts and Status” on page 580).

### 7.3.5 Real-Time Clock

The RTC module is designed to keep wall time. The RTC can operate in seconds counter mode or calendar mode. A 32.768 kHz clock source along with a 15-bit predivider reduces the clock to 1 Hz. The 1 Hz clock is used to increment the 32-bit counter and keep track of seconds. In calendar mode, registers are provided which support the tracking of date, month, year and day-of-week. A match register can be configured to interrupt or wake the system from hibernate. In addition, a software trim register is implemented to allow the user to compensate for oscillator inaccuracies using software.

#### 7.3.5.1 RTC Counter - Seconds/Subseconds Mode

The clock signal to the RTC is provided by either of the 32.768-kHz clock sources available to the Hibernation module. The **Hibernation RTC Counter (HIBRTCC)** register displays the seconds value. The **Hibernation RTC Sub Seconds register (HIBRTCSS)** is provided for additional time resolution of an application requiring less than one-second divisions.

The RTC is enabled by setting the RTCEN bit of the **HIBCTL** register. The RTCEN bit is also used along with the CALEN bit in the **Hibernation Calendar Control (HIBCALCTL)** register to enable the calender. Thus, if the calendar is enabled, the RTC registers, **HIBRTCC**, **HIBRTCSS**, **HIBRTCM0** and **HIBRTCLD**, cannot be used. The RTC counter and sub-seconds counters begin counting immediately once RTCEN is set. Both counters count up. The RTC continues counting as long as the RTC is enabled and a valid $V_{BAT}$ is present, regardless of whether $V_{DD}$ is present or if the device is in hibernation.

The **HIBRTCC** register is set by writing the **Hibernation RTC Load (HIBRTCLD)** register. A write to the **HIBRTCLD** register clears the 15-bit sub-seconds counter field, RTCSSC, in the **HIBRTCSS** register. To ensure a valid read of the RTC value, the **HIBRTCC** register should be read first, followed by a read of the RTCSSC field in the **HIBRTCSS** register and then a re-read of the **HIBRTCC** register. If the two values for the **HIBRTCC** are equal, the read is valid. By following this procedure, errors in the application caused by the **HIBRTCC** register rolling over by a count of 1 during a read of the RTCSSC field are prevented. The RTC can be configured to generate an alarm by setting the RTCAL0 bit in the **HIBIM** register. When an RTC match occurs, an interrupt is generated and displayed in the **HIBRIS** register. Refer to “RTC Match - Seconds/Subseconds Mode” on page 572 for more information.

If the RTC is enabled, only a cold POR, where both $V_{BAT}$ and $V_{DD}$ are removed, resets the RTC registers. If any other reset occurs while the RTC is enabled, such as an external $\overline{RST}$ assertion or BOR reset, the RTC is not reset. The RTC registers can be reset under any type of system reset as long as the RTC, external wake pins and tamper pins are not enabled.

A buffered version of the 32.768-kHz signal Hibernate clock source is available on the RTCCLK signal output, which is muxed with a GPIO pin. The RTCCLK signal can be the external 32.786-kHz clock source or the HIB LFIOSC depending on the value of the OSCSEL bit in the **HIBCTL** register. See "Signal Description" on page 565 or pin mux information and "General-Purpose Input/Outputs (GPIOs)" on page 772 for additional details on initialization and configuration of this signal. The pin does not output RTCCLK when Hibernate mode is active or before the RTCCLK GPIO digital function has been selected through the **GPIO Digital Enable (GPIODEN)** register in the GPIO module. This includes selecting the RTCCLK signal as an output source in the **GPIO Port Control (GPIOPCTL)** register and setting the SYSCLKEN bit within the **Hibernate Clock Control (HIBCC)** register.

**Note:** The HIB low-frequency oscillator (HIB LFIOSC) has a wide frequency variation, therefore the RTC is not accurate when using this clock source. In addition, the RTCCLK signal may not meet the specification shown in Table 32-30 on page 2122.

#### 7.3.5.2 RTC Match - Seconds/Subseconds Mode

The Hibernation module includes a 32-bit match register, **HIBRTCM0**, which is compared to the value of the RTC 32-bit counter, **HIBRTCC**. The match functionality also extends to the sub-seconds counter. The 15-bit field (RTCSSM) in the **HIBRTCSS** register is compared to the value of the 15-bit sub-seconds counter. When a match occurs, the RTCALT0 bit is set in the **HIBRIS** register. For applications using Hibernate mode, the processor can be programmed to wake from Hibernate mode by setting the RTCWEN bit in the **HIBCTL** register. The processor can also be programmed to generate an interrupt to the interrupt controller by setting the RTCALT0 bit in the **HIBIM** register.

The match interrupt generation takes priority over an interrupt clear. Therefore, writes to the RTCALT0 bit in the **Hibernation Interrupt Clear (HIBIC)** register do not clear the RTCALT0 bit if the **HIBRTCC** value and the **HIBRTCM0** value are equal. There are several methodologies to avoid this occurrence, such as writing a new value to the **HIBRTCLD** register prior to writing the **HIBIC** to clear the RTCALT0. Another example, would be to disable the RTC and re-enable the RTC by clearing and setting the RTCEN bit in the **HIBCTL** register.

**Note:** A Hibernate request made while a match event is valid causes the module to immediately wake up. This occurs when the RTCWEN bit is set and the RTCALT0 bit in the **HIBRIS** register is set at the same time the HIBREQ bit in the **HIBCTL** register is written to a 1. This can be avoided by clearing the RTCAL0 bit in the **HIBRIS** register by writing a 1 to the corresponding bit in the **HIBIC** register before setting the HIBREQ bit. Another example would be to disable the RTC and re-enable the RTC by clearing and setting the RTCEN bit in the **HIBCTL** register.

#### 7.3.5.3 RTC Calendar

The RTC Calendar function is selected by setting the CALEN bit in the **HIB Calendar Control (HIBCALCTL)** register. In this mode, six 32-bit registers provide the read (**HIBCAL0/1**), match (**HIBCALM0/1**), and load (**HIBCALLD0/1**) interface. The standard RTC registers: **HIBRTCC**, **HIBRTCLD**, **HIBRTCSS**, and **HIBRTCM0** are disabled when the calendar function is enabled and read back as all 0s in this mode. In addition, writes have no effect on these registers when the calendar function is enabled.

The **Hibernation Calendar n (HIBCALn)**, **Hibernation Calendar Match (HIBCALMn)** and **Hibernation Calendar Load (HIBCALLDn)** register fields are written or stored in hexadecimal.

When reading the **Hibernation Calendar n (HIBCALn)** registers, the status of the VALID bit in the **HIBCAL0/1** register must be checked to ensure the registers are in sync before reading.

The calendar function will keep track of the following:

- Seconds (0-59 seconds)
- Minutes (0-59 minutes)
- Hours (0-23 or 0-11 hours with an AM/PM option)
- Day of the week (0-6)
- Day of the month (1-31 days)
- Month (1-12 months)
- Year (00-99 years)

The hours may be reported with AM/PM or 24-hour based on the CAL24 bit in the **HIBCALCTL** register. The leap year compensation is handled within the calendar function. The number of days in February are adjusted to 29 whenever the year is divisible by four.

***RTC Calendar Match***

The HIB Calendar Match function can be used to generate an interrupt on a match of seconds, minutes, hours, and day of month. The day of the week, year and month are not included in the match function. To ignore a match function for the hours, minutes, or seconds, set each of the upper two bits to 1 in the respective fields of the **HIBCALMn** register. To ignore the day of the month, set the DOM field to all zeros in the **HIBCALM1** register. If a match occurs in any field, the RTCALT0 bit is set in the **HIBRIS** register.

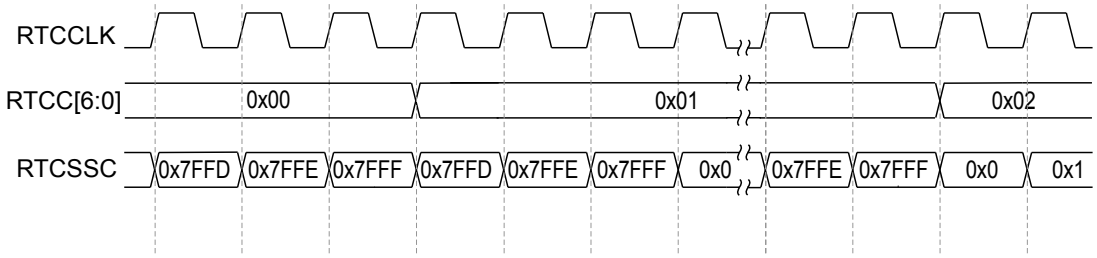
### 7.3.5.4 RTC Trim

The RTC counting rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register, **HIBRTCT**. This register has a nominal value of 0x7FFF, and is used for one second out of every 64 seconds in RTC counter mode, when bits [5:0] in the **HIBRTCC** register change from 0x00 to 0x01, to divide the input clock. Trim is applied every 60 seconds in calendar mode. This configuration allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from 0x7FFF. The predivider trim should be adjusted up from 0x7FFF in order to slow down the RTC rate and down from 0x7FFF in order to speed up the RTC rate.

Care must be taken when using trim values that are near to the sub seconds match value in the **HIBRTCSS** register. It is possible when using trim values above 0x7FFF to receive two match interrupts for the same counter value. In addition, it is possible when using trim values below 0x7FFF to miss a match interrupt.

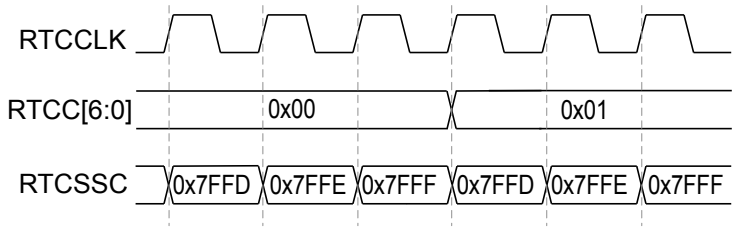
In the case of a trim value above 0x7FFF, when the RTCSSC value in the **HIBRTCSS** register reaches 0x7FFF, the RTCC value increments from 0x0 to 0x1 while the RTCSSC value is decreased by the trim amount. The RTCSSC value is counted up again to 0x7FFF before rolling over to 0x0 to begin counting up again. If the match value is within this range, the match interrupt is triggered twice. For example, as shown in Figure 7-5 on page 574, if the match interrupt was configured with RTCM0=0x1 and RTCSSM=0x7FFD, two interrupts would be triggered.

**Figure 7-5. Counter Behavior with a TRIM Value of 0x8002**




In the case of a trim value below 0x7FFF, the RTCSSC value is advanced from 0x7FFF to the trim value while the RTCC value is incremented from 0x0 to 0x1. If the match value is within that range, the match interrupt is not triggered. For example, as shown in Figure 7-6 on page 574, if the match interrupt was configured with RTCM0=0x1 and RTCSSM=0x2,an interrupt would never be triggered.

**Figure 7-6. Counter Behavior with a TRIM Value of 0x7FFC**



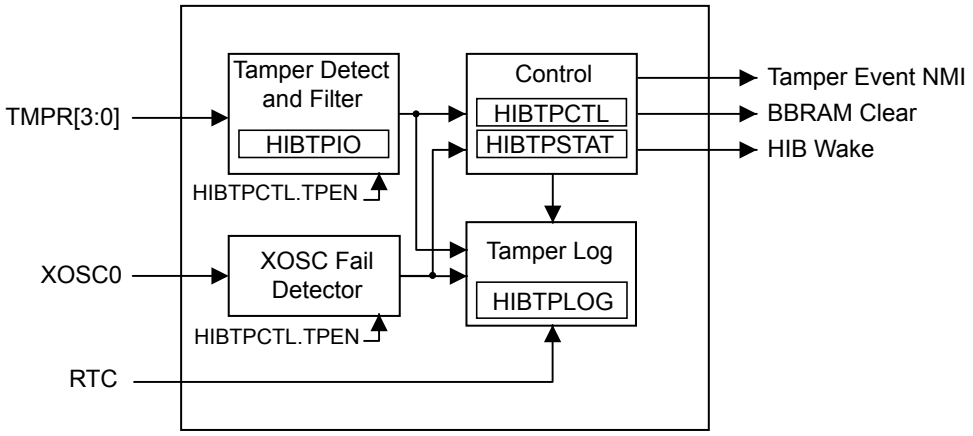

## 7.3.6 Tamper

The Tamper module provides a user with mechanisms to detect, respond to, and log system tampering events. The Tamper module is designed to be low power and operate either from a battery or the MCU I/O voltage supply. This module is a sub-module of the Hibernate module.

### 7.3.6.1 Tamper Block Diagram

Figure 7-7 on page 574 shows the Tamper block diagram.

**Figure 7-7. Tamper Block Diagram**



### 7.3.6.2 Functional Description

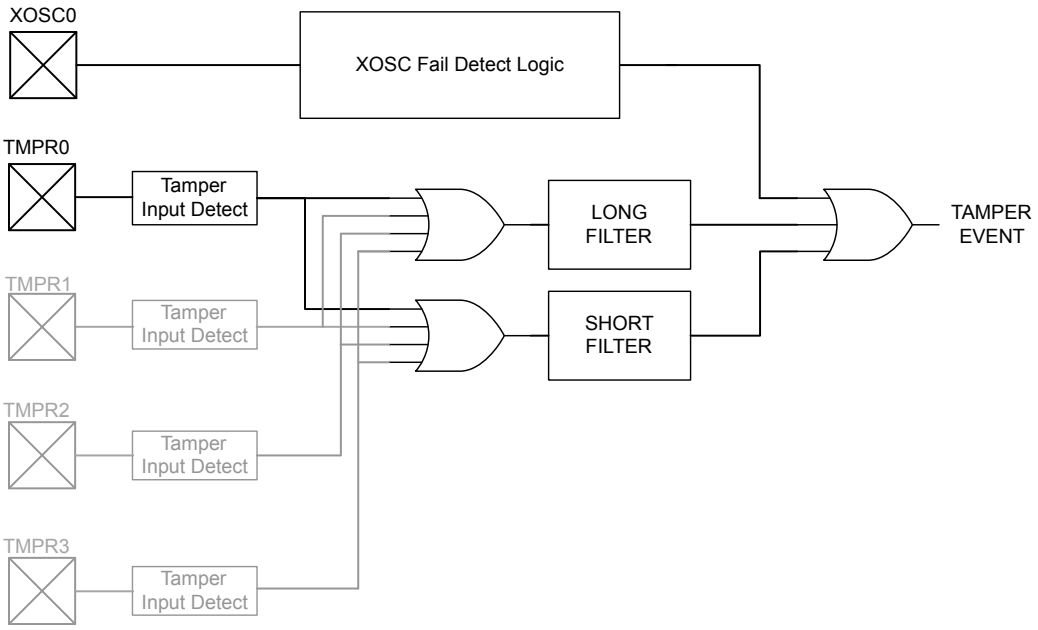
The Tamper module provides mechanisms to detect, respond, and log system tamper events. A tamper event is detected by state transitions on up to four GPIOs. The module may respond to a tamper event by clearing all or part of the hibernate module memory, generating a tamper event signal to the System Control module. The event will also be logged with a RTC time stamp to allow for tamper investigation.

#### *Tamper Detection*

Qualified tamper events are detected through an XOSCn pin failure or through tamper I/O level matches which pass through a glitch filter. Tamper I/O pad events are detected by comparing the level on a tamper I/O pad with an expected value. The tamper I/O is sampled using the hibernate clock source and when the glitch filtering is enabled, must be stable for about 100 ms. This provides debounce filtering of a breakaway switch as a results of a drop impact. The tamper module contains one long glitch filter and one short glitch filter which uses an OR of the inputs as shown in Figure 7-8 on page 575. This implies if two Tamper inputs are asserted and one deasserts, the glitch filter runs to timeout or until the second Tamper input is deasserted. The glitch filter or tamper logging logic does not re-trigger if the tamper event match continues. The glitch filter resets on the deassertion of the tamper conditions or when a qualified tamper event is logged.

If the XOSCn pins are enabled for use with the Hibernation module and subsequently fail, a tamper event is detected and is indicated by the STATE field in the **HIB Tamper Status (HIBTPSTAT)** register. In addition, the XOSCST and XOSCFAIL bits can be read for further details on the external oscillator source state.

**Figure 7-8. Tamper Pad with Glitch Filtering**




#### *Tamper Event Responses*

There are many responses to a tamper event including clearing some or all of Hibernate memory and generating a tamper signal to the System Control Module. The descriptions of the possible event responses follows.

- Tamper Register Status

  The tamper status is indicated by the STATE bit field of the **HIB Tamper Status (HIBTPSTAT)** register. The register bits are reset to 0x0 on cold POR. When the tamper I/O is enabled/configured, the STATE field shows 0x1. The STATE field is set to 0x2 when a tamper event is detected. The software may reset the trigger source and the STATE field by writing to the TPCLR bit in the **HIBTPCTL** register.

- System Event Response

  When a tamper event is detected, an NMI is generated. The NMI handler is responsible for performing any other system responses, including a simulate POR. If the tamper event was an XOSC fail condition, the part switches to the HIB LFIOSC. Once XOSC is stable, the XOSC may be enabled as the clock source once again.

- Hibernate Memory Clearing

  On a tamper event, software has the option to clear all, the upper half, lower half, or none of the Hibernate memory. The feature is controlled through the MEMCLR field of the **HIBTPCTL** register.

- Wake from Hibernate

  A tamper event will assert a wake event to the MCU if the WAKE bit in the **HIBTPCTL** register is set.

#### *Tamper Event Logging*

Up to four tamper events are stored in **HIB Tamper Log n (HIBTPLOGn**) registers within the Hibernate module. When a tamper event occurs the following status is logged:

- The RTC seconds or calendar values of year, minutes, day of month, hours and seconds in the **HIBTPLOG0/2/4/6** registers

  **Note:** 24-hour mode must be used if RTC calendar mode is enabled. This mode is selected by setting the CAL24 bit in **HIB Calendar Control (HIBCALCTL)** register.

- The tamper status of the TMPRn pins and the XOSCn pins in the **HIBTPLOG1/3/5** registers. The **HIBTPLOG7** register captures the OR of all events occurring after the 3rd event is logged in the **HIBTPLOG5** register.

On the assertion of a qualified tamper event (rising edge) on any of the TMPRn pins or an XOSC failure signal, the current status of all tamper inputs are logged in the **HIBTPLOGn** register.

#### *Clearing a Tamper Event*

After a tamper event, the **HIB Tamper Log (HIBTPLOGn)** registers and the NMI to the processor may be cleared by writing a 1 to the TPCLR bit in the **HIBTPCTL** register. This clear status is reflected by the STATE bit in the **HIBSTPSTAT** register changing from 0x2 back to a 0x1. If the source of the tamper event comes from an XOSC failure, the clearing of a tamper event is delayed while the clock is switched to LFIOSC. The NMI interrupt handler may access the module immediately, but should read the **HIBTPLOGn** registers before issuing a tamper clear in the **HIBTPCTL** register.

**Note:** The **HIBTPLOG7** register is sticky and is only cleared by a Hibernate module reset.

#### *Tamper I/O Control*

Up to four tamper I/Os are available. These signals are individually enabled and the detection level can be configured per pin. Enabling the tamper IO will override all settings made in the GPIO module. Each tamper IO has a weak pull-up.

#### *Tamper Clocking*

The Hibernate clock is the clock source for the Tamper module. When an external oscillator is used and tamper is enabled, the external oscillator is monitored by the Tamper module. If the external oscillator stops for any reason, the XOSCFAIL bit is set in the **HIBTPSTAT** register and the Hibernate clock source is switched to the HIB LFIOSC immediately. When the XOSCST bit in the **HIBTPSTAT** register is 0, indicating the external oscillator is active, a 1 can be written to the XOSCFAIL bit to clear it and re-enable the external 32.768-kHz oscillator.

**Note:** Because the HIB LFIOSC has a wide frequency variation, it should not be configured as the HIB clock source when accurate monitoring of the tamper logs are important.

#### *Tamper Resets*

The Tamper module uses the resets from the Hibernate module.

---

**Important:** The Hibernation module registers are reset under two conditions:

1. Any type of system reset (if the RTCEN and the PINWEN bits in the **HIBCTL** register are clear and the TPEN bit in the **HIBTPCTL** register is clear).
2. A cold POR occurs when both the $V_{DD}$ and $V_{BAT}$ supplies are removed.

Any other reset condition is ignored by the Hibernation module.

---

### 7.3.7 Battery-Backed Memory

The Hibernation module contains 16 32-bit words of memory that are powered from the battery or an auxiliary power supply and therefore retained during hibernation. The processor software can save state information in this memory prior to hibernation and recover the state upon waking. To access the upper eight words of memory, the processor must be in privilege mode. Refer to “Processor Mode and Privilege Levels for Software Execution” on page 94 for more information about processor privilege mode. The battery-backed memory can be accessed through the **HIBDATA** registers. If both $V_{DD}$ and $V_{BAT}$ are removed, the contents of the **HIBDATA** registers are not retained.

### 7.3.8 Power Control Using $\overline{\text{HIB}}$

---

**Important:** The Hibernation Module requires special system implementation considerations when using $\overline{\text{HIB}}$ to control power, as it is intended to power-down all other sections of the microcontroller. All system signals and power supplies that connect to the chip must be driven to 0 V or powered down with the same regulator controlled by $\overline{\text{HIB}}$.

---

The Hibernation module controls power to the microcontroller through the use of the $\overline{\text{HIB}}$ pin which is intended to be connected to the enable signal of the external regulator(s) providing 3.3 V to the microcontroller and other circuits. When the $\overline{\text{HIB}}$ signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the microcontroller and any parts of the system that are powered by the regulator. The Hibernation module remains powered from the $V_{BAT}$ supply until a Wake event. Power to the microcontroller is restored by deasserting the $\overline{\text{HIB}}$ signal, which causes the external regulator to turn power back on to the chip.

### 7.3.9 Power Control Using VDD3ON Mode

The Hibernation module may also be configured to cut power to all internal modules during Hibernate mode. While in this state, if VDD3ON is set in the **HIBCTL** register, all pins are held in the state they were in prior to entering hibernation. For example, inputs remain inputs; outputs driven high remain driven high, and so on. There are important procedural and functional items to note when in VDD3ON mode:

- JTAG Ports C[0] - C[3] do not retain their state in Hibernate VDD3ON mode.
- If GPIO pins K[7:4] are not used as a wake source, they should not be left floating. An internal pull-up resistor may be configured by the application before entering Hibernate mode by programming the **GPIO Pull-Up Select (GPIOPUR)** register in the GPIO module.
- In the VDD3ON mode, the regulator should maintain 3.3 V power to the microcontroller during Hibernate. GPIO retention is disabled when the RETCLR bit is cleared in the **HIBCTL** register.
- When entering hibernation in VDD3ON mode, the supply rails to the Ethernet resistors R1, R2, R3, R4 found in Figure 24-15 on page 1650 must be switched off.

### 7.3.10 Initiating Hibernate

Hibernate mode is initiated when the HIBREQ bit of the **HIBCTL** register is set. If a wake-up condition has not been configured using the PINWEN or RTCWEN bits in the **HIBCTL** register, the hibernation request is ignored. In addition, if the battery voltage is below the threshold voltage defined by the VBATSEL field in the **HIBCTL** register, the hibernation request is ignored.

### 7.3.11 Waking from Hibernate

The Hibernation module can be configured to wake from Hibernate mode if any of the following are enabled:

- External $\overline{\text{WAKE}}$
- External $\overline{\text{RST}}$
- GPIO K[7:4]
- Tamper TMPR[3:0]
- Tamper XOSC failure

The Hibernation module can also be configured to wake from hibernate when the following events occur:

- RTC match wake event
- Low Battery wake event

The external $\overline{\text{WAKE}}$ pin is enabled by setting the PINWEN bit in the **HIBCTL** register. The external $\overline{\text{WAKE}}$ pin can generate an interrupt by programming the EXTWEN bit in the **Hibernation Interrupt Mask (HIBIM)** register.

**Note:** If an external $\overline{\text{WAKE}}$ signal is asserted, the application is responsible for clearing the signal source once the EXTWEN bit has been registered in the **Hibernation Raw Interrupt Status (HIBRIS)** register.

To use the RST pin as a wake source, the WURSTEN bit must be set in the **Hibernate I/O Configuration (HIBIO)** register and the WUUNLK bit must be set in the same register.

To enable any of the assigned GPIO pins as a wake source, the WUUNLK bit must be set in the **HIBIO** register and the wake configuration must be programmed through the **GPIOWAKEPEN** and **GPIOWAKELVL** registers in the GPIO module. Please refer to “General-Purpose Input/Outputs (GPIOs)” on page 772 for more information on programming the GPIOs.

**Note:** The RST pin and GPIO wake sources are cleared by a write to either or both the RSTWK and PADIOWK bits. This clears the source of interrupts for RSTWK, PADIOWK and the **GPIOWAKESTAT** register.

TMPR[3:0] are enabled by setting the appropriate ENn bits the **Tamper IO Control and Status (HIBTPIO)** register. The **HIBTPIO** register overrides the GPIO port configuration registers. By setting the WAKE bit in the **Tamper Control (HIBTPCTL)** register, a tamper event can cause a wake from Hibernate. If a tamper event occurs, the time of the event and the status of the tamper pins are logged in the **Tamper Log (HIBTPLOG)** register.

By setting the RTCWEN bit in the **HIBCTL** register a wake from hibernate can occur when the value of the **HIBRTCC** register matches the value of the **HIBRTCM0** register and the value of the RTCSSC field matches the RTCSSM field in the **HIBRTCSS** register.

To allow a wake from Hibernate on a low battery event, the BATWKEN bit in the **HIBCTL** register must be set. In this configuration, the battery voltage is checked every 512 seconds while in hibernation. If the voltage is below the level specified by the VBATSEL field, the LOWBAT interrupt is set in the **HIBRIS** register.

Upon external wake-up, external reset, tamper event, or RTC match, the Hibernation module delays coming out of hibernation until $V_{DD}$ is above the minimum specified voltage, see Table 32-6 on page 2097.

When the Hibernation module wakes, the microcontroller performs a normal power-on reset. The normal power-on reset does not reset the Hibernation module or Tamper module, but does reset the rest of the microcontroller. Software can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see “Interrupts and Status” on page 580) and by looking for state data in the battery-backed memory (see “Battery-Backed Memory” on page 577).

### 7.3.12 Arbitrary Power Removal

The microcontroller goes into hibernation if $V_{DD}$ is arbitrarily removed when the CLK32EN bit is set and any of the following bits are set:

- TPEN bit in the **HIBTPCTL** register
- PINWEN bit in the **HIBCTL** register
- RTCEN bit in the **HIBCTL** register

The microcontroller wakes from hibernation when power is reapplied.

If the CLK32EN bit is set but the TPEN, PINWEN, and RTCEN bits are all clear, the microcontroller still goes into hibernation if power is removed; however, when $V_{DD}$ is reapplied, the MCU executes a cold POR and the Hibernation module is reset. If the CLK32EN bit is not set and $V_{DD}$ is arbitrarily removed, the part is simply powered off and executes a cold POR when power is reapplied.

If $V_{DD}$ is arbitrarily removed while a Flash memory or **HIBDATA** register write operation is in progress, the write operation must be retried after $V_{DD}$ is reapplied.

### 7.3.13 Interrupts and Status

The Hibernation module can generate interrupts when the following conditions occur:

- Assertion of $\overline{\text{WAKE}}$ pin
- RTC match
- Low battery detected
- Write complete/capable
- Assertion of an external $\overline{\text{RESET}}$ pin
- Assertion of an external wake-enabled GPIO pin (port K[7:4])

All of the interrupts except for the tamper signals are ORed together before being sent to the interrupt controller, so the Hibernate module can only generate a single interrupt request to the controller at any given time. The software interrupt handler can service multiple interrupt events by reading the **Hibernation Masked Interrupt Status (HIBMIS)** register. Software can also read the status of the Hibernation module at any time by reading the **HIBRIS** register which shows all of the pending events. This register can be used after waking from hibernation to see if a wake condition was caused by one of the events above or by a power loss.

The $\overline{\text{WAKE}}$ pin can generate interrupts in Run, Sleep and Deep Sleep Mode. The events that can trigger an interrupt are configured by setting the appropriate bits in the **Hibernation Interrupt Mask (HIBIM)** register. Pending interrupts can be cleared by writing the corresponding bit in the **Hibernation Interrupt Clear (HIBIC)** register.

## 7.4 Initialization and Configuration

The Hibernation module has several different configurations. The following sections show the recommended programming sequence for various scenarios. Because the Hibernation module runs at a low frequency and is asynchronous to the rest of the microcontroller, which is run off the system clock, software must allow a delay of $t_{HIB_REG_ACCESS}$ after writes to registers (see “Register Access Timing” on page 567). The WC interrupt in the **HIBMIS** register can be used to notify the application when the Hibernation modules registers can be accessed.

### 7.4.1 Initialization

The Hibernation module comes out of reset with the system clock enabled to the module, but if the system clock to the module has been disabled, then it must be re-enabled, even if the RTC feature is not used. See page 404.

If a 32.768-kHz crystal is used as the Hibernation module clock source, perform the following steps:

1. Write 0x0000.0010 to the **HIBIM** register to enable the WC interrupt.
2. Write 0x40 to the **HIBCTL** register at offset 0x10 to enable the oscillator input.
3. Wait until the WC interrupt in the **HIBMIS** register has been triggered before performing any other operations with the Hibernation module.

If a 32.768-kHz single-ended oscillator is used as the Hibernation module clock source, then perform the following steps:

1. Write 0x0000.0010 to the **HIBIM** register to enable the WC interrupt.
2. Write 0x0001.0040 to the **HIBCTL** register at offset 0x10 to enable the oscillator input and bypass the on-chip oscillator.
3. Wait until the WC interrupt in the **HIBMIS** register has been triggered before performing any other operations with the Hibernation module.

If the internal low frequency oscillator is used as the Hibernation module clock source, then perform the following steps:

1. Write 0x0000.0010 to the **HIBIM** register to enable the WC interrupt.
2. Write 0x0008.0040 to the **HIBCTL** register at offset 0x10 to enable the internal low frequency oscillator.
3. Wait until the WC interrupt in the **HIBMIS** register has been triggered before performing any other operations with the Hibernation module.

The above steps are only necessary when the entire system is initialized for the first time. If the microcontroller has been in hibernation, then the Hibernation module has already been powered up and the above steps are not necessary. The software can detect that the Hibernation module and clock are already powered by examining the CLK32EN bit of the **HIBCTL** register.

### 7.4.2 RTC Match Functionality (No Hibernation)

Use the following steps to implement the RTC match functionality of the Hibernation module:

1. Write 0x0000.0040 to the **HIBCTL** register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write the required RTC match value to the **HIBRTCM0** register at offset 0x004 and the RTCSSM field in the **HIBRTCSS** register at offset 0x028.
3. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
4. Set the required RTC match interrupt mask in the RTCALT0 in the **HIBIM** register at offset 0x014.
5. Write 0x0000.0041 to the **HIBCTL** register at offset 0x010 to enable the RTC to begin counting.

### 7.4.3 RTC Match/Wake-Up from Hibernation

Use the following steps to implement the RTC match and wake-up functionality of the Hibernation module:

1. Write 0x0000.0040 to the **HIBCTL** register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write the required RTC match value to the **HIBRTCM0** register at offset 0x004 and the RTCSSM field in the **HIBRTCSS** register at offset 0x028.
3. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C. This write causes the 15-bit sub seconds counter to be cleared.

4. Write any data to be retained during hibernation to the **HIBDATA** register at offsets 0x030-0x06F.

5. Set the RTC Match Wake-Up and start the hibernation sequence by writing 0x0000.004B to the **HIBCTL** register at offset 0x010.

### 7.4.4 External Wake-Up from Hibernation

Use the following steps to implement the Hibernation module with the external $\overline{\text{WAKE}}$ pin as the wake-up source for the microcontroller:

1. Write 0x0000.0040 to the **HIBCTL** register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.

2. Write any data to be retained during hibernation to the **HIBDATA** register at offsets 0x030-0x06F.

3. Enable the external wake and start the hibernation sequence by writing 0x0000.0052 to the **HIBCTL** register at offset 0x010.

Use the following steps to program the external $\overline{\text{RESET}}$ pin as the wake source for the microcontroller:

1. Write 0x0000.0040 to the **HIBCTL** register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.

2. Write any data to be retained during hibernation to the **HIBDATA** register at offsets 0x030-0x06F.

3. Enable the external $\overline{\text{RESET}}$ pin as a wake source by writing a 0x0000.0011 to the **HIBIO** register at offset 0x02C.

4. When the IOWRC bit in the **HIBIO** register is read as 1, clear the WUUNLK bit in the **HIBIO** register to lock the current pad configuration so that any other writes to the WURSTEN bit in the **HIBIO** register will be ignored.

5. The hibernation sequence may be initiated by writing 0x4000.0152 to the **HIBCTL** register. Note that when using $\overline{\text{RESET}}$, the user must enable VDD3ON mode and set the RETCLR bit in the **HIBCTL** register.

Use the following steps to program GPIO port K pins K[7:4] as the wake source for the microcontroller:

1. Write 0x0000.0040 to the **HIBCTL** register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.

2. Write any data to be retained during hibernation to the **HIBDATA** register at offsets 0x030-0x06F.

3. Configure the **GPIOWAKEPEN** and **GPIOWAKELVL** registers at offsets 0x540 and 0x544 in the GPIO module. Enable the I/O wake pad configuration by writing 0x0000.0001 to the **HIBIO** register at offset 0x010.

4. When the IOWRC bit in the **HIBIO** register is read as 1, write 0x0000.0000 to the **HIBIO** register to lock the current pad configuration so that any other writes to the **GPIOWAKEPEN** and **GPIOWAKELVL** register will be ignored.

5. Clear any pending interrupts by writing a 1 to the PADIOWK bit in the **HIBIC** register.

6. The hibernation sequence may be initiated by writing 0x4000.0152 to the **HIBCTL** register. Note for Port M external wake, the user must enable VDD3ON mode and set the RETCLR bit in the **HIBCTL** register.

### 7.4.5 RTC or External Wake-Up from Hibernation

1. Write 0x0000.0040 to the **HIBCTL** register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write the required RTC match value to the **HIBRTCM0** register at offset 0x004 and the RTCSSM field in the **HIBRTCSS** register at offset 0x028.
3. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C. This write causes the 15-bit sub seconds counter to be cleared.
4. Write any data to be retained during hibernation to the **HIBDATA** register at offsets 0x030-0x06F.
5. Set the RTC Match/External Wake-Up and start the hibernation sequence by writing 0x0000.005B to the **HIBCTL** register at offset 0x010.

### 7.4.6 Tamper Initialization

Use the following steps to configure the Tamper module to interrupt the processor when a TMPR signal has triggered:

**Note:** Unlike other functions, the Tamper pins do not need to be configured for the GPIO in the **GPIOAFSEL** register. The **Tamper IO Control and Status (HIBTPIO)** register overrides configurations made to the GPIO module.

1. Write 0x0000.0041 to the **HIBCTL** register at offset 0x010 to enable the 32.768-kHz Hibernate oscillator and enable the RTC.
2. Enable the four Tamper I/O to trigger on the a high state on any of the pins by writing 0x0F0F.0F0F to the **HIBTPIO** register at offset 0x410.
3. Write 0x0000.0001 to the **HIBTPCTL** register to enable the tamper.

**Note:** Once tamper is enabled, the following **HIBCTL** register bits are locked and cannot be modified:

- OSCSEL
- OSCDRV
- OSCBYP
- VDD3ON
- CLK32EN
- RTCEN

## 7.5 Register Map

Table 7-3 on page 584 lists the Hibernation registers. All addresses given are relative to the Hibernation Module base address at 0x400F.C000. Note that the system clock to the Hibernation module must be enabled before the registers can be programmed (see page 404). There must be a delay of 3 system clocks after the Hibernation module clock is enabled before any Hibernation module registers

are accessed. In addition, the CLK32EN bit in the **HIBCTL** register must be set before accessing any other Hibernation module register.

**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See "Register Access Timing" on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

**Important:** The Hibernation module registers are reset under two conditions:

1. Any type of system reset (if the RTCEN and the PINWEN bits in the **HIBCTL** register are clear and the TPEN bit in the **HIBTPCTL** register is clear).
2. A cold POR occurs when both the $V_{DD}$ and $V_{BAT}$ supplies are removed.

Any other reset condition is ignored by the Hibernation module.

Note that the following registers are only accessed through privileged mode (see "System Control" on page 231 for more details):

- **HIBTPCTL**
- **HIBPTSTAT**
- **HIBTPIO**
- **HIBTPLOG**
- Upper eight words of memory (**HIBDATA** register 0x50 to 0x6F)

**Table 7-3. Hibernation Module Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | HIBRTCC | RO | 0x0000.0000 | Hibernation RTC Counter | 586 |
| 0x004 | HIBRTCM0 | RW | 0xFFFF.FFFF | Hibernation RTC Match 0 | 587 |
| 0x00C | HIBRTCLD | WO | 0x0000.0000 | Hibernation RTC Load | 588 |
| 0x010 | HIBCTL | RW | 0x8000.2000 | Hibernation Control | 589 |
| 0x014 | HIBIM | RW | 0x0000.0000 | Hibernation Interrupt Mask | 594 |
| 0x018 | HIBRIS | RO | 0x0000.0000 | Hibernation Raw Interrupt Status | 596 |
| 0x01C | HIBMIS | RO | 0x0000.0000 | Hibernation Masked Interrupt Status | 598 |
| 0x020 | HIBIC | RW1C | 0x0000.0000 | Hibernation Interrupt Clear | 600 |
| 0x024 | HIBRTCT | RW | 0x0000.7FFF | Hibernation RTC Trim | 602 |
| 0x028 | HIBRTCSS | RW | 0x0000.0000 | Hibernation RTC Sub Seconds | 603 |

**Table 7-3. Hibernation Module Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x02C | HIBIO | RW | 0x8000.0000 | Hibernation IO Configuration | 604 |
| 0x030-0x06F | HIBDATA | RW | - | Hibernation Data | 606 |
| 0x300 | HIBCALCTL | RW | 0x0000.0000 | Hibernation Calendar Control | 607 |
| 0x310 | HIBCAL0 | RO | 0x0000.0000 | Hibernation Calendar 0 | 608 |
| 0x314 | HIBCAL1 | RO | 0x0000.0000 | Hibernation Calendar 1 | 610 |
| 0x320 | HIBCALLD0 | WO | 0x0000.0000 | Hibernation Calendar Load 0 | 612 |
| 0x324 | HIBCALLD1 | WO | 0x0000.0000 | Hibernation Calendar Load | 614 |
| 0x330 | HIBCALM0 | RW | 0x0000.0000 | Hibernation Calendar Match 0 | 615 |
| 0x334 | HIBCALM1 | RW | 0x0000.0000 | Hibernation Calendar Match 1 | 617 |
| 0x360 | HIBLOCK | RW | 0x0000.0000 | Hibernation Lock | 618 |
| 0x400 | HIBTPCTL | RW | 0x0000.0000 | HIB Tamper Control | 619 |
| 0x404 | HIBTPSTAT | RW1C | 0x0000.0000 | HIB Tamper Status | 621 |
| 0x410 | HIBTPIO | RW | 0x0000.0000 | HIB Tamper I/O Control | 623 |
| 0x4E0 | HIBTPLOG0 | RO | 0x0000.0000 | HIB Tamper Log 0 | 627 |
| 0x4E4 | HIBTPLOG1 | RO | 0x0000.0000 | HIB Tamper Log 1 | 628 |
| 0x4E8 | HIBTPLOG2 | RO | 0x0000.0000 | HIB Tamper Log 2 | 627 |
| 0x4EC | HIBTPLOG3 | RO | 0x0000.0000 | HIB Tamper Log 3 | 628 |
| 0x4F0 | HIBTPLOG4 | RO | 0x0000.0000 | HIB Tamper Log 4 | 627 |
| 0x4F4 | HIBTPLOG5 | RO | 0x0000.0000 | HIB Tamper Log 5 | 628 |
| 0x4F8 | HIBTPLOG6 | RO | 0x0000.0000 | HIB Tamper Log 6 | 627 |
| 0x4FC | HIBTPLOG7 | RO | 0x0000.0000 | HIB Tamper Log 7 | 628 |
| 0xFC0 | HIBPP | RO | 0x0000.0002 | Hibernation Peripheral Properties | 630 |
| 0xFC8 | HIBCC | RW | 0x0000.0000 | Hibernation Clock Control | 631 |

## 7.6 Register Descriptions

The remainder of this section lists and describes the Hibernation module registers, in numerical order by address offset.

## Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000

This register is the current 32-bit value of the RTC counter.

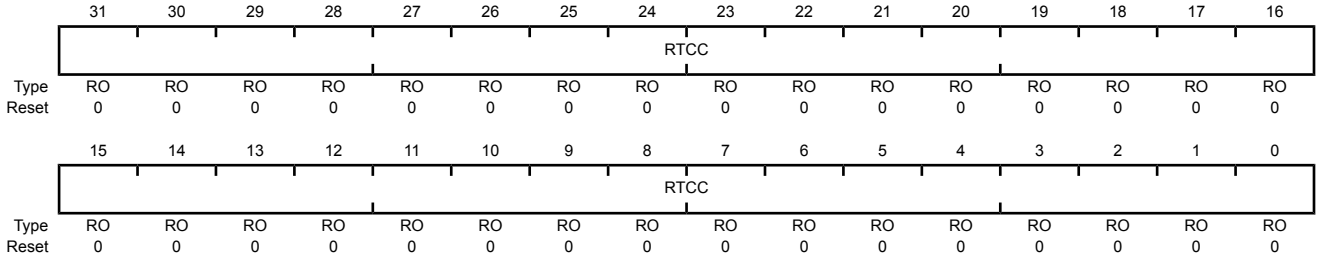
The RTC counter consists of a 32-bit seconds counter and a 15-bit sub seconds counter. The RTC counters are reset by the Hibernation module reset. The RTC 32-bit seconds counter can be set by the user using the **HIBRTCLD** register. When the 32-bit seconds counter is set, the 15-bit sub second counter is cleared.

The RTC value can be read by first reading the **HIBRTCC** register, reading the RTCSSC field in the **HIBRTCSS** register, and then rereading the **HIBRTCC** register. If the two values for **HIBRTCC** are equal, the read is valid.

**Note:** There is a minimum system clock rate of three times the HIB clock rate to properly read the **HIBRTCC** register.

Hibernation RTC Counter (HIBRTCC)

Base 0x400F.C000
Offset 0x000
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTCC | RO | 0x0000.0000 | RTC Counter<br>A read returns the 32-bit counter value, which represents the seconds elapsed since the RTC was enabled. This register is read-only. To change the value, use the **HIBRTCLD** register. |

## Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004

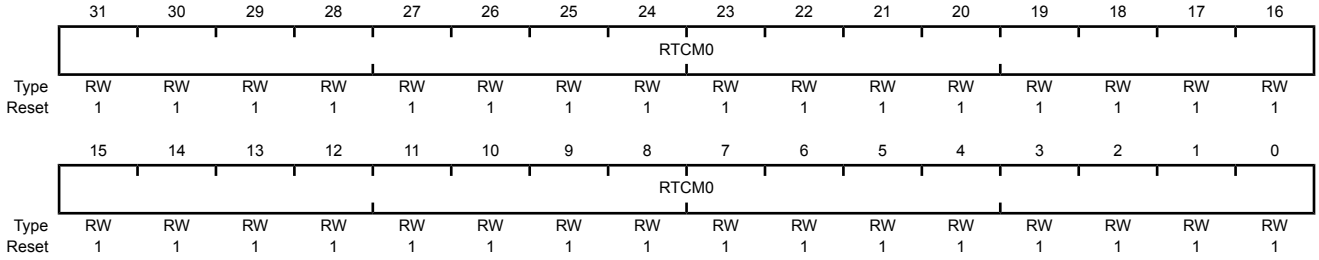
This register is the 32-bit seconds match register for the RTC counter. The 15-bit sub second match value is stored in the reading the RTCSSC field in the **HIBRTCSS** register and can be used in conjunction with this register for a more precise time match.

**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

Hibernation RTC Match 0 (HIBRTCM0)

Base 0x400F.C000
Offset 0x004
Type RW, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTCM0 | RW | 0xFFFF.FFFF | RTC Match 0<br>A write loads the value into the RTC match register.<br>A read returns the current match value. |

## Register 3: Hibernation RTC Load (HIBRTCLD), offset 0x00C

This register is used to load a 32-bit value loaded into the RTC counter. The load occurs immediately upon this register being written. When this register is written, the 15-bit sub seconds counter is also cleared.

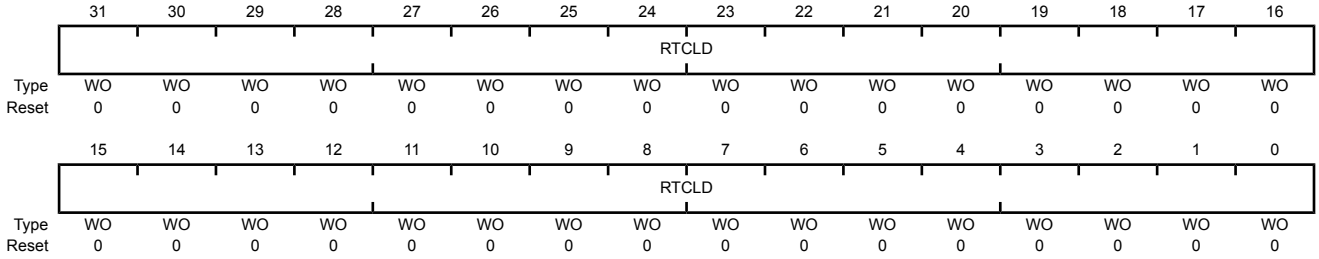
**Note:** This register is protected from errant code by using the **HIBLOCK** register. This register is write-only; any reads to this register read back as zeros.

**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See "Register Access Timing" on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

Hibernation RTC Load (HIBRTCLD)

Base 0x400F.C000
Offset 0x00C
Type WO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTCLD | WO | 0x0000.0000 | RTC Load<br>A write loads the current value into the RTC counter (RTCC).<br>A read returns the 32-bit load value. |

## Register 4: Hibernation Control (HIBCTL), offset 0x010

This register is the control register for the Hibernation module. This register must be written last before a hibernate event is issued. Writes to other registers after the HIBREQ bit is set are not guaranteed to complete before hibernation is entered.

**Note:** Writes to this register have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required synchronization has elapsed. While the WRC bit is clear, any attempts to write this register are ignored. Reads may occur at any time.

Note that once tamper is enabled, the following **HIBCTL** clock configuration bits and bus write stall bit are locked and cannot be modified:

- OSCSEL
- OSCDRV
- OSCBYP
- VDD3ON
- CLK32EN
- RTCEN

Hibernation Control (HIBCTL)

Base 0x400F.C000
Offset 0x010
Type RW, reset 0x8000.2000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WRC | RETCLR | reserved | | | | | | | | | | OSCSEL | reserved | OSCDRV | OSCBYP |
| Type | RO | RW | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RO | RW | RW |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | VBATSEL | | reserved | | BATCHK | BATWKEN | VDD3ON | VABORT | CLK32EN | reserved | PINWEN | RTCWEN | reserved | HIBREQ | RTCEN |
| Type | RO | RW | RW | RO | RO | RW | RW | RW | RW | RW | RO | RW | RW | RO | RW | RW |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | WRC | RO | 1 | Write Complete/Capable |

| Value | Description |
|---|---|
| 0 | The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior. |
| 1 | The interface is ready to accept a write. |

Software must poll this bit between write requests and defer writes until WRC=1 to ensure proper operation. An interrupt can be configured to indicate the WRC has completed.

The bit name WRC means "Write Complete," which is the normal use of the bit (between write accesses). However, because the bit is set out-of-reset, the name can also mean "Write Capable" which simply indicates that the interface may be written to by software. This difference may be exploited by software at reset time to detect which method of programming is appropriate: 0 = software delay loops required; 1 = WRC paced available.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 30 | RETCLR | RW | 0 | GPIO Retention/Clear<br>This bit is used when the VDD3ON bit is set.This bit is must be set when entering the hibernate state when the VDD3ON bit is set. This does not affect behavior when VDD3ON is clear.<br>**Note:** This bit must be set when enabling VDD3ON mode.<br>Value Description<br>0 GPIO retention is released when power is reapplied. The GPIOs are initialized to default values.<br>1 GPIO retention set until software clears this bit. |
| 29:20 | reserved | RO | 0 | 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. |
| 19 | OSCSEL | RW | 0 | Oscillator Select<br>This bit is used to select between the use of an external 32.768-kHz source or the HIB internal low frequency oscillator (HIB LFIOSC).<br>**Note:** To enable the HIB LFIOSC, CLK32EN must be programmed to 1 at the same time the OSCSEL bit is set. Thus the **HIBCTL** register should be written with 0x0008.0040<br>Value Description<br>0 External 32.786-kHZ clock source is enabled.<br>1 HIB Low frequency oscillator (HIB LFIOSC) is enabled.<br>**Note:** The HIB low-frequency oscillator has a wide frequency variation, therefore the RTC is not accurate when using this clock source. |
| 18 | reserved | RO | 0 | 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. |
| 17 | OSCDRV | RW | 0 | Oscillator Drive Capability<br>This bit is used to compensate for larger or smaller filtering capacitors.<br>**Note:** This bit is not meant to be changed once the Hibernation oscillator has started. Oscillator stability is not guaranteed if the user changes this value after the oscillator is running.<br>Value Description<br>0 Low drive strength is enabled, 12 pF.<br>1 High drive strength is enabled, 24 pF. |
| 16 | OSCBYP | RW | 0 | Oscillator Bypass<br>Value Description<br>0 The internal 32.768-kHz Hibernation oscillator is enabled. This bit should be cleared when using an external 32.768-kHz crystal.<br>1 The internal 32.768-kHz Hibernation oscillator is disabled and powered down. This bit should be set when using a single-ended oscillator attached to XOSC0. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | reserved | RO | 0 | 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. |
| 14:13 | VBATSEL | RW | 0x1 | Select for Low-Battery Comparator<br>This field selects the battery level that is used when checking the battery status. If the battery voltage is below the specified level, the LOWBAT interrupt bit in the **HIBRIS** register is set.<br>Value Description<br>0x0 1.9 Volts<br>0x1 2.1 Volts (default)<br>0x2 2.3 Volts<br>0x3 2.5 Volts |
| 12:11 | reserved | RO | 0x0 | 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. |
| 10 | BATCHK | RW | 0 | Check Battery Status<br>Value Description<br>0 When read, indicates that the low-battery comparator cycle is not active.<br>Writing a 0 has no effect.<br>1 When read, indicates the low-battery comparator cycle has not completed.<br>Setting this bit initiates a low-battery comparator cycle. If the battery voltage is below the level specified by VBATSEL field, the LOWBAT interrupt bit in the **HIBRIS** register is set. A hibernation request is held off if a battery check is in progress. |
| 9 | BATWKEN | RW | 0 | Wake on Low Battery<br>Value Description<br>0 The battery voltage level is not automatically checked. Low battery voltage does not cause the microcontroller to wake from hibernation.<br>1 In RTC mode, when this bit is set, the battery voltage level is checked every 512 seconds while in hibernation.<br>In calendar mode, the battery voltage is checked on minutes divisible by 8 while in hibernation.<br>If the voltage is below the level specified by VBATSEL field, the microcontroller wakes from hibernation and the LOWBAT interrupt bit in the **HIBRIS** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | VDD3ON | RW | 0 | VDD Powered<br><br>Value Description<br>0 The internal switches are not used. The HIB signal should be used to control an external switch or regulator.<br>1 The internal switches control the power to the on-chip modules (VDD3ON mode).<br><br>Regardless of the status of the VDD3ON bit, the HIB signal is asserted during Hibernate mode. Thus, when VDD3ON is set, the HIB signal should not be connected to the 3.3V regulator, and the 3.3V power source should remain connected. When this bit is set while in hibernation, all pins are held in the state they were in prior to entering hibernation. For example, inputs remain inputs; outputs driven high remain driven high, and so on.<br>Ports retain their state in VDD3ON mode until the RETCLR bit is cleared. The RETCLR bit must be set when the VDD3ON bit is set. |
| 7 | VABORT | RW | 0 | Power Cut Abort Enable<br><br>Value Description<br>0 The microcontroller goes into hibernation regardless of the voltage level of the battery.<br>1 When this bit is set, the battery voltage level is checked before entering hibernation. If $V_{BAT}$ is less than the voltage specified by VBATSEL, the microcontroller does not go into hibernation. |
| 6 | CLK32EN | RW | 0 | Clocking Enable<br>This bit must be enabled to use the Hibernation module.<br><br>Value Description<br>0 The Hibernation module clock source is disabled.<br>1 The Hibernation module clock source is enabled. |
| 5 | reserved | RO | 0 | 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. |
| 4 | PINWEN | RW | 0 | External Wake and Interrupt Pin Enable<br><br>Value Description<br>0 The status of the WAKE or an external I/O wake pad source pin has no effect on hibernation.<br>1 An assertion of the WAKE pin or an external I/O wake pad source takes the microcontroller out of hibernation. An external I/O wake pad interrupt may be generated in active mode.<br><br>**Note:** The external I/O wake pad interrupt is set if the WAKE pin is asserted in Run, Sleep, or Deep Sleep mode regardless of whether the PINWEN bit is 0x0 or 0x1. The interrupt may be forwarded to the processor by setting the EXTW bit in the **HIBIM** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | RTCWEN | RW | 0 | RTC Wake-up Enable<br><br>Value: 0 — An RTC match event has no effect on hibernation.<br>Value: 1 — An RTC match event (the value the **HIBRTCC** register matches the value of the **HIBRTCM0** register and the value of the RTCSSC field matches the RTCSSM field in the **HIBRTCSS** register) takes the microcontroller out of hibernation. |
| 2 | reserved | RO | 0 | 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. |
| 1 | HIBREQ | RW | 0 | Hibernation Request<br><br>Value: 0 — No hibernation request.<br>Value: 1 — Set this bit to initiate hibernation.<br><br>After a wake-up event, this bit is automatically cleared by hardware.<br>A hibernation request is ignored if both the PINWEN and RTCWEN bits are clear. |
| 0 | RTCEN | RW | 0 | RTC Timer/Calendar Enable<br>This is bit must be set to enable RTC or calendar mode. For calendar mode enable, the CALEN bit in the **HIBCALCTL** register must also be set.<br><br>Value: 0 — The Hibernation module RTC and calendar mode are disabled.<br>Value: 1 — The Hibernation module RTC and calendar mode are enabled.<br><br>**Note:** The low-frequency oscillator has a wide frequency variation, therefore the RTC is not accurate when using this clock source. |

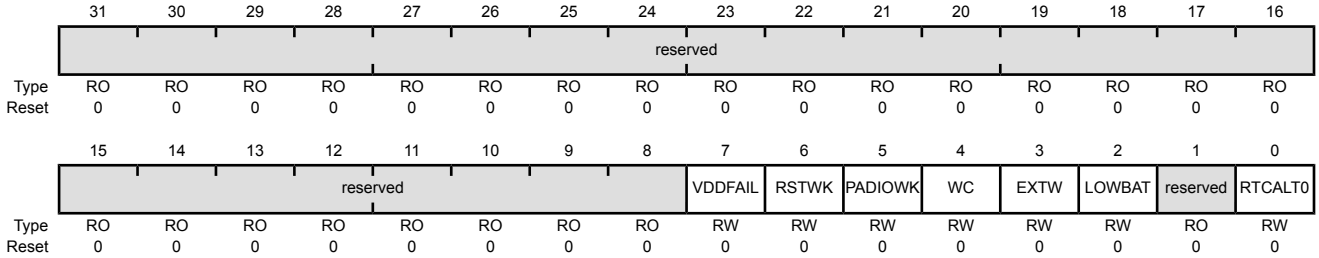
## Register 5: Hibernation Interrupt Mask (HIBIM), offset 0x014

This register is the interrupt mask register for the Hibernation module interrupt sources. Each bit in this register masks the corresponding bit in the **Hibernation Raw Interrupt Status (HIBRIS)** register. If a bit is unmasked, the interrupt is sent to the interrupt controller. If the bit is masked, the interrupt is not sent to the interrupt controller. The WC bit of the **HIBIM** register may be set before the CLK32EN bit of the **HIBCTL** register is set. This allows software to use the WC interrupt trigger to detect when the RTCOSC clock is stable, which may be in excess of one second. If the WC bit is set before the CLK32EN has been set, the mask value is not preserved over a hibernate cycle unless the bit is written a second time.

**Note:** The WC bit of this register is in the system clock domain such that a write to this bit is immediate and may be done before the CLK32EN bit is set in the **HIBCTL** register.

Hibernation Interrupt Mask (HIBIM)

Base 0x400F.C000
Offset 0x014
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.000 | 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. |
| 7 | VDDFAIL | RW | 0 | VDD Fail Interrupt Mask<br>Value Description<br>0 The VDDFAIL interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the VDDFAIL bit in the **HIBRIS** register is set. |
| 6 | RSTWK | RW | 0 | Reset Pad I/O Wake-Up Interrupt Mask<br>Value Description<br>0 The RSTWK interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the RSTWK bit in the **HIBRIS** register is set. |

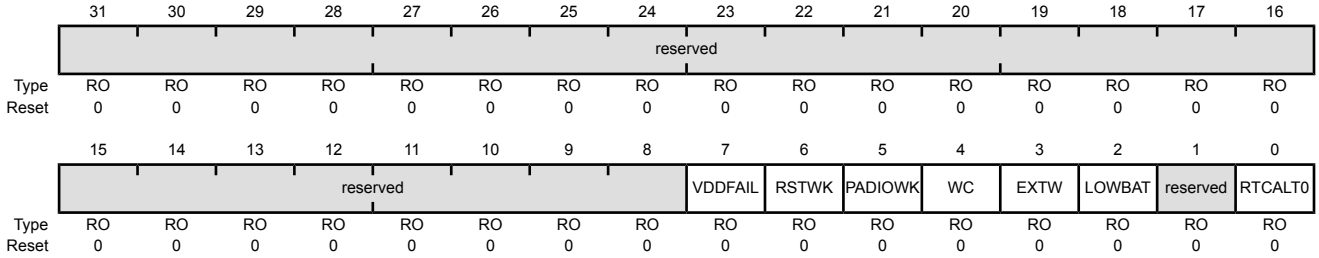
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | PADIOWK | RW | 0 | Pad I/O Wake-Up Interrupt Mask<br>Value Description<br>0 The PADIOWK interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PADIOWK bit in the **HIBRIS** register is set. |
| 4 | WC | RW | 0 | External Write Complete/Capable Interrupt Mask<br>Value Description<br>0 The WC interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the WC bit in the **HIBRIS** register is set. |
| 3 | EXTW | RW | 0 | External Wake-Up Interrupt Mask<br>Value Description<br>0 The EXTW interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the EXTW bit in the **HIBRIS** register is set. |
| 2 | LOWBAT | RW | 0 | Low Battery Voltage Interrupt Mask<br>Value Description<br>0 The LOWBAT interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the LOWBAT bit in the **HIBRIS** register is set. |
| 1 | reserved | RO | 0 | 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. |
| 0 | RTCALT0 | RW | 0 | RTC Alert 0 Interrupt Mask<br>Value Description<br>0 The RTCALT0 interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the RTCALT0 bit in the **HIBRIS** register is set. |

## Register 6: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018

This register is the raw interrupt status for the Hibernation module interrupt sources. Each bit can be masked by clearing the corresponding bit in the **HIBIM** register. When a bit is masked, the interrupt is not sent to the interrupt controller. Bits in this register are cleared by writing a 1 to the corresponding bit in the **Hibernation Interrupt Clear (HIBIC)** register or by entering hibernation.

Hibernation Raw Interrupt Status (HIBRIS)

Base 0x400F.C000
Offset 0x018
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.000 | 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. |
| 7 | VDDFAIL | RO | 0 | VDD Fail Raw Interrupt Status<br>Value Description<br>0 No VDDFAIL interrupt condition exists.<br>1 An interrupt is sent to the interrupt controller because of arbitrary power removal or because one or more of the supplies ($V_{DD}$, $V_{DDA}$ or $V_{DDC}$) has dropped below the defined operating range. |
| 6 | RSTWK | RO | 0 | Reset Pad I/O Wake-Up Raw Interrupt Status<br>Value Description<br>0 The $\overline{\text{RESET}}$ pin has not been asserted or has not been enabled to wake the device from hibernation.<br>1 An interrupt is sent to the interrupt controller because the $\overline{\text{RESET}}$ pin has been programmed to wake the device from hibernation. |
| 5 | PADIOWK | RO | 0 | Pad I/O Wake-Up Raw Interrupt Status<br>Value Description<br>0 One of the wake-enabled GPIO pins or the external $\overline{\text{RESET}}$ pin has not been asserted or has not been enabled to wake the device from hibernation.<br>1 An interrupt is sent to the interrupt controller because one of the wake-enabled GPIO pins or the external $\overline{\text{RESET}}$ pin has been asserted. |

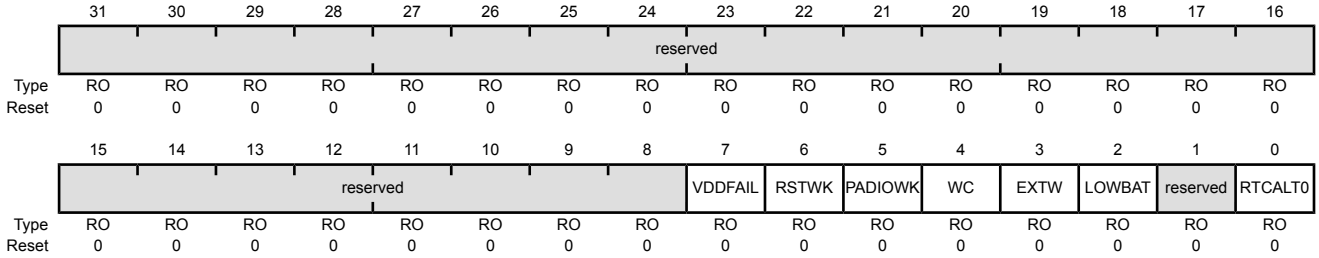
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | WC | RO | 0 | Write Complete/Capable Raw Interrupt Status<br><br>Value Description<br>0 The WRC bit in the **HIBCTL** has not been set.<br>1 The WRC bit in the **HIBCTL** has been set.<br><br>This bit is cleared by writing a 1 to the WC bit in the **HIBIC** register. |
| 3 | EXTW | RO | 0 | External Wake-Up Raw Interrupt Status<br>Note that a wake signal source must be cleared by the application after the interrupt has been registered.<br><br>Value Description<br>0 The $\overline{\text{WAKE}}$ pin has not been asserted.<br>1 The $\overline{\text{WAKE}}$ pin has been asserted.<br><br>This bit is cleared by writing a 1 to the EXTW bit in the **HIBIC** register.<br><br>**Note:** The EXTW bit is set if the $\overline{\text{WAKE}}$ pin is asserted in any mode of operation (Run, Sleep, Deep Sleep) regardless of whether the PINWEN bit is set in the **HIBCTL** register. |
| 2 | LOWBAT | RO | 0 | Low Battery Voltage Raw Interrupt Status<br><br>Value Description<br>0 The battery voltage has not dropped below $V_{LOWBAT}$.<br>1 The battery voltage dropped below $V_{LOWBAT}$.<br><br>This bit is cleared by writing a 1 to the LOWBAT bit in the **HIBIC** register. |
| 1 | reserved | RO | 0 | 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. |
| 0 | RTCALT0 | RO | 0 | RTC Alert 0 Raw Interrupt Status<br><br>Value Description<br>0 No match<br>1 If the RTC is enabled, t he value of the **HIBRTCC** register matches the value in the **HIBRTCM0** register and the value of the RTCSSC field matches the RTCSSM field in the **HIBRTCSS** register.<br>If the Calendar function is enabled, this interrupt status indicates that one or more of the allowed fields in the **HIBCAL0/1** register matches in the **HIBCALM0/1** register..<br><br>This bit is cleared by writing a 1 to the RTCALT0 bit in the **HIBIC** register. |

## Register 7: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C

This register is the masked interrupt status for the Hibernation module interrupt sources. Bits in this register are the AND of the corresponding bits in the **HIBRIS** and **HIBIM** registers. When both corresponding bits are set, the bit in this register is set, and the interrupt is sent to the interrupt controller.

Hibernation Masked Interrupt Status (HIBMIS)

Base 0x400F.C000
Offset 0x01C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.000 | 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. |
| 7 | VDDFAIL | RO | 0 | VDD Fail Interrupt Mask<br>Value Description<br>0 An VDDFAIL interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a an arbitrary loss of power or because on or more of the voltage supplies (VDD, VDDA or VDDC) has dropped below the defined operating range. |
| 6 | RSTWK | RO | 0 | Reset Pad I/O Wake-Up Interrupt Mask<br>Value Description<br>0 An external reset interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a RESET pin assertion. |
| 5 | PADIOWK | RO | 0 | Pad I/O Wake-Up Interrupt Mask<br>Value Description<br>0 An external GPIO or reset interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a wake-enabled GPIO or RESET pin assertion. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | WC | RO | 0 | Write Complete/Capable Masked Interrupt Status<br><br>Value Description<br>0 The WRC bit has not been set or the interrupt is masked.<br>1 An unmasked interrupt was signaled due to the WRC bit being set.<br><br>This bit is cleared by writing a 1 to the WC bit in the **HIBIC** register. |
| 3 | EXTW | RO | 0 | External Wake-Up Masked Interrupt Status<br><br>Value Description<br>0 An external wake-up interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a WAKE pin assertion.<br><br>This bit is cleared by writing a 1 to the EXTW bit in the **HIBIC** register. |
| 2 | LOWBAT | RO | 0 | Low Battery Voltage Masked Interrupt Status<br><br>Value Description<br>0 A low-battery voltage interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a low-battery voltage condition.<br><br>This bit is cleared by writing a 1 to the LOWBAT bit in the **HIBIC** register. |
| 1 | reserved | RO | 0 | 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. |
| 0 | RTCALT0 | RO | 0 | RTC Alert 0 Masked Interrupt Status<br><br>**Note:** The MIS may apply to either the RTC or calendar block depending on which is enabled.<br><br>Value Description<br>0 An RTC or calendar match interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to an RTC or calendar match.<br><br>This bit is cleared by writing a 1 to the RTCALT0 bit in the **HIBIC** register. |

## Register 8: Hibernation Interrupt Clear (HIBIC), offset 0x020

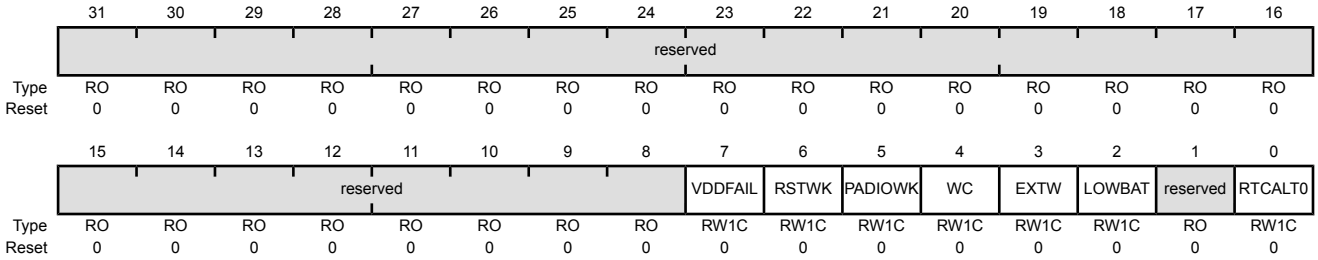
This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources. Writing a 1 to a bit clears the corresponding interrupt in the **HIBRIS** register.

**Note:** Writes to the RSTWK, PADIOWK and WC bits of this register are immediate and the status may be read from the **HIBRIS** and **HIBMIS** registers without monitoring the WRC bit of the **HIBCTL** register.

**Note:** All I/O wake sources are cleared by a write to either or both the RSTWK and PADIOWK bits. This clears the source of interrupts for RSTWK, PADIOWK and the **GPIOWAKESTAT** register.

Hibernation Interrupt Clear (HIBIC)

Base 0x400F.C000
Offset 0x020
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.000 | 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. |
| 7 | VDDFAIL | RW1C | 0 | VDD Fail Interrupt Clear<br>Writing a 1 to this bit clears the VDDFAIL bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return the raw interrupt status. |
| 6 | RSTWK | RW1C | 0 | Reset Pad I/O Wake-Up Interrupt Clear<br>Writing a 1 to this bit clears the RSTWK bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return the raw interrupt status. |
| 5 | PADIOWK | RW1C | 0 | Pad I/O Wake-Up Interrupt Clear<br>Writing a 1 to this bit clears the PADIOWK bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return the raw interrupt status. |
| 4 | WC | RW1C | 0 | Write Complete/Capable Interrupt Clear<br>Writing a 1 to this bit clears the WC bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return the raw interrupt status. |
| 3 | EXTW | RW1C | 0 | External Wake-Up Interrupt Clear<br>Writing a 1 to this bit clears the EXTW bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return the raw interrupt status. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | LOWBAT | RW1C | 0 | Low Battery Voltage Interrupt Clear<br>Writing a 1 to this bit clears the LOWBAT bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return the raw interrupt status. |
| 1 | reserved | RO | 0 | 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. |
| 0 | RTCALT0 | RW1C | 0 | RTC Alert0 Masked Interrupt Clear<br>Writing a 1 to this bit clears the RTCALT0 bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return the raw interrupt status.<br>**Note:** The timer interrupt source cannot be cleared if the RTC value and the **HIBRTCM0** register / RTCMSS field values are equal. The match interrupt takes priority over the interrupt clear. |

## Register 9: Hibernation RTC Trim (HIBRTCT), offset 0x024

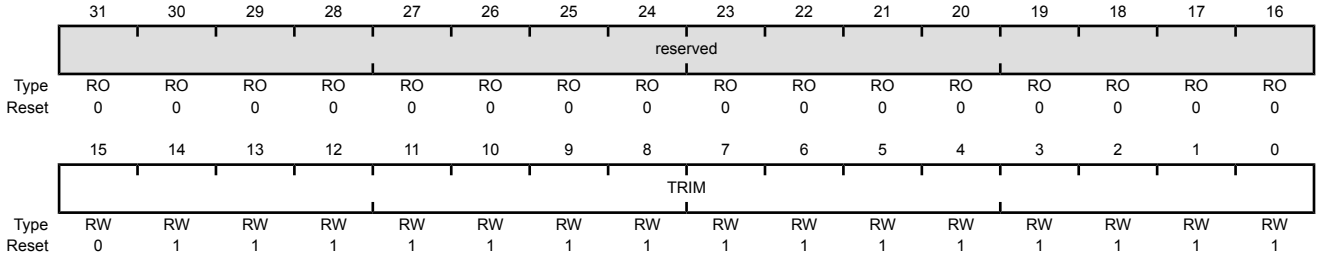
This register contains the value that is used to trim the RTC clock predivider. It represents the computed underflow value that is used during the trim cycle. It is represented as 0x7FFF ± N clock cycles, where N is the number of clock cycles to add or subtract every 64 seconds in RTC mode or 60 seconds in calendar mode.

**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See "Register Access Timing" on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

Hibernation RTC Trim (HIBRTCT)

Base 0x400F.C000
Offset 0x024
Type RW, reset 0x0000.7FFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | TRIM | RW | 0x7FFF | RTC Trim Value<br>This value is loaded into the RTC predivider every 64 seconds in RTC counter mode.<br>In calendar mode, the value is loaded every 60 seconds.<br>It is used to adjust the RTC rate to account for drift and inaccuracy in the clock source. Compensation can be adjusted by software by moving the default value of 0x7FFF up or down. Moving the value up slows down the RTC and moving the value down speeds up the RTC. |

## Register 10: Hibernation RTC Sub Seconds (HIBRTCSS), offset 0x028

This register contains the RTC sub seconds counter and match values. The RTC value can be read by first reading the **HIBRTCC** register, reading the RTCSSC field in the **HIBRTCSS** register, and then rereading the **HIBRTCC** register. If the two values for **HIBRTCC** are equal, the read is valid.

**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

**Note:** There is a minimum system clock rate of three times the HIB clock rate to properly read the **HIBRTCSS** register.

Hibernation RTC Sub Seconds (HIBRTCSS)

Base 0x400F.C000
Offset 0x028
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | RTCSSM | | | | | | | |
| Type RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | RTCSSC | | | | | | | |
| Type RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | reserved | RO | 0 | 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. |
| 30:16 | RTCSSM | RW | 0x0000 | RTC Sub Seconds Match<br>The match value is contained in this field in one RTCOSC clock increments. A read returns the current seconds match value. |
| 15 | reserved | RO | 0 | 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. |
| 14:0 | RTCSSC | RO | 0x0000 | RTC Sub Seconds Count<br>This field contains the sub second RTC count and is read as RTCOSC clock units. For the 32.768-kHz clock source, this would be in units of 1/32,768 seconds. |

## Register 11: Hibernation IO Configuration (HIBIO), offset 0x02C

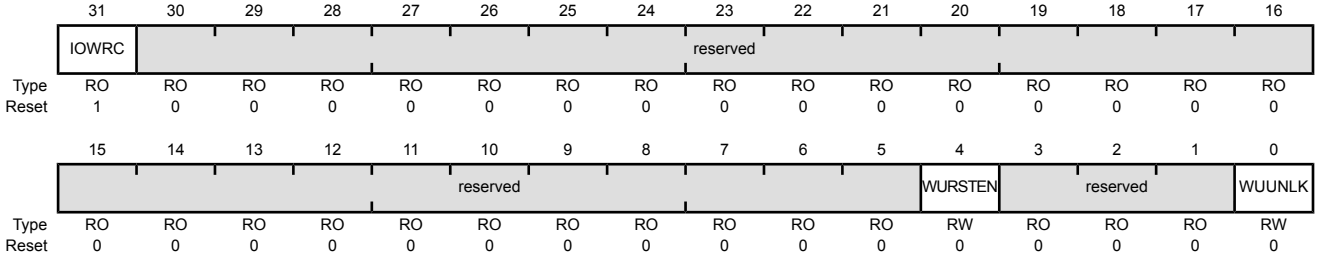
This register is used to lock and unlock the external wake pin levels and enable the external RST pin and/or GPIO pins, Port K[7:4], as valid external WAKE sources.

**Note:** This register is in the system clock domain and does not require monitoring the WRC bit of the **HIBCTL** register before issuing a read or write of this register. Writes to this register are immediate.

**Note:** This register is in the core voltage domain and will not retain values over a hibernate cycle

Hibernation IO Configuration (HIBIO)

Base 0x400F.C000
Offset 0x02C
Type RW, reset 0x8000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | IOWRC | RO | 0x1 | I/O Write Complete<br>Indicates whether or not the configuration that was programmed by the WURSTEN bit or **GPIOWAKEPEN** and **GPIOWAKELVL** registers have propagated through the pad ring.<br>Value Description<br>0 The changes programmed in the external pad I/O wake source registers have not propagated through the pad I/O.<br>1 The changes programmed in the external pad I/O wake source registers have propagated through the pad I/O. |
| 30:5 | reserved | RO | 0x0000.000 | 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. |
| 4 | WURSTEN | RW | 0 | Reset Wake Source Enable<br>This register bit programming takes affect after WUUNLK has been set.<br>Value Description<br>0 The $\overline{\text{RST}}$ signal is not enabled as a wake source.<br>1 The $\overline{\text{RST}}$ signal is enabled as a wake source. |
| 3:1 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | WUUNLK | RW | 0 | I/O Wake Pad Configuration Enable |

| Value | Description |
|---|---|
| 0 | The I/O WAKE configuration set by the WURSTEN bit or in the GPIO module registers **GPIOWAKEPEN** and **GPIOWAKELVL** is ignored. |
| 1 | Implement the I/O WAKE configuration, level and enables for the external RST pin and/or GPIO wake-enabled pins. |

**Note:** This bit must be cleared before issuing a hibernate request by setting the HIBREQ bit in the **HIBCTL** register.

## Register 12: Hibernation Data (HIBDATA), offset 0x030-0x06F

This address space is implemented as a 16x32-bit memory (64 bytes). It can be loaded by the system processor in order to store state information and retains its state during a power cut operation as long as a battery is present. **HIBDATA** registers 0x050 to 0x064 (upper eight words) may only be accessed using the processor privileged mode (default).

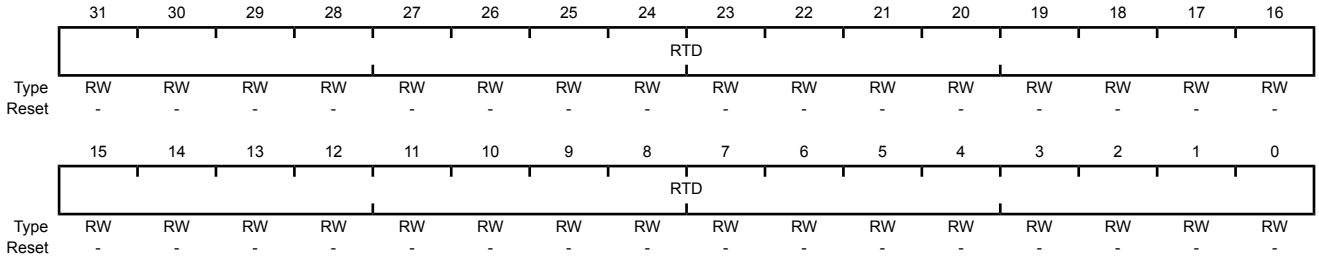
**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

**Note:** If $V_{DD}$ is arbitrarily removed while a **HIBDATA** register write operation is in progress, the write operation must be retried after $V_{DD}$ is reapplied.

Hibernation Data (HIBDATA)

Base 0x400F.C000
Offset 0x030-0x06F
Type RW, reset -



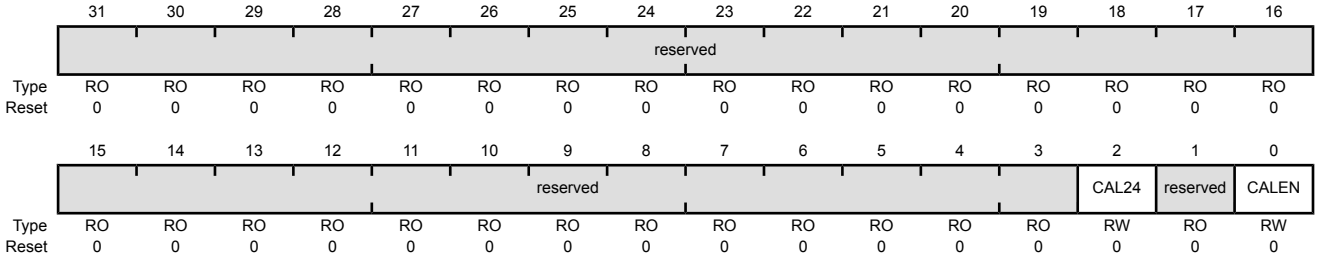

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTD | RW | - | Hibernation Module NV Data |

## Register 13: Hibernation Calendar Control (HIBCALCTL), offset 0x300

The Hibernate calendar is enabled by setting the CALEN bit in the **HIBCALCTL** register. If the BCD bit is set, the fields are reported in BCD format.

Hibernation Calendar Control (HIBCALCTL)

Base 0x400F.C000
Offset 0x300
Type RW, reset 0x0000.0000



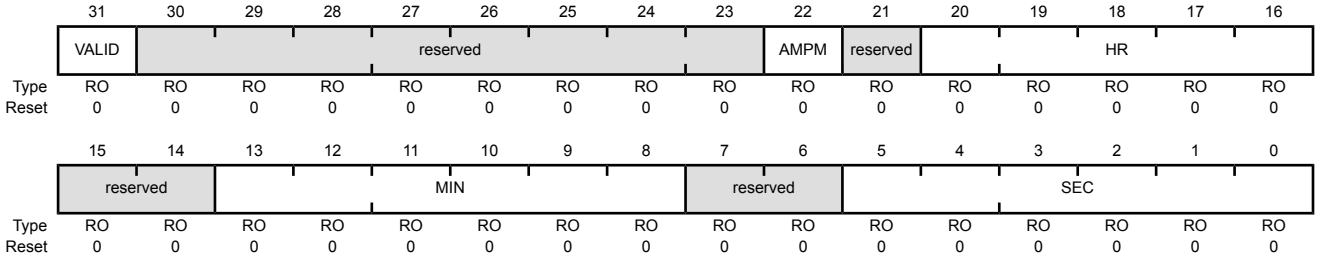

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | CAL24 | RW | 0 | Calendar Mode<br>Value Description<br>0 12 hour, AM/PM Mode<br>1 24 hour mode |
| 1 | reserved | RO | 0 | 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. |
| 0 | CALEN | RW | 0 | RTC Calendar/Counter Mode Select<br>Note that the RTC must be enabled by setting the RTCEN bit in the **HIBCTL** register to use this mode select.<br>Value Description<br>0 RTC Counter mode enabled.<br>1 Calendar mode enabled |

## Register 14: Hibernation Calendar 0 (HIBCAL0), offset 0x310

The **Hibernation Calendar 0 (HIBCAL0)** register is used when the CALEN bit is set in the **HIBCALCTL** register.

Hibernation Calendar 0 (HIBCAL0)

Base 0x400F.C000
Offset 0x310
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | VALID | RO | 0 | Valid Calendar Load<br>The calendar may take several cycles to update as the values roll over. This bit indicates whether the **HIBCAL0** register contents are valid.<br>Value Description<br>0 Register currently updating or initializing<br>1 **HIBCAL0** register valid and ready. |
| 30:23 | reserved | RO | 0 | 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. |
| 22 | AMPM | RO | 0 | AM/PM Designation<br>This bit is used when CAL24=0 in the **HIBCALCTL** register.<br>Value Description<br>0 AM<br>1 PM |
| 21 | reserved | RO | 0 | 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. |
| 20:16 | HR | RO | 0 | Hours<br>This field holds the hour information in hexadecimal.<br>For military time, bits 20:16 range from 0x0 to 0x17 (0 to 23 hours).<br>For standard time (AM/PM mode) bits 20:16 range from 0x0 to 0x11, with 0x0 representing 12AM or 12 PM. |
| 15:14 | reserved | RO | 0 | 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. |

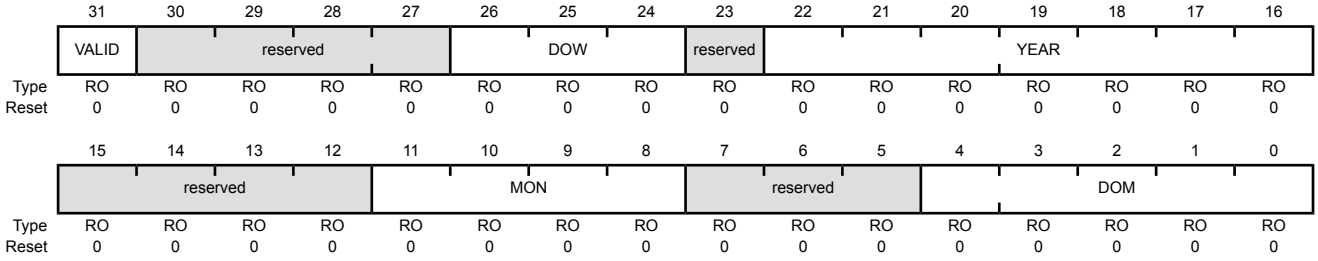
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13:8 | MIN | RO | 0 | Minutes<br>This field holds the minute information in hexadecimal. Bits 13:8 correspond to hex values from 0x0 to 0x3b (0 to 59 minutes). |
| 7:6 | reserved | RO | 0 | Software should not rely on the value 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. |
| 5:0 | SEC | RO | 0 | Seconds<br>This field holds the seconds value in hexadecimal. Bits 5:0 correspond to hex values from 0x0 to 0x3b (0 to 59 seconds). |

## Register 15: Hibernation Calendar 1 (HIBCAL1), offset 0x314

The **Hibernation Calendar 1 (HIBCAL1)** register is used when the CALEN bit is set in the **HIBCALCTL** register.

Hibernation Calendar 1 (HIBCAL1)

Base 0x400F.C000
Offset 0x314
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | VALID | RO | 0 | Valid Calendar Load<br>The calendar may take several cycles to update as the values roll over. This bit indicates whether the **HIBCAL1** register contents are valid.<br>Value Description<br>0 Register currently updating or initializing<br>1 **HIBCAL1** register valid and ready. |
| 30:27 | reserved | RO | 0 | 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. |
| 26:24 | DOW | RO | 0 | Day of Week<br>This field displays the day of the week in the encodings 0x0 to 0x6. The application defines which days are assigned to each encoding. |
| 23 | reserved | RO | 0 | 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. |
| 22:16 | YEAR | RO | 0 | Year Value<br>The last two digits of the year are stored in hexadecimal in this field.<br>Bits 22:16 correspond to hex values from 0x0 to 0x63 (0 to 99 years). |
| 15:12 | reserved | RO | 0 | 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. |
| 11:8 | MON | RO | 0 | Month<br>This field holds the month value in hexadecimal.<br>Bits 11:8 correspond to hex values from 0x1 to 0xC (1 to 12 months). |
| 7:5 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4:0 | DOM | RO | 0 | Day of Month<br>This field holds the day of the month value in hexadecimal.<br>Bits 4:0 correspond to hex values from 0x1 to 1F (1 to 31 days). The value 0 is used to show an ignore match. |

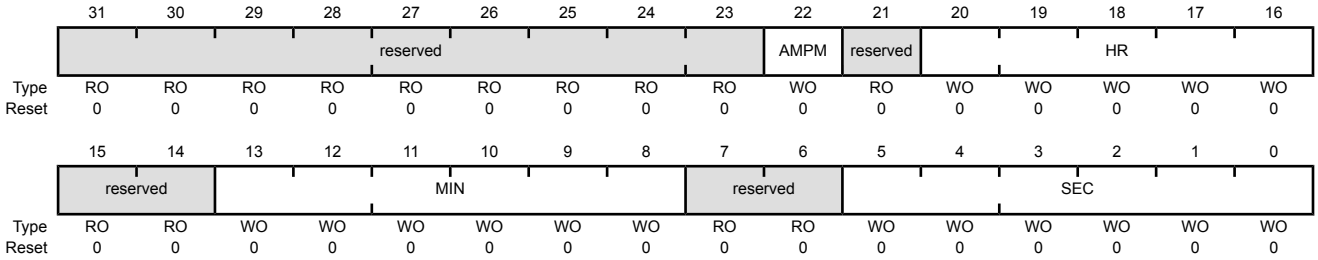
## Register 16: Hibernation Calendar Load 0 (HIBCALLD0), offset 0x320

The **Hibernation Calendar Load (HIBCALLD0)** register is used when the CALEN bit is set in the **HIBCALCTL** register.

**Note:** This register is write-only; any reads to this register read back as zeros. Errant writes to the **HIBCALLD0/1** registers are protected by the Hibernate **HIBLOCK** register.

Hibernation Calendar Load 0 (HIBCALLD0)

Base 0x400F.C000
Offset 0x320
Type WO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:23 | reserved | RO | 0 | 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. |
| 22 | AMPM | WO | 0 | AM/PM Designation<br>This bit is used when CAL24=0 in the **HIBCALCTL** register.<br><br>Value Description<br>0 AM<br>1 PM |
| 21 | reserved | RO | 0 | 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. |
| 20:16 | HR | WO | 0 | Hours<br>This field holds the hour information in hexadecimal.<br>Bits 20:16 correspond to hex values from 0x0 to 0x17 (0 to 23 hours). |
| 15:14 | reserved | RO | 0 | 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. |
| 13:8 | MIN | WO | 0 | Minutes<br>This field holds the minute information in hexadecimal.<br>Bits 13:8 correspond to hex values from 0x0 to 0x3B (0 to 59 minutes). |
| 7:6 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:0 | SEC | WO | 0 | Seconds<br>This field holds the seconds value in hexadecimal.<br>Bits 5:0 correspond to hex values from 0x0 to 0x3B (0 to 59 seconds). |

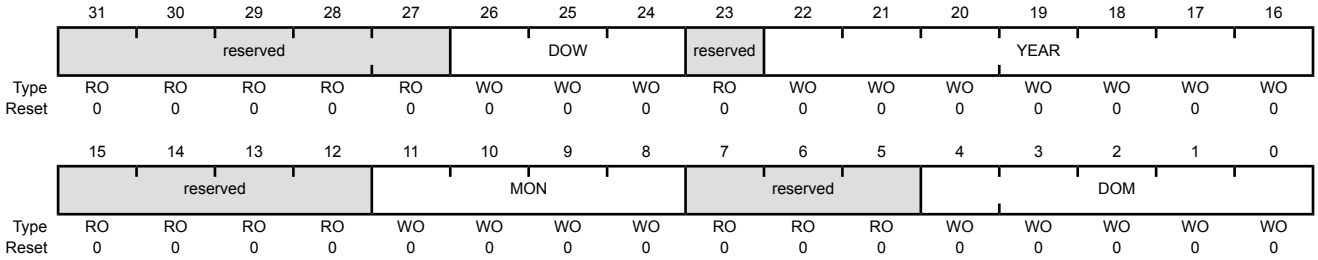
## Register 17: Hibernation Calendar Load (HIBCALLD1), offset 0x324

The **Hibernation Calendar Load 1 (HIBCALLD1)** register is used when the CALEN bit is set in the **HIBCALCTL** register.

**Note:** This register is write-only; any reads to this register read back as zeros. Errant writes to the **HIBCALLD0/1** registers are protected by the Hibernate **HIBLOCK** register.

Hibernation Calendar Load (HIBCALLD1)

Base 0x400F.C000
Offset 0x324
Type WO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:27 | reserved | RO | 0 | 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. |
| 26:24 | DOW | WO | 0 | Day of Week<br>This field is written with the day of the week in the encodings 0x0 to 0x6. The application defines which days are assigned to each encoding. |
| 23 | reserved | RO | 0 | 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. |
| 22:16 | YEAR | WO | 0 | Year Value<br>The last two digits of the year are written in this field in hexadecimal. For example, "12" would be programmed into this field for 2012.<br>Bits 22:16 correspond to hex values from 0x0 to 0x63 (0 to 99 years). |
| 15:12 | reserved | RO | 0 | 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. |
| 11:8 | MON | WO | 0 | Month<br>The month value is written in this field in hexadecimal.<br>Bits 11:8 correspond to hex values from 0x1 to 0xC (1 to 12 months). |
| 7:5 | reserved | RO | 0 | 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. |
| 4:0 | DOM | WO | 0 | Day of Month<br>The day of the month value is written in this field in hexadecimal.<br>Bits 4:0 correspond to hex values from 0x1 to 1F (1 to 31 days). The encoding 0x0 is reserved for the ignore match function. |

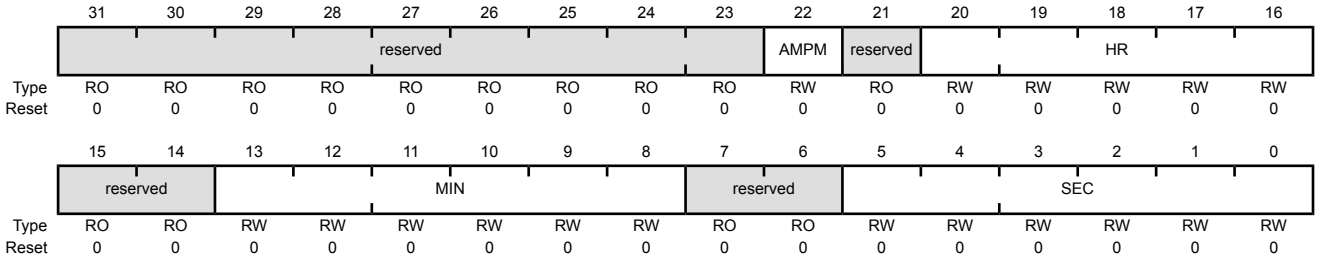
## Register 18: Hibernation Calendar Match 0 (HIBCALM0), offset 0x330

The **Hibernation Calendar Match 0 (HIBCALM0)** register is used when the CALEN bit is set in the **HIBCALCTL** register. This register is loaded with desired match values for calendar mode. Once the **HIBCAL0/1** register values equal the **HIBCALM0/1** register values, the RTCALT0 bit is set in the **HIBRIS** register.

**Note:** The day of week, month and year are not included in the match functionality.

Hibernation Calendar Match 0 (HIBCALM0)

Base 0x400F.C000
Offset 0x330
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:23 | reserved | RO | 0 | 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. |
| 22 | AMPM | RW | 0 | AM/PM Designation<br>This bit is used when CAL24=0 in the **HIBCALCTL** register.<br>Value Description<br>0 AM<br>1 PM |
| 21 | reserved | RO | 0 | 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. |
| 20:16 | HR | RW | 0 | Hours<br>This field match value for the hours in hexadecimal units.<br>Bits 20:16 correspond to hex values from 0x0 to 0x17 (0 to 23 hours). To ignore the hours match, write this field to all 1s. |
| 15:14 | reserved | RO | 0 | 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. |
| 13:8 | MIN | RW | 0 | Minutes<br>This field holds the match value for minutes in hexadecimal units.<br>Bits 13:8 correspond to hex values from 0x0 to 0x3B (0 to 59 minutes). To ignore the hours match, write this field to all 1s. |
| 7:6 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:0 | SEC | RW | 0 | Seconds<br>This field holds the match value for seconds. The value is represented in hexadecimal.<br>Bits 5:0 correspond to hex values from 0x0 to 0x3b (0 to 59 seconds). To ignore the hours match, write this field to all 1s. |

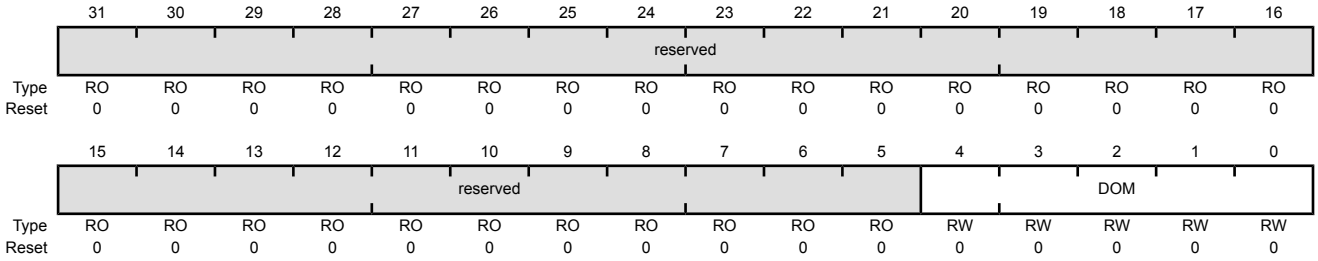
## Register 19: Hibernation Calendar Match 1 (HIBCALM1), offset 0x334

The **Hibernation Calendar Match 1 (HIBCALM1)** register is used when the CALEN bit is set in the **HIBCALCTL** register. This register is loaded with desired match values for calendar mode. Once the **HIBCAL0/1** register values equal the **HIBCALM0/1** register values, the RTCALT0 bit is set in the **HIBRIS** register.

**Note:** The day of week, month and year are not included in the match functionality.

Hibernation Calendar Match 1 (HIBCALM1)

Base 0x400F.C000
Offset 0x334
Type RW, reset 0x0000.0000



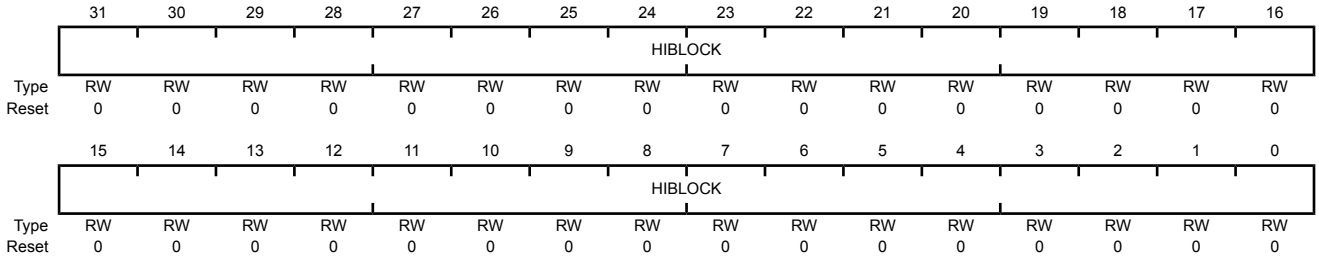

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4:0 | DOM | RW | 0 | Day of Month<br>This field holds the match value for the day of the month in hexadecimal.<br>Bits 4:0 correspond to hex values from 0x1 to 1F (1 to 31 days). To disable match for the day of the month, the value 0x0 is used. |

## Register 20: Hibernation Lock (HIBLOCK), offset 0x360

Writing 0xA335.9554 to the **HIBLOCK** register enables write access to the **HIBRTCLD**, **HIBCALLD0**, **HIBCALLD1** and Tamper registers. Writing any other value to the **HIBLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **HIBLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **HIBLOCK** register returns 0x0000.0001 when locked; otherwise, the returned value is 0x0000.0000 (unlocked).

Hibernation Lock (HIBLOCK)

Base 0x400F.C000
Offset 0x360
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | HIBLOCK | RW | 0x0000 | HIbernate Lock<br>A write of 0xA335.9554 unlocks the **HIBRCTL** and Tamper registers. |

## Register 21: HIB Tamper Control (HIBTPCTL), offset 0x400

The **Tamper Control (HIBTPCTL)** register provides control of the module.

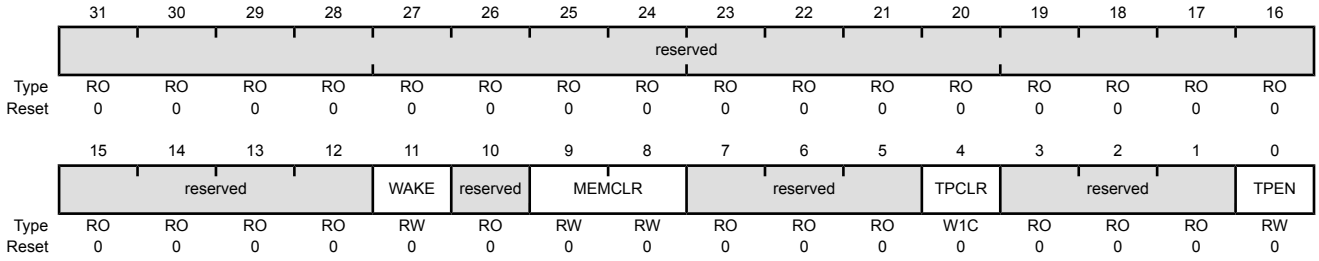
**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See "Register Access Timing" on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

**Note:** Errant writes to the Tamper registers are protected by the Hibernate **HIBLOCK** register.

HIB Tamper Control (HIBTPCTL)

Base 0x400F.C000
Offset 0x400
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0 | 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. |
| 11 | WAKE | RW | 0 | Wake from Hibernate on a Tamper Event<br>Value Description<br>0 Do not wake from hibernate on a tamper event.<br>1 Wake from hibernate on a tamper event. |
| 10 | reserved | RO | 0 | 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. |
| 9:8 | MEMCLR | RW | 0 | HIB Memory Clear on Tamper Event<br>Value Description<br>0x0 Do not Clear HIB memory on tamper event.<br>0x1 Clear Lower 32 Bytes of HIB memory on tamper event<br>0x2 Clear upper 32 Bytes of HIB memory on tamper event<br>0x3 Clear all HIB memory on tamper event |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:5 | reserved | RO | 0 | 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. |
| 4 | TPCLR | W1C | 0 | Tamper Event Clear<br>Writing a 1 to this bit clears the tamper event. The status of the clear is reflected in the STATE bit field. |
| 3:1 | reserved | RO | 0 | 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. |
| 0 | TPEN | RW | 0 | Tamper Module Enable<br>This bit enables the Tamper module.<br>Value Description<br>0 Tamper module disabled.<br>1 Tamper module Enabled.<br>**Note:** Once tamper is enabled, the following **HIBCTL** register bits are locked and cannot be modified:<br>■ OSCSEL<br>■ OSCDRV<br>■ OSCBYP<br>■ VDD3ON<br>■ CLK32EN<br>■ RTCEN |

## Register 22: HIB Tamper Status (HIBTPSTAT), offset 0x404

The **HIB Tamper Status (HIBTPCTL)** register provides status of the module.

**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See "Register Access Timing" on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

**Note:** Errant writes to the Tamper registers are protected by the Hibernate **HIBLOCK** register.

HIB Tamper Status (HIBTPSTAT)

Base 0x400F.C000
Offset 0x404
Type RW1C, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | STATE | | XOSCST | XOSCFAIL |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW1C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3:2 | STATE | RO | 0 | Tamper Module Status<br>Tamper is defined as being configured when the tamper I/Os have been enabled (setting the ENx bits in the **HIBTPIO** register).<br>Value Description<br>0x0 Tamper disabled.<br>0x1 Tamper configured.<br>0x2 Tamper pin event occurred. |
| 1 | XOSCST | RO | 0 | External Oscillator Status<br>Value Description<br>0 Active<br>1 Stopped |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | XOSCFAIL | RW1C | 0 | External Oscillator Failure<br>Write a 1 to this bit to clear it.<br><br>Value Description<br>0 External oscillator is valid.<br>1 External oscillator has failed |

## Register 23: HIB Tamper I/O Control (HIBTPIO), offset 0x410

The **HIB Tamper I/O Control (HIBTPIO)** register provides control of the Tamper I/O.

**Note:** Except for the **HIBIO** and a portion of the **HIBIC** register, all other Hibernation module registers are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See "Register Access Timing" on page 567. The **HIBIO** register and bits RSTWK, PADIOWK and WC of the **HIBIC** register do not require waiting for write to complete. Because these registers are clocked by the system clock, writes to these registers/bits are immediate.

Writing to registers other than the **HIBCTL** and **HIBIM** before the CLK32EN bit in the **HIBCTL** register has been set may produce unexpected results.

**Note:** Errant writes to the Tamper registers are protected by the Hibernate **HIBLOCK** register.

HIB Tamper I/O Control (HIBTPIO)

Base 0x400F.C000
Offset 0x410
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | GFLTR3 | PUEN3 | LEV3 | EN3 | reserved | | | | GFLTR2 | PUEN2 | LEV2 | EN2 |
| Type | RO | RO | RO | RO | RW | RW | RW | RW | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | GFLTR1 | PUEN1 | LEV1 | EN1 | reserved | | | | GFLTR0 | PUEN0 | LEV0 | EN0 |
| Type | RO | RO | RO | RO | RW | RW | RW | RW | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | reserved | RO | 0 | 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. |
| 27 | GFLTR3 | RW | 0 | TMPR3 Glitch Filtering<br>Value Description<br>0 A trigger match level is ignored until the TMPR3 signal is stable for two hibernate clocks.<br>1 A trigger match level is ignored until the TMPR3 signal is stable for 3,071 Hibernate Clocks (93.7ms using 32.768 kHz). |
| 26 | PUEN3 | RW | 0 | TMPR3 Internal Weak Pull-up Enable<br>Value Description<br>0 Pull-up disabled<br>1 Pull-up enabled |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25 | LEV3 | RW | 0 | TMPR3 Trigger Level<br>Value Description<br>0 Trigger on level low<br>1 Trigger on level high |
| 24 | EN3 | RW | 0 | TMPR3 Enable<br>Value Description<br>0 Detect disabled<br>1 Detect enabled |
| 23:20 | reserved | RO | 0 | 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. |
| 19 | GFLTR2 | RW | 0 | TMPR2 Glitch Filtering<br>Value Description<br>0 A trigger match level is ignored until the TMPR2 signal is stable for two hibernate clocks.<br>1 A trigger match level is ignored until the TMPR2 signal is stable for 3,071 Hibernate Clocks (93.7ms using 32.768 kHz). |
| 18 | PUEN2 | RW | 0 | TMPR2 Internal Weak Pull-up Enable<br>Value Description<br>0 Pull-up disabled<br>1 Pull-up enabled |
| 17 | LEV2 | RW | 0 | TMPR2 Trigger Level<br>Value Description<br>0 Trigger on level low<br>1 Trigger on level high |
| 16 | EN2 | RW | 0 | TMPR2 Enable<br>Value Description<br>0 Detect disabled<br>1 Detect enabled |
| 15:12 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | GFLTR1 | RW | 0 | TMPR1 Glitch Filtering<br>Value Description<br>0 A trigger match level is ignored until the TMPR1 signal is stable for two hibernate clocks.<br>1 A trigger match level is ignored until the TMPR1 signal is stable for 3,071 Hibernate Clocks (93.7ms using 32.768 kHz). |
| 10 | PUEN1 | RW | 0 | TMPR1 Internal Weak Pull-up Enable<br>Value Description<br>0 Pull-up disabled<br>1 Pull-up enabled |
| 9 | LEV1 | RW | 0 | TMPR1 Trigger Level<br>Value Description<br>0 Trigger on level low<br>1 Trigger on level high |
| 8 | EN1 | RW | 0 | TMPR1Enable<br>Value Description<br>0 Detect disabled<br>1 Detect enabled |
| 7:4 | reserved | RO | 0 | 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. |
| 3 | GFLTR0 | RW | 0 | TMPR0 Glitch Filtering<br>Value Description<br>0 A trigger match level is ignored until the TMPR0 signal is stable for two hibernate clocks.<br>1 A trigger match level is ignored until the TMPR0 signal is stable for 3,071 Hibernate Clocks (93.7ms using 32.768 kHz). |
| 2 | PUEN0 | RW | 0 | TMPR0 Internal Weak Pull-up Enable<br>Value Description<br>0 Pull-up disabled<br>1 Pull-up enabled |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | LEV0 | RW | 0 | TMPR0 Trigger Level<br><br>Value Description<br>0 Trigger on level low<br>1 Trigger on level high |
| 0 | EN0 | RW | 0 | TMPR0 Enable<br><br>Value Description<br>0 Detect disabled<br>1 Detect enabled |

### Register 24: HIB Tamper Log 0 (HIBTPLOG0), offset 0x4E0

### Register 25: HIB Tamper Log 2 (HIBTPLOG2), offset 0x4E8

### Register 26: HIB Tamper Log 4 (HIBTPLOG4), offset 0x4F0

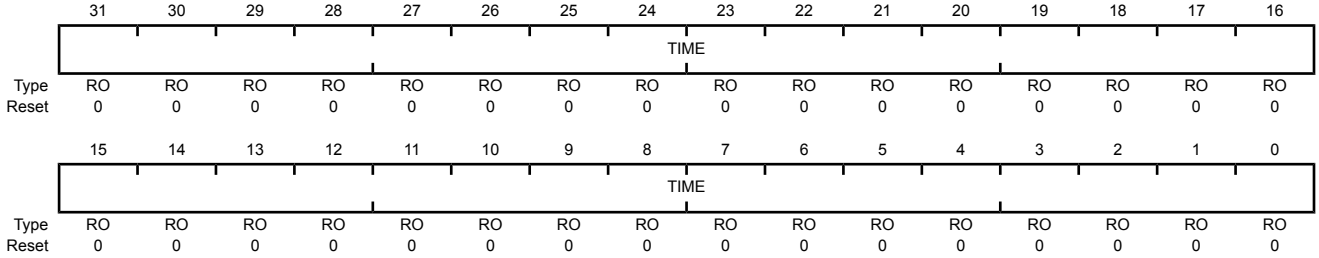
### Register 27: HIB Tamper Log 6 (HIBTPLOG6), offset 0x4F8

The **HIB Tamper Log (HIBTPLOG)** even registers capture the time information during a tamper event. Up to four tamper logs can be stored. The **HIBTPLOG** registers are cleared when the TPCLR bit is written in the **HIBTPCTL** register.

**Note:** It is recommended that an external oscillator is used if accurate time stamps on the tamper log are critical.

HIB Tamper Log 0 (HIBTPLOG0)

Base 0x400F.C000
Offset 0x4E0
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TIME | RO | 0x0 | Tamper Log Calendar Information.<br>When the hibernate module is configured for RTC count mode, the time from the **RTCCC** register is captured on a tamper event.<br>If the calendar function is enabled, the captured time is configured as the hex values for year, month, day, hour, minute and seconds. 24 hour mode should be used by setting the CAL24 bit in **HIBCALCTL** register. The format of the calendar information is as follows:<br>■ TIME[31:26]: Year (0-64)<br>■ TIME[25:22]: Month<br>■ TIME[21:17]: Day of month<br>■ TIME[16:12]: Hours<br>■ TIME[11:6]: Minutes<br>■ TIME[5:0]: Seconds |

## Register 28: HIB Tamper Log 1 (HIBTPLOG1), offset 0x4E4

## Register 29: HIB Tamper Log 3 (HIBTPLOG3), offset 0x4EC

## Register 30: HIB Tamper Log 5 (HIBTPLOG5), offset 0x4F4

## Register 31: HIB Tamper Log 7 (HIBTPLOG7), offset 0x4FC

The **HIB Tamper Log (HIBTPLOGn)** odd registers capture the trigger information during a tamper event. Up to four tamper logs can be stored. The **HIBTPLOG** registers are cleared when the TPCLR bit is set to 1 in the **HIBTPCTL** register. The **HIBTPLOG7** register contains to OR of all events after the 3rd event is logged in **HIBTPLOG5**. The **HIBTPLOG7** register is cleared on a Hibernation module reset.

HIB Tamper Log 1 (HIBTPLOG1)

Base 0x400F.C000
Offset 0x4E4
Type RO, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | XOSC |
| Type RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | TRIG3 | TRIG2 | TRIG1 | TRIG0 |
| Type RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0 | 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. |
| 16 | XOSC | RO | 0 | Status of external 32.768-kHz oscillator<br>Value Description<br>0 Default<br>1 32.768-kHz oscillator has failed |
| 15:4 | reserved | RO | 0 | 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. |
| 3 | TRIG3 | RO | 0 | Status of TMPR[3] Trigger<br>Value Description<br>0 Default<br>1 A tamper event has been detected on TMPR[3] |
| 2 | TRIG2 | RO | 0 | Status of TMPR[2] Trigger<br>Value Description<br>0 Default<br>1 A tamper event has been detected on TMPR[2] |

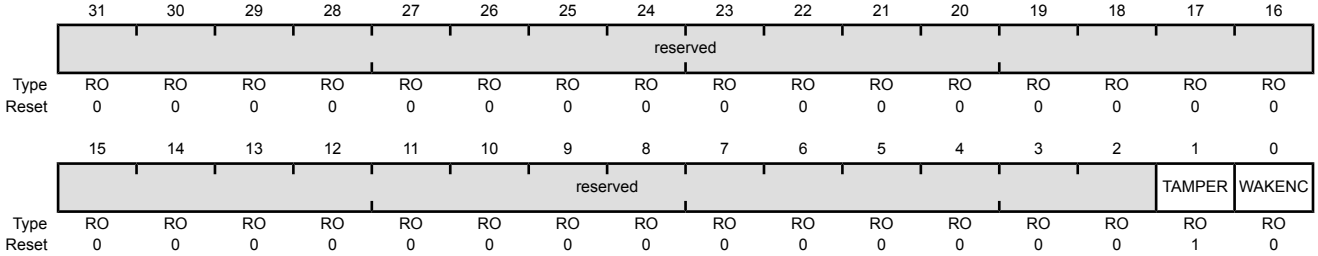
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | TRIG1 | RO | 0 | Status of TMPR[1] Trigger<br><br>Value Description<br>0 Default<br>1 A tamper event has been detected on TMPR[1] |
| 0 | TRIG0 | RO | 0 | Status of TMPR[0] Trigger<br><br>Value Description<br>0 Default<br>1 A tamper event has been detected on TMPR[0] |

## Register 32: Hibernation Peripheral Properties (HIBPP) , offset 0xFC0

This register describes the features available within the Hibernation Module.

Hibernation Peripheral Properties (HIBPP)

Base 0x400F.C000
Offset 0xFC0
Type RO, reset 0x0000.0002




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0 | 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. |
| 1 | TAMPER | RO | 0x1 | Tamper Pin Presence<br><br>Value Description<br>0 Tamper module is not present.<br>1 Tamper module is present. |
| 0 | WAKENC | RO | 0x0 | Wake Pin Presence<br><br>Value Description<br>0 $\overline{\text{WAKE}}$ pin is present.<br>1 $\overline{\text{WAKE}}$ pin is not part of the package pinout. |

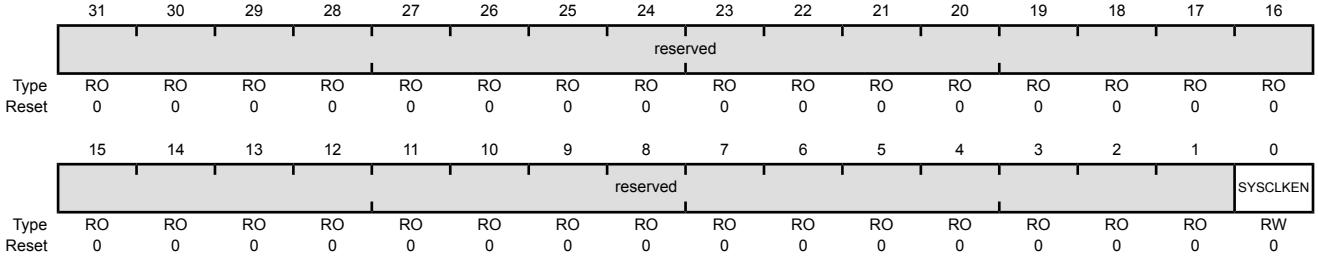
## Register 33: Hibernation Clock Control (HIBCC), offset 0xFC8

This register enables alternate clock sources.

**Note:** This register is in the system clock domain. Writes to this register do not require waiting for the WRC bit of the **HIBCTL** register to be set.

Hibernation Clock Control (HIBCC)

Base 0x400F.C000
Offset 0xFC8
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000 | 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. |
| 0 | SYSCLKEN | RW | 0x0 | RTCOSC to System Clock Enable<br>This bit RTCOSC clock to be sent to the system control for selection as a possible system clock source. Default mode is disabled to support low power modes.<br><br>Value Description<br>0 RTCOSC is not available as a system clock source.<br>1 RTCOSC is available for use as a system clock source. |

# 8 Internal Memory

The TM4C129XKCZAD microcontroller comes with 256 KB of bit-banded SRAM, internal ROM, 512 KB of Flash memory, and 6KB of EEPROM.

The TM4C129XKCZAD microcontroller provides 512 KB of on-chip Flash memory. The Flash memory is configured as four banks of 8 K x 128 bits (4 * 128 KB total) which are two-way interleaved. Memory blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

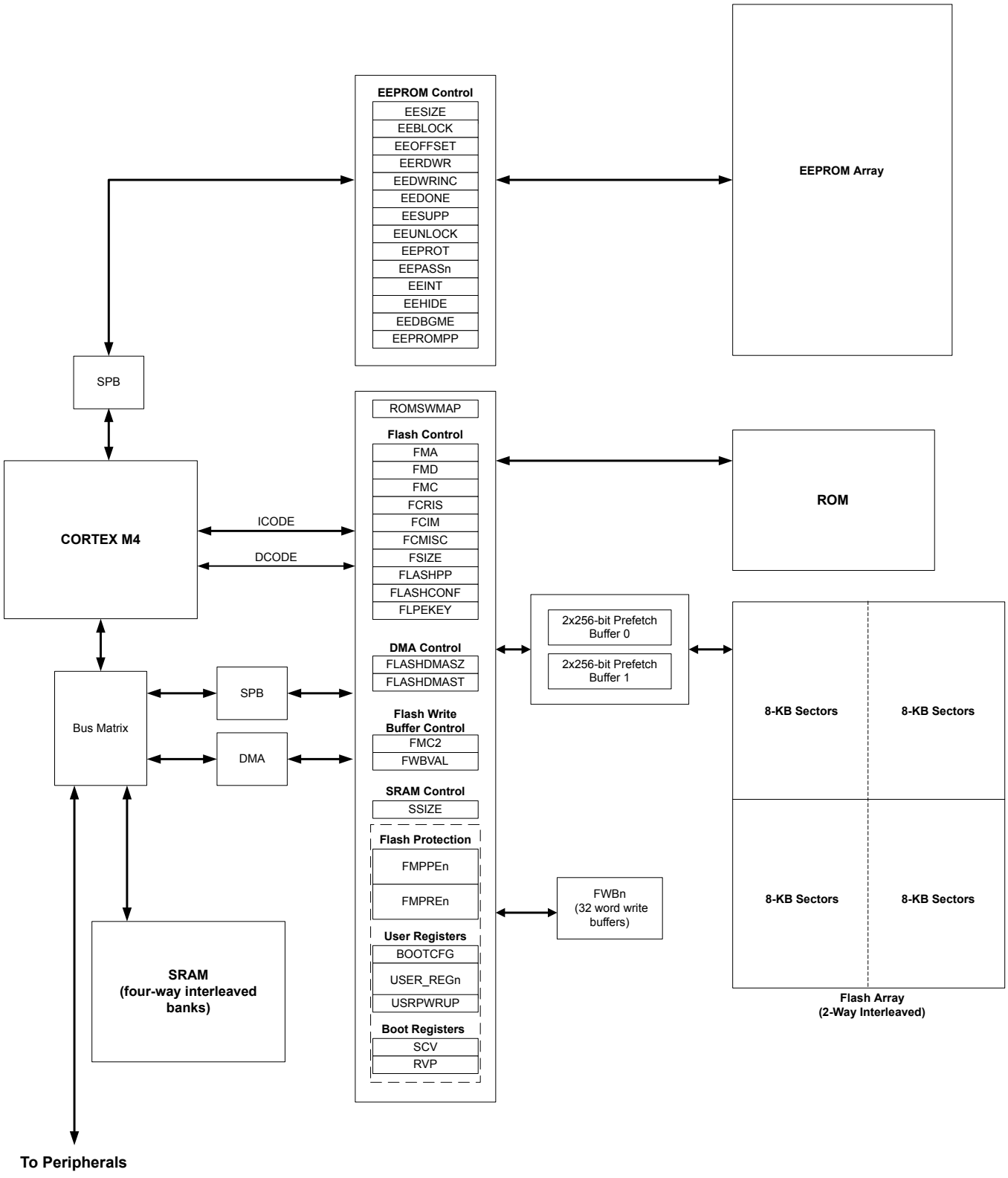
The TM4C129XKCZAD microcontroller provides enhanced performance and power savings by implementation of two sets of instruction prefetch buffers. Each prefetch buffer is 2 x 256 bits and can be combined as a 4 x 256-bit prefetch buffer.

The EEPROM module provides a well-defined register interface to support accesses to the EEPROM with both a random access style of read and write as well as a rolling or sequential access scheme. A password model allows the application to lock one or more EEPROM blocks to control access on 16-word boundaries.

## 8.1 Block Diagram

Figure 8-1 on page 633 illustrates the internal memory and control structure . The dashed box in the figure indicate registers residing in the System Control module.

**Figure 8-1. Internal Memory Block Diagram**



## 8.2 Functional Description

This section describes the functionality of the SRAM, ROM, Flash, and EEPROM memories.

**Note:** The µDMA has read-only access to flash (in Run Mode only).

### 8.2.1 SRAM

The internal system SRAM of the Tiva™ C Series devices is located at address 0x2000.0000 of the device memory map. To reduce the number of time consuming read-modify-write (RMW) operations, ARM provides bit-banding technology in the processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation. The bit-band base is located at address 0x2200.0000.

The bit-band alias is calculated by using the formula:

```
bit-band alias = bit-band base + (byte offset * 32) + (bit number * 4)
```

For example, if bit 3 at address 0x2000.1000 is to be modified, the bit-band alias is calculated as:

```
0x2200.0000 + (0x1000 * 32) + (3 * 4) = 0x2202.000C
```

With the alias address calculated, an instruction performing a read/write to address 0x2202.000C allows direct access to only bit 3 of the byte at address 0x2000.1000.

For details about bit-banding, see “Bit-Banding” on page 119.

**Note:** The SRAM is implemented using four-way 32-bit wide interleaved SRAM banks (separate SRAM arrays) which allow for increased speed between memory accesses. When using interleaving, a write to one bank followed by a read of another bank can occur in successive clock cycles without incurring any delay. However, a write access that is followed immediately by a read access to the same bank incurs a stall of a single clock cycle.

The SRAM memory layout allows for multiple masters to access different SRAM banks simultaneously. If two masters attempt to access the same SRAM bank, the master with the higher priority gains access to the memory bus and the master with the lower priority is stalled by one wait state. If four masters attempt to access the same SRAM bank, access by the master with the lowest priority is delayed by three wait states. The CPU core always has the highest priority for SRAM memory accesses.

### 8.2.2 ROM

The internal ROM of the Tiva™ C Series device is located at address 0x0100.0000 of the device memory map. Detailed information on the ROM contents can be found in the *Tiva™ C Series TM4C129x ROM User’s Guide (literature number SPMU363)*.

The ROM contains the following components:

- TivaWare™ Boot Loader and vector table
- TivaWare Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces
- Advanced Encryption Standard (AES) cryptography tables
- Cyclic Redundancy Check (CRC) error detection functionality

The boot loader is used as an initial program loader (when the Flash location 0x0000.0004, the reset vector location is all 1s (that is, erased state of Flash)) as well as an application-initiated

firmware upgrade mechanism (by calling back to the boot loader). The Peripheral Driver Library APIs in ROM can be called by applications, reducing Flash memory requirements and freeing the Flash memory to be used for other purposes (such as additional features in the application). Advanced Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government. Cyclic Redundancy Check (CRC) is a technique to validate whether a block of data has the same contents as when previously checked.

**Note:** CRC and AES software programs are available in TivaWare for backward-compatibility. A device that has enhanced CRC and AES integrated modules should utilize this hardware for best performance. Please refer to “Cyclical Redundancy Check (CRC)” on page 979 and “Advance Encryption Standard Accelerator (AES)” on page 988 for more information.

#### 8.2.2.1 Boot Configuration

After Power-On-Reset (POR) and device initialization occurs, the hardware loads the stack pointer from either flash or ROM based on the presence of an application in flash and the state of the EN bit in the **BOOTCFG** register. If the flash address 0x0000.0004 contains an erased word (value 0xFFFF.FFFF) or the EN bit is of the **BOOTCFG** register is clear, the stack pointer and reset vector pointer are loaded from ROM at address 0x0100.0000 and 0x0100.0004, respectively. The boot loader executes and configures the available boot slave interfaces and waits for an external memory to load its software. The boot loader uses a simple packet interface to provide synchronous communication with the device. The speed of the boot loader is determined by the internal oscillator (PIOSC) frequency. The following serial interfaces can be used:

- UART0
- SSI0
- $I^2C0$
- USB

If the check of the Flash at address 0x0000.0004 contains a valid reset vector value and the EN bit in the **BOOTCFG** register is set, the stack pointer and reset vector values are fetched from the beginning of flash. This application stack pointer and reset vector are loaded and the processor executes the application directly. Otherwise, the stack pointer and reset vector values are fetched from the beginning of ROM.

#### 8.2.2.2 TivaWare Peripheral Driver Library

The TivaWare Peripheral Driver Library contains a file called `driverlib/rom.h` that assists with calling the peripheral driver library functions in the ROM. The detailed description of each function is available in the *Tiva™ C Series TM4C129x ROM User’s Guide (literature number SPMU363)*. See the "Using the ROM" chapter of the *TivaWare™ Peripheral Driver Library for C Series User's Guide (literature number SPMU298)* for more details on calling the ROM functions and using `driverlib/rom.h`.

A table at the beginning of the ROM points to the entry points for the APIs that are provided in the ROM. Accessing the API through these tables provides scalability; while the API locations may change in future versions of the ROM, the API tables do not. The tables are split into two levels; the main table contains one pointer per peripheral which points to a secondary table that contains one pointer per API that is associated with that peripheral. The main table is located at 0x0100.0010, right after the Cortex-M4F vector table in the ROM.

DriverLib functions are described in detail in the *TivaWare™ Peripheral Driver Library for C Series User's Guide (literature number SPMU298)*.

Additional APIs are available for graphics and USB functions, but are not preloaded into ROM. The TivaWare Graphics Library provides a set of graphics primitives and a widget set for creating graphical

user interfaces on Tiva™ C Series microcontroller-based boards that have a graphical display (for more information, see the *TivaWare™ Graphics Library for C Series User's Guide (literature number SPMU300)*). The TivaWare USB Library is a set of data types and functions for creating USB Device, Host or On-The-Go (OTG) applications on Tiva™ C Series microcontroller-based boards (for more information, see the *TivaWare™ USB Library for C Series User's Guide (literature number SPMU297)*).

#### 8.2.2.3 Advanced Encryption Standard (AES) Cryptography Tables

AES is a strong encryption method with reasonable performance and size. AES is fast in both hardware and software, is fairly easy to implement, and requires little memory. AES is ideal for applications that can use prearranged keys, such as setup during manufacturing or configuration. Four data tables used by the XySSL AES implementation are provided in the ROM. The first is the forward S-box substitution table, the second is the reverse S-box substitution table, the third is the forward polynomial table, and the final is the reverse polynomial table. See the *Tiva™ C Series TM4C129x ROM User's Guide (literature number SPMU363)* for more information on AES.

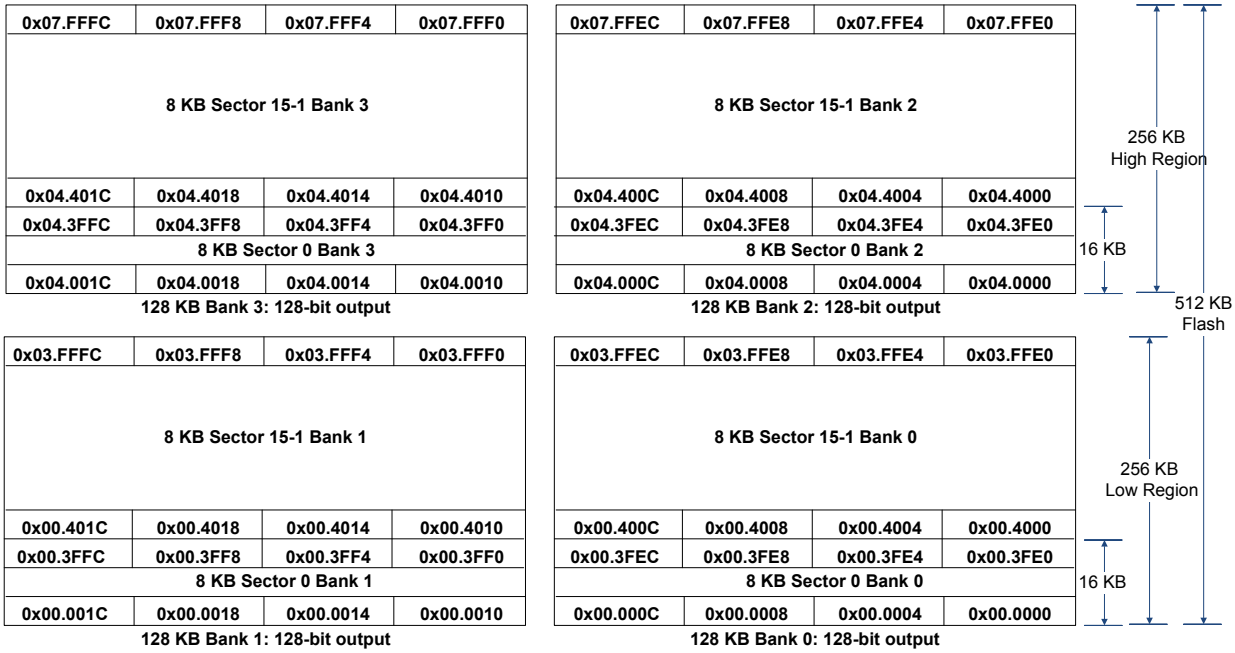
#### 8.2.2.4 Cyclic Redundancy Check (CRC) Error Detection

The CRC technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that Flash memory contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (for example, XOR all bits) because it catches changes more readily. When device initialization is executing from ROM, a CRC-32 validates the data being transferred into registers and memory. The CRC ensures no instructions were skipped in a sequence or no data was corrupted during transfer. See the *Tiva™ C Series TM4C129x ROM User's Guide (literature number SPMU363)* for more information on CRC.

### 8.2.3 Flash Memory

The Flash memory is configured in groups of four banks four banks of 8 K x 128 bits (4 * 128 KB total) which are two-way interleaved as shown below.

**Figure 8-2. Flash Memory Configuration**




The interleaved memory prefetchs 256 bits at a time. The prefetch buffers allow the maximum performance of a 120 MHz CPU speed to be maintained with linear code or loops that fit within the prefetch buffer. It is recommended that code be compiled with switches set to eliminate "literals" as much as possible as a literal causes a flash access for that word and a stall for the wait states. Most compilers support transforming literals into "in-line" code, which executes faster in a system where the memory subsystem is slower than the CPU.

Because the memory is two-way interleaved and each bank individually is an 8-KB sector, when the user erases a sector, using the ERASE bits in the **Flash Memory Control (FMC)** register, it is a 16 KB erase. Erasing a block causes the entire contents of the block to be reset to all 1s.

### 8.2.3.1 Flash Configuration

Depending on the CPU frequency, the application must program the Flash clock high time (FBCHT), Flash Bank Clock Edge (FBCE) and Flash wait states (FWS) in the **Memory Timing Parameter Register 0 for Main Flash and EEPROM (MEMTIM0)**, System Control Module offset 0x0C0. The following table details the bit field values that are required for the given CPU frequency ranges.

**Table 8-1. MEMTIM0 Register Configuration versus Frequency**

| CPU Frequency range (f) in MHz | Time Period Range (t) in ns | Flash Bank Clock High Time (FBCHT) | Flash Bank Clock Edge (FBCE) | Flash Wait States (FWS) |
|---|---|---|---|---|
| 16 | 62.5 | 0x0 | 1 | 0x0 |
| 16 < f ≤ 40 | 62.5 > t ≥ 25 | 0x2 | 0 | 0x1 |
| 40 < f ≤60 | 25 > t ≥ 16.67 | 0x3 | 0 | 0x2 |
| 60< f ≤80 | 16.67 > t ≥ 12.5 | 0x4 | 0 | 0x3 |
| 80 < f ≤100 | 12.5 > t ≥ 10 | 0x5 | 0 | 0x4 |

**Table 8-1. MEMTIM0 Register Configuration versus Frequency *(continued)***

| CPU Frequency range (f) in MHz | Time Period Range (t) in ns | Flash Bank Clock High Time (FBCHT) | Flash Bank Clock Edge (FBCE) | Flash Wait States (FWS) |
|---|---|---|---|---|
| 100< f ≤120 | 10 > t ≥ 8.33 | 0x6 | 0 | 0x5 |

To update the **MEMTIM0** register with the new Flash configuration values, the MEMTIMU bit should be set in the **Run and Sleep Mode Configuration Register (RSCLKCFG)**, System Control offset 0x0B0.

**Note:** The associated Flash and EEPROM fields in the **MEMTIM0** register must be programmed to the same values. For example, the FWS field must be programmed to the same value as the EWS field.

### 8.2.3.2 Prefetch Buffers

The prefetch buffers can exist as a single set of 2x256-bit buffers or 4x256-bit buffers depending on the SPFE bit programmed in the **Flash Configuration Register (FLASHCONF)** register, offset 0xFC8. At reset, all four buffers are enabled. The buffers are filled using a "least-recently-used" (LRU) method. When operating in a single set buffer configuration, the two, 256-bit buffers create a deterministic configuration as each "next" write is sent to the previous buffer that was written. Figure 8-3 on page 638 depicts the single 256-bit buffer set. The single prefetch buffer set should only be used when the code execution must be purely deterministic for the number of clock cycles it takes to execute. Utilizing the four prefetch buffer configuration is the preferred method of configuration.

**Figure 8-3. Single 256-Bit Prefetch Buffer Set**

| | | 255 | 224 | 223 | 192 | 191 | 160 | 159 | 128 | 127 | 96 | 95 | 64 | 63 | 32 | 31 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Prefetch Buffer 0 | **TAG** | WORD 7 | | WORD 6 | | WORD 5 | | WORD 4 | | WORD 3 | | WORD 2 | | WORD 1 | | WORD 0 | |
| Prefetch Buffer 1 | **TAG** | WORD 7 | | WORD 6 | | WORD 5 | | WORD 4 | | WORD 3 | | WORD 2 | | WORD 1 | | WORD 0 | |

When the buffers are configured as four, 256-bit buffers, they function as one set, with one of the four buffers tagged as the LRU and the next to be used when an auto-fill or miss occurs.

**Figure 8-4. Four 256-Bit Prefetch Buffer Configuration**

| | | 255 | 224 | 223 | 192 | 191 | 160 | 159 | 128 | 127 | 96 | 95 | 64 | 63 | 32 | 31 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Prefetch Buffer 0 | **TAG** | WORD 7 | | WORD 6 | | WORD 5 | | WORD 4 | | WORD 3 | | WORD 2 | | WORD 1 | | WORD 0 | |
| Prefetch Buffer 1 | **TAG** | WORD 7 | | WORD 6 | | WORD 5 | | WORD 4 | | WORD 3 | | WORD 2 | | WORD 1 | | WORD 0 | |
| Prefetch Buffer 2 | **TAG** | WORD 7 | | WORD 6 | | WORD 5 | | WORD 4 | | WORD 3 | | WORD 2 | | WORD 1 | | WORD 0 | |
| Prefetch Buffer 3 | **TAG** | WORD 7 | | WORD 6 | | WORD 5 | | WORD 4 | | WORD 3 | | WORD 2 | | WORD 1 | | WORD 0 | |

The address of the auto-fill is stored in this tag register so that address violations can be identified immediately and miss processing can begin directly. Every ICODE access is checked against valid tags to see if the target word is already in the buffers.

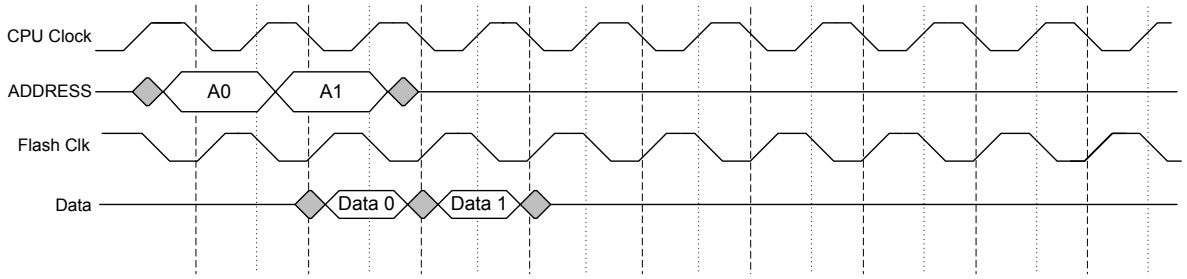
If there is a hit, the target word is immediately sent to the CPU with no wait states. If there is a miss, then the prefetch buffer is invalidated and the miss is processed as a 256-bit read from the flash

subsystem to fill the next, least-recently used prefetch buffer. Two memory banks are read in parallel to retrieve 256-bits worth of data.

If an auto-fill has been started and a miss occurs, the auto-fill completes before the miss is processed. If an auto-fill occurs that hits the prefetch buffer being processed for the auto-fill, then the ICODE bus is stalled until the auto fill is complete and new entry can be accessed. For an instruction miss, access to the flash bank starts immediately after the address is available provided the flash sub-system is not already processing a DCODE bus access or a PROGRAM/ERASE operation in the same banks. The target word is passed to the CPU one cycle after it is written to the prefetch buffer.

Figure 8-5 on page 639 shows the timing diagram for a hit in the prefetch buffer.

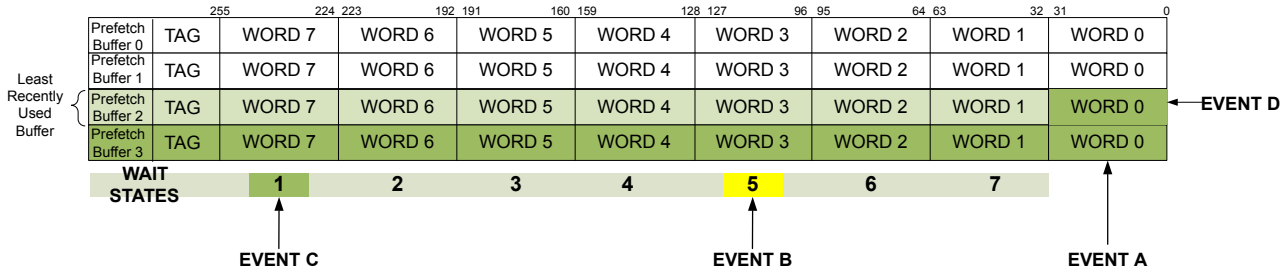
**Figure 8-5. Single Cycle Access, 0 Wait States**




The Flash memory can operate at the CPU clock speed with zero-wait-state accesses when data is resident in the prefetch buffers. When an access does not hit in the prefetch buffer, there is a delay that is incurred while the data is transferred from the Flash. This delay is dependent on the programmed CPU frequency. Refer to Table 8-1 on page 637 for required CPU frequency versus programmed wait-state delay information. Figure 8-6 on page 640 depicts the events that occur as the CPU steps through the words in the prefetch buffer that has just been loaded until it reaches the end of the current prefetch line. The notable events are as follows (refer to Figure 8-6 on page 640):

- EVENT A: When the CPU has a miss in the prefetch buffer, a line is fetched from Flash. The target word is written to the prefetch buffer and sent to the CPU one cycle after.
- EVENT B: When the CPU reaches Word 3, the next 256-bit buffer line is fetched, resulting in a zero-wait-state access of next line's Word 0
- EVENT C: After this word, if the CPU is still executing sequentially, Word 0 of the next buffer line that was fetched is sent to the CPU, with zero-wait-state delay
- EVENT D: Word 0 from the second fetch that occurred is sent to the CPU

**Figure 8-6. Prefetch Fills from Flash**




Note that if the CPU target word is beyond Word 2 (Word 3 through Word 7) then the next prefetch fill begins immediately and, depending on the CPU frequency, a delay is incurred between CPU access of Word 7 and Word 0 of the next line.

**Note:** For optimal prefetch buffer performance, align application code/branches on 8-word boundaries.

**Note:** Because the prefetch buffers and Flash memory can effectively be utilized at 20 Mhz and above, an application may see an improvement in current consumption from 16 MHz to 20 MHz.

The prefetch buffers can be forced ON and OFF by setting the FPFON and FPFOFF bits in the **Flash Configuration (FLASHCONF)** register at 0xFC8. If the application sets the FPFON or FPFOFF bit while the CPU is currently reading or writing to Flash, the prefetch buffer action of turning on or off happens only after the Flash operation has completed. This feature can be used in test modes when determining optimum memory configuration for code.

Prefetch buffer valid tags can be cleared in the following ways:

- Any **Flash Configuration (FLASHCONF)** register changes, such as:
  - Disabling the prefetch buffer by setting the FPFOFF bit
  - Setting the CLRTV bit to clear the prefetch buffer tags
- A system reset
- ROM accesses
- Error during ICODE accesses
- System aborts
- Mirror mode changes

**Note:** If the prefetch buffers are enabled and application code branches to a location other than flash memory which then modifies the flash memory, the prefetch tags must be cleared before returning to flash code execution. Prefetch buffer valid tags can be cleared by setting the CLRTV bit in the **FLASHCONF** register.

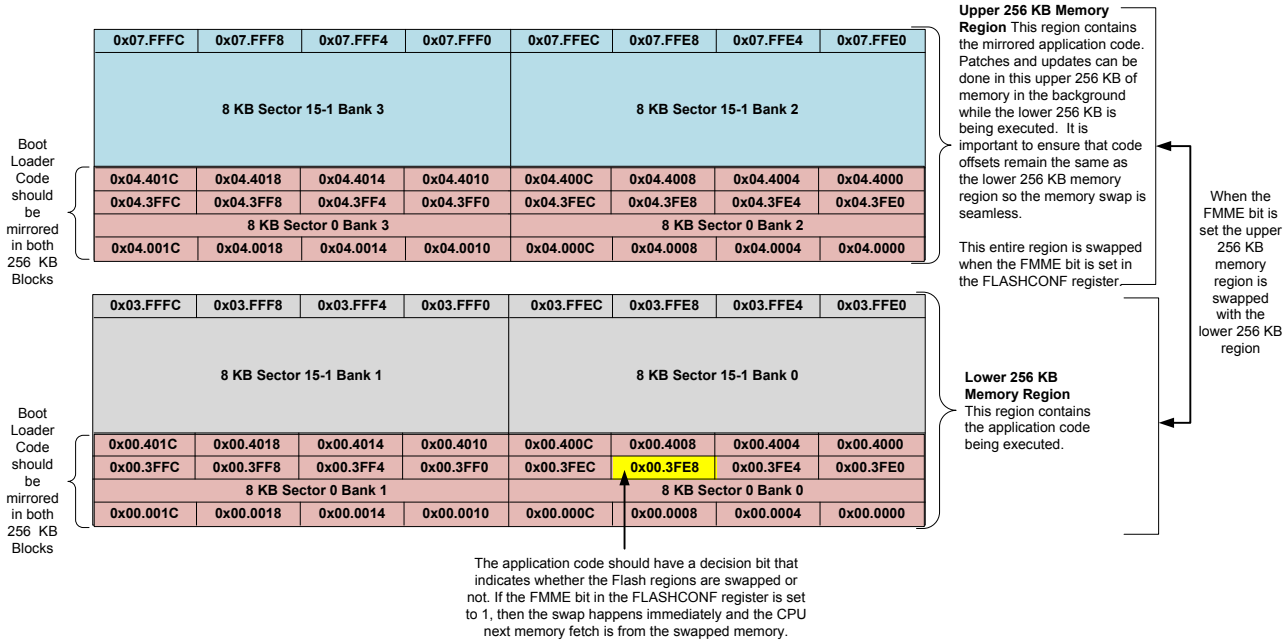
### 8.2.3.3 Flash Mirror Mode

Flash mirroring allows multiple copies of software to exist in Flash simultaneously. The software can run from the lower banks at the same time software is updating a mirrored copy on the upper bank. In addition to the data, the boot loader in both the lower and upper banks must be mirrored

while programming the flash contents. If data needs to be recovered, a hot swap can be done by setting the FMME bit in the **FLASHCONF** register to ensure the flash banks are idle during the swap. The prefetch buffers must be invalidated during the execution of a hot swap. Next, the address translation logic decodes up to 256 KB from the upper banks to the lower banks. Once the banks are swapped, the mirrored flash image is then used. The address translation logic translates the address to the upper banks until the next swap. Figure 8-7 on page 641 depicts the configuration necessary when executing Flash mirroring.

**Note:** After a mirror mode has been executed and the code locations have been swapped from the upper memory banks to the lower, the application can continue to read from the lower memory bank address locations. However, when erasing or programming the swapped memory, the application must use the "real" upper memory address of the code before it was swapped. For example, in Figure 8-7 on page 641, when the yellow highlighted location 0x00.3FE8 is swapped with 0x04.3FE8 the application's next read location is 0x00.3FEC. However, if the application were to program or erase the next location it would need to write or erase location 0x04.3FEC.

**Figure 8-7. Mirror Mode Function**




### 8.2.3.4 Protected Flash Memory Registers

The user is provided execution protection through of 32-bit wide registers. The policy for each protection form is controlled by individual bits (per policy per block) in the **FMPPEn** and **FMPREn** registers.

- **Flash Memory Protection Program Enable (FMPPEn)**: In the Flash, 16-KB blocks can be individually protected from being programed or erased. Because each bit of the **FMPPE** register represents a 2-KB block, the application must clear all the bits in one byte to protect one 16-KB block. Execute-only protection can only be programmed in 16-KB increments. For example, to protect the first 16-KB block, bits [7:0] all need to be set to 0s. When bits in the **FMPPEn** register are set, the corresponding block may be programmed (written) or erased. When bits are cleared,

the corresponding block may not be changed. When a block is protected by clearing bits in both **FMPPEn** and **FMPREn** registers, execute-only protection can be achieved.

- **Flash Memory Protection Read Enable (FMPREn)**: If a bit is set in this register, the corresponding block may be executed or read by software or debuggers. If a bits in this register are cleared and the same block in the **FMPREn** register is cleared, the corresponding block may only be executed, and contents of the memory block are prohibited from being read as data. **FMPREn** protection can be programmed in 2-KB increments, unlike the **FMPPEn**, which must be programmed in 16-KB increments. However, if an application does want to read-protect a 16-KB block, eight bits need to be written from 1s to 0s.

The policies may be combined as shown in Table 8-2 on page 642.

**Table 8-2. Flash Memory Protection Policy Combinations**

| FMPPEn | FMPREn | Protection |
|---|---|---|
| 0 | 0 | Execute-only protection. The block may only be executed and may not be written or erased. This mode is used to protect code. |
| 1 | 0 | The block may be written, erased or executed, but not read. This combination is unlikely to be used. |
| 0 | 1 | Read-only protection. The block may be read or executed but may not be written or erased. This mode is used to lock the block from further modification while allowing any read or execute access. |
| 1 | 1 | No protection. The block may be written, erased, executed or read. |

A Flash memory access that attempts to read a read-protected block (**FMPREn** bit is clear) is prohibited and generates a bus fault. A Flash memory access that attempts to program or erase a program-protected block (**FMPPEn** bit is clear) is prohibited and can optionally generate an interrupt (by setting the AMASK bit in the **Flash Controller Interrupt Mask (FCIM)** register) to alert software developers of poorly behaving software during the development and debug phases.

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. These settings create a policy of open access and programmability. The register bits may be changed by clearing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a any simulated power-on-reset (SIM_POR) event. The changes are committed using the **Flash Memory Control (FMC)** register. Details on programming these bits are discussed in “Non-Volatile Register Programming-- Flash Memory Resident Registers” on page 645.

### 8.2.3.5 Execute-Only Protection

Execute-only protection prevents both modification and visibility to a protected flash block. This mode is intended to be used in situations where a device requires debug capability, yet portions of the application space must be protected from external access. An example of this is a company that wishes to sell Tiva™ C Series devices with their proprietary software preprogrammed, yet allow the end user to add custom code to an unprotected region of the flash (such as a motor control module with a customizable motor configuration section in flash).

Literal data introduces a complication to the protection mechanism. When C code is compiled and linked, literal data (constants, and so on) is typically placed in the text section, between functions, by the compiler. The literal data is accessed at run time through the use of the LDR instruction, which loads the data from memory using a PC-relative memory address. The execution of the LDR instruction generates a read transaction across the Cortex-M3's DCode bus, which is subject to the execute-only protection mechanism. If the accessed block is marked as execute only, the transaction

is blocked, and the processor is prevented from loading the constant data and, therefore, inhibiting correct execution. Therefore, using execute-only protection requires that literal data be handled differently. There are three ways to address this:

1. Use a compiler that allows literal data to be collected into a separate section that is put into one or more read-enabled flash blocks. Note that the LDR instruction may use a PC-relative address, in which case the literal pool cannot be located outside the span of the offset, or the software may reserve a register to point to the base address of the literal pool and the LDR offset is relative to the beginning of the pool.
2. Use a compiler that generates literal data from arithmetic instruction immediate data and subsequent computation.
3. Use method 1 or 2, but in assembly language, if the compiler does not support either method.

#### 8.2.3.6 Read-Only Protection

Read-only protection prevents the contents of the flash block from being re-programmed, while still allowing the content to be read by processor or the debug interface. Note that if a **FMPREn** bit is cleared, all read accesses to the Flash memory block are disallowed, including any data accesses. Care must be taken not to store required data in a Flash memory block that has the associated **FMPREn** bit cleared.

The read-only mode does not prevent read access to the stored program, but it does provide protection against accidental (or malicious) erasure or programming. Read-only is especially useful for utilities like the boot loader when the debug interface is permanently disabled. In such combinations, the boot loader, which provides access control to the Flash memory, is protected from being erased or modified.

#### 8.2.3.7 Permanently Disabling Debug

For extremely sensitive applications, the debug interface to the processor and peripherals can be permanently disabled, blocking all accesses to the device through the JTAG or SWD interfaces. With the debug interface disabled, it is still possible to perform standard IEEE instructions (such as boundary scan operations), but access to the processor and peripherals is blocked.

The DBG0 and DBG1 bits of the **Boot Configuration (BOOTCFG)** register control whether the debug interface is turned on or off.

The debug interface should not be permanently disabled without providing some mechanism, such as the boot loader, to provide customer-installable updates or bug fixes. Disabling the debug interface is permanent and cannot be reversed.

#### 8.2.3.8 Interrupts

The Flash memory controller can generate interrupts when the following conditions are observed:

- Programming Interrupt: Signals when a program or erase action is complete. (PRIS).
- Access Interrupt: Signals when a program or erase action has been attempted on a 16-kB block of memory that is protected by its corresponding **FMPPEn** bit. (ARIS).
- EEPROM Interrupt
- Pump Voltage Interrupt: Indicates if the regulated voltage of the pump went out of specification during a Flash operation and the operation was terminated. (VOLTRIS).

- Invalid Data Interrupt: Signals when a bit in Flash that was previously programmed as a 0 is now requested to be programmed as a 1. (INVDRIS).
- ERASE Operation Interrupt: Indicates an ERASE operation failed. (ERRIS).

The interrupt events that can trigger a controller-level interrupt are defined in the **Flash Controller Masked Interrupt Status (FCMIS)** register (see page 664) by setting the corresponding MASK bits. If interrupts are not used, the raw interrupt status is always visible via the **Flash Controller Raw Interrupt Status (FCRIS)** register (see page 661).

Interrupts are always cleared (for both the **FCMIS** and **FCRIS** registers) by writing a 1 to the corresponding bit in the **Flash Controller Masked Interrupt Status and Clear (FCMISC)** register (see page 666).

#### 8.2.3.9 µDMA

The µDMA can be programmed to read from Flash. The **Flash DMA Address Size (FLASHDMASZ)** register configures 2-KB regions of Flash that can be accessed by the µDMA. The starting address for this µDMA-accessible region is defined in the **Flash DMA Starting Address (FLASHDMAST)** register. When the DFA bit is set in the **FLASHPP** register, the µDMA can access the enabled region configured by the **FLASHDMASZ** and **FLASHDMAST** registers. The µDMA checks the **Flash Protection Program Enable n (FMPPEn)** registers for masked 2-KB Flash regions before initiating the transfer. If the access is out of range, then a bus fault is generated.

**Note:** The µDMA can access Flash in Run Mode only (not available in low power modes).

#### 8.2.3.10 Flash Memory Programming

The Tiva™ C Series devices provide a user-friendly interface for Flash memory programming. All erase/program operations are handled via three registers: **Flash Memory Address (FMA)**, **Flash Memory Data (FMD)**, and **Flash Memory Control (FMC)**. Note that if the debug capabilities of the microcontroller have been deactivated, resulting in a "locked" state, a recovery sequence must be performed in order to reactivate the debug module. See “Recovering a "Locked" Microcontroller” on page 224.

When a Flash memory operation write, page erase, or mass erase is executed in a Flash bank, access to that particular bank pair is inhibited. As a result, instruction and literal fetches to the bank pair are held off until the Flash memory operation is complete. If instruction execution is required during a Flash memory operation, the code that is executing must be placed in SRAM and executed from there while the flash operation is in progress.

**Note:** When programming Flash memory, the following characteristics of the memory must be considered:

- Only an erase can change bits from 0 to 1.
- A write can only change bits from 1 to 0. If the write attempts to change a 0 to a 1, the write fails and no bits are changed.
- All Flash operations are completed before entering sleep or deep sleep.

***To program a 32-bit word***

1. Write source data to the **FMD** register.
2. Write the target address to the **FMA** register.

3. Write the Flash memory write key and the WRITE bit (a value of 0xA442.0001) to the **FMC** register. The write key may be 0xA442 or the value programmed into the **FLPEKEY** register depending on the KEY value in the **BOOTCFG** register. See page 704 and page 671 for more information.

4. Poll the **FMC** register until the WRITE bit is cleared.

***To perform an erase of a 16-KB sector***

1. Write the 16-KB aligned address to the **FMA** register.

2. Write the Flash memory write key and the ERASE bit to the **FMC** register.

3. Poll the **FMC** register until the ERASE bit is cleared or, alternatively, enable the programming interrupt using the PMASK bit in the **FCIM** register.

***To perform a mass erase of the Flash memory***

1. Write the Flash memory write key and the MERASE bit to the **FMC** register.

2. Poll the **FMC** register until the MERASE bit is cleared or, alternatively, enable the programming interrupt using the PMASK bit in the **FCIM** register.

#### 8.2.3.11 32-Word Flash Memory Write Buffer

A 32-word write buffer provides the capability to perform faster write accesses to the Flash memory by programming two 32-bit words at a time, allowing 32 words to be programmed in the same time as 16 would take using the method described above. The data for the buffered write is written to the **Flash Write Buffer (FWBn)** registers.

The registers are 32-word aligned with Flash memory, and therefore the register **FWB0** corresponds with the address in **FMA** where bits [6:0] of **FMA** are all 0. **FWB1** corresponds with the address in **FMA** + 0x4 and so on. Only the **FWBn** registers that have been updated since the previous buffered Flash memory write operation are written. The **Flash Write Buffer Valid (FWBVAL)** register shows which registers have been written since the last buffered Flash memory write operation. This register contains a bit for each of the 32 **FWBn** registers, where bit[n] of **FWBVAL** corresponds to **FWBn**. The **FWBn** register has been updated if the corresponding bit in the **FWBVAL** register is set.

***To program 32 words with a single buffered Flash memory write operation***

1. Write the source data to the **FWBn** registers.

2. Write the target address to the **FMA** register. This must be a 32-word aligned address (that is, bits [6:0] in **FMA** must be 0s).

3. Write the Flash memory write key and the WRBUF bit to the **FMC2** register.

4. Poll the **FMC2** register until the WRBUF bit is cleared or wait for the PMIS interrupt to be signaled.

#### 8.2.3.12 Non-Volatile Register Programming-- Flash Memory Resident Registers

**Note:** The **Boot Configuration (BOOTCFG)** register requires a POR before the committed changes take effect.

This section discusses how to update the registers shown in Table 8-3 on page 646, which are resident within the Flash Memory. These registers exist in a separate space from the main Flash

memory array and are not affected by an ERASE or MASS ERASE operation. The bits in these registers can be changed from 1 to 0 with a commit operation. The register contents are unaffected by any reset condition except power-on reset, which returns the register contents to 0xFFFF.FFFE for the **BOOT Configuration (BOOTCFG)** register and 0xFFFF.FFFF for all others.

By committing the register values using the COMT bit in the **Flash Memory Control (FMC)** register, the register contents become non-volatile and are therefore retained following power cycling. Once the register contents are committed, the only way to restore the factory default values is to perform the sequence described in “Recovering a "Locked" Microcontroller” on page 224.

All of the **FMPREn**, **FMPPEn** and **USER_REGn** registers, in addition to the **BOOTCFG** register can be committed in non-volatile memory. The **FMPREn**, **FMPPEn**, and **USER_REGn** registers can be tested before being committed; the **BOOTCFG** register cannot. To program the **BOOTCFG** register, the value must be written into the **Flash Memory Data (FMD)** register before it is committed. The **BOOTCFG** configuration cannot be tried and verified before committing to non-volatile memory.

**Important:** All Flash memory resident registers can only have bits changed from 1 to 0 by user programming. The **FMPREn, FMPPEn** and **BOOTCFG** registers can be committed multiple times, but the **USER_REGn** registers can only be committed once, after the entire register has been set to 1s. After being committed, the **USER_REGn** registers can only be returned to their factory default values of all 1s by performing the sequence described in “Recovering a "Locked" Microcontroller” on page 224. The mass erase of the main Flash memory array caused by the sequence is performed prior to restoring these registers.

Table 8-3 on page 646 provides the **FMA** address required for commitment of each of the registers and the source of the data to be written when the **FMC** register is written with a key value of 0xA442 or the PEKEY value of the **FLPEKEY** register. The key value used is determined by the KEY bit in the **BOOTCFG** register at reset. If the KEY value is 0x0, the PEKEY value in the **FLPEKEY** register is used for commits in the **FMC/FMC2** register. If the KEY value is 0x1, the value 0xA442 is used as the WRKEY in the **FMC/FMC2** register. If the After writing the COMT bit, the user may poll the **FMC** register to wait for the commit operation to complete.

**Note:** To ensure non-volatile register data integrity, non-volatile register commits should not be interrupted with a power loss. If data integrity is compromised during a commit because of a power loss, a toggle mass erase function can be performed to clear these registers. See Table 8-3 on page 646 for the list of non-volatile registers.

**Table 8-3. User-Programmable Flash Memory Resident Registers**

| Register to be Committed | FMA Value | Data Source |
|---|---|---|
| FMPRE0 | 0x0000.0000 | FMPRE0 |
| FMPRE1 | 0x0000.0002 | FMPRE1 |
| FMPRE2 | 0x0000.0004 | FMPRE2 |
| FMPRE3 | 0x0000.0006 | FMPRE3 |
| FMPRE4 | 0x0000.0008 | FMPRE4 |
| FMPRE5 | 0x0000.000A | FMPRE5 |
| FMPRE6 | 0x0000.000C | FMPRE6 |
| FMPRE7 | 0x0000.000E | FMPRE7 |
| FMPPE0 | 0x0000.0001 | FMPPE0 |
| FMPPE1 | 0x0000.0003 | FMPPE1 |
| FMPPE2 | 0x0000.0005 | FMPPE2 |

**Table 8-3. User-Programmable Flash Memory Resident Registers *(continued)***

| Register to be Committed | FMA Value | Data Source |
|---|---|---|
| FMPPE3 | 0x0000.0007 | FMPPE3 |
| FMPPE4 | 0x0000.0009 | FMPPE4 |
| FMPPE5 | 0x0000.000B | FMPPE5 |
| FMPPE6 | 0x0000.000D | FMPPE6 |
| FMPPE7 | 0x0000.000F | FMPPE7 |
| USER_REG0 | 0x8000.0000 | USER_REG0 |
| USER_REG1 | 0x8000.0001 | USER_REG1 |
| USER_REG2 | 0x8000.0002 | USER_REG2 |
| USER_REG3 | 0x8000.0003 | USER_REG3 |
| BOOTCFG | 0x7510.0000 | FMD |

## 8.2.4 EEPROM

The TM4C129XKCZAD microcontroller includes an EEPROM with the following features:

- 6Kbytes of memory accessible as 1536 32-bit words
- 96 blocks of 16 words (64 bytes) each
- Built-in wear leveling
- Access protection per block
- Lock protection option for the whole peripheral as well as per block using 32-bit to 96-bit unlock codes (application selectable)
- Interrupt support for write completion to avoid polling
- Endurance of 500K writes (when writing at fixed offset in every alternate page in circular fashion) to 15M operations (when cycling through two pages ) per each 2-page block.

### 8.2.4.1 Functional Description

The EEPROM module provides a well-defined register interface to support accesses to the EEPROM with both a random access style of read and write as well as a rolling or sequential access scheme.

A protection mechanism allows locking EEPROM blocks to prevent writes under a set of circumstances as well as reads under the same or different circumstances. The password model allows the application to lock one or more EEPROM blocks to control access on 16-word boundaries.

#### *Blocks*

There are 96 blocks of 16 words each in the EEPROM. These are readable and writable as words. Bytes and half-words can be read, and these accesses do not have to occur on a word boundary. The entire word is read and any unneeded data is simply ignored. The EEPROM blocks are writable only on a word basis. To write a byte, it is necessary to read the word value, modify the appropriate byte, and write the word back.

Each block is addressable as an offset within the EEPROM, using a block select register. Each word is offset addressable within the selected block.

The current block is selected by the **EEPROM Current Block (EEBLOCK)** register. The current offset is selected and checked for validity by the **EEPROM Current Offset (EEOFFSET)** register. The application may write the **EEOFFSET** register any time, and it is also automatically incremented when the **EEPROM Read-Write with Increment (EERDWRINC)** register is accessed. However, the **EERDWRINC** register does not increment the block number, but instead wraps within the block.

Blocks are individually protectable. Attempts to read from a block for which the application does not have permission return 0xFFFF.FFFF. Attempts to write into a block for which the application does not have permission results in an error in the **EEPROM Done Status (EEDONE)** register.

### *Timing Considerations*

After enabling or resetting the EEPROM module, software must wait until the WORKING bit in the **EEDONE** register is clear before accessing any EEPROM registers.

**Note:** Software must ensure there are no Flash memory writes or erases pending before performing an EEPROM operation. When the **FMC** register reads as 0x0000.00000 and the WRBUF bit of the **FMC2** register is clear, there are no Flash memory writes or erases pending.

EEPROM operations must be completed before entering Sleep or Deep-Sleep mode. Ensure the EEPROM operations have completed by checking the **EEPROM Done Status (EEDONE)** register before issuing a WFI instruction to enter Sleep or Deep-Sleep.

Writes to words within a block are delayed by a variable amount of time. The application may use an interrupt to be notified when the write is done, or alternatively poll for the done status in the **EEDONE** register. The variability ranges from the write timing of the EEPROM to the erase timing of EEPROM, where the erase timing is less than the write timing of most external EEPROMs.

Depending on the CPU frequency, the application must program the EEPROM Clock High Time (EBCHT), EEPROM Bank Clock Edge (EBCE) and the EEPROM Wait States (EWS) in the **Memory Timing Parameter Register 0 for Main Flash and EEPROM (MEMTIM0)** register at System Control Module offset 0x0C0.

**Table 8-4. MEMTIM0 Register Configuration versus Frequency**

| CPU Frequency range (f) in MHz | Time Period Range (t) in ns | EEPROM Bank Clock High Time (EBCHT) | EEPROM Bank Clock Edge (EBCE) | EEPROM Wait States (EWS) |
|---|---|---|---|---|
| 16 | 62.5 | 0x0 | 1 | 0x0 |
| 16 < f ≤ 40 | 62.5 > t ≥ 25 | 0x2 | 0 | 0x1 |
| 40 < f ≤60 | 25 > t ≥ 16.67 | 0x3 | 0 | 0x2 |
| 60< f ≤80 | 16.67 > t ≥ 12.5 | 0x4 | 0 | 0x3 |
| 80 < f ≤100 | 12.5 > t ≥ 10 | 0x5 | 0 | 0x4 |
| 100< f ≤120 | 10 > t ≥ 8.33 | 0x6 | 0 | 0x5 |

**Note:** The associated Flash and EEPROM fields in the **MEMTIM0** register must be programmed to the same values. For example, the FWS field must be programmed to the same value as the EWS field.

### *Locking and Passwords*

The EEPROM can be locked at both the module level and the block level. The lock is controlled by a password that is stored in the **EEPROM Password (EEPASSn)** registers and can be any 32-bit to 96-bit value other than all 1s. Block 0 is the master block, the password for block 0 protects the control registers as well as all other blocks. Each block can be further protected with a password for that block.

If a password is registered for block 0, then the whole module is locked at reset. As a result, the **EEBLOCK** register cannot be changed from 0 until block 0 is unlocked.

A password registered with any block, including block 0, allows for protection rules that control access of that block based on whether it is locked or unlocked. Generally, the lock can be used to prevent write accesses when locked or can prevent read and write accesses when locked.

All password protected blocks are locked at reset. To unlock a block, the correct password value must be written to the **EEPROM Unlock (EEUNLOCK)** register by writing to it once, twice, or three times, depending on the size of the password. A block or the module may be re-locked by writing 0xFFFF.FFFF to the **EEUNLOCK** register because 0xFFFF.FFFF is not a valid password.

### *Protection and Access Control*

The PROT protection field in the **EEPROM Protection (EEPROT)** register provides discrete control of read and write access for each block which allows various protection models per block. The protection configurations allowed are as follows:

- PROT = 0x0
  - Without password: Readable and writable at any time. This mode is the default when there is no password.
  - With password: Readable, but only writable when unlocked by the password. This mode is the default when there is a password.
- PROT = 0x1
  - With password: Readable or writable only when unlocked.
  - This value has no meaning when there is no password.
- PROT = 0x2
  - Without password: Readable but not writable.
  - With password: Readable only when unlocked, not writable under any conditions.

Additionally, access protection may be applied based on the processor mode. This configuration allows for supervisor-only access or supervisor and user access, which is the default. Supervisor-only access mode also prevents access by the µDMA and Debugger.

Additionally, the master block may be used to control access protection for the protection mechanism itself. If access control for block 0 is for supervisor only, then the whole module may only be accessed in supervisor mode.

### *Hidden Blocks*

Hiding provides a temporary form of protection. Every block except block 0 can be hidden, which prevents all accesses until the next reset.

This mechanism can allow a boot or initialization routine to access some data which is then made inaccessible to all further accesses. Because boot and initialization routines control the capabilities of the application, hidden blocks provide a powerful isolation of the data when debug is disabled.

A typical use model would be to have the initialization code store passwords, keys, and/or hashes to use for verification of the rest of the application. Once performed, the block is then hidden and made inaccessible until the next reset which then re-enters the initialization code.

### Power and Reset Safety

Once the **EEDONE** register indicates that a location has been successfully written, the data is retained until that location is written again. There is no power or reset race after the **EEDONE** register indicates a write has completed.

### Interrupt Control

The EEPROM module allows for an interrupt when a write completes to prevent the use of polling. The interrupt can be used to drive an application ISR which can then write more words or verify completion. The interrupt mechanism is used any time the **EEDONE** register goes from working to done, whether because of an error or the successful completion of a program or erase operation. This interrupt mechanism works for data writes, writes to password and protection registers, and mass erase using the **EEPROM Debug Mass Erase (EEDGBME)** register. The EEPROM interrupt is signaled to the core using the Flash memory interrupt vector. Software can determine that the source of the interrupt was the EEPROM by examining bit 2 of the **Flash Controller Masked Interrupt Status and Clear (FCMISC)** register.

### Theory of Operation

The EEPROM operates using a traditional bank model which implements EEPROM-type cells, but uses sector erase. Additionally, words are replicated in the blocks to allow 500K+ erase cycles when needed, which means that each word has a latest version. As a result, a write creates a new version of the word in a new location, making the previous value obsolete. When a block runs out of room to store the latest version of a word, a copy buffer is used. The copy buffer copies the latest words of each block. The original block is then erased. Finally, the copy buffer contents are copied back to the block.

The EEPROM module includes functionality to prevent data corruption due to power-loss or a brown-out event during programming or erase operations. These conditions prevent corruption of non-targeted memory areas but cannot guarantee that the operation is completed successfully. Refer to “EEPROM” on page 2125 for important timing information on EEPROM protection. The EEPROM mechanism properly tracks all state information to provide complete safety and protection. Although it should not normally be possible, errors during programming can occur in certain circumstances, for example, the voltage rail dropping during programming. In these cases, the **EESUPP** register can be used to know if a program or an erase had failed.

### Debug Mass Erase

The EEPROM debug mass erase allows the developer to mass erase the EEPROM. For the mass erase to occur correctly, there can be no active EEPROM operations. After the last EEPROM operation, the application must ensure that no EEPROM registers are updated, including modifying the **EEBLOCK** and the **EEOFFSET** registers without doing an actual read or write operation. To hold off these operations, the application should reset the EEPROM module by setting the `R0` bit in the **EEPROM Software Reset (SREEPROM**) register, wait until `WORKING` bit in the **EEPROM Done Status (EEDONE)** register is clear, and then enable the debug mass erase by setting the `ME` bit in the **EEPROM Debug Mass Erase (EEDBGME)** register.

### Error During Programming

Operations such as data-write, password set, protection set, and copy buffer erase may perform multiple operations. For example, a normal write performs two underlying writes: the control word write and the data write. If the control word writes but the data fails (for example, due to a voltage drop), the overall write fails with indication provided in the **EEDONE** register. Failure and the corrective action is broken down by the type of operation:

- If a normal write fails such that the control word is written but the data fails to write, the safe course of action is to retry the operation once the system is otherwise stable, for example, when the voltage is stabilized. After the retry, the control word and write data are advanced to the next location.
- If a password or protection write fails, the safe course of action is to retry the operation once the system is otherwise stable. In the event that multi-word passwords may be written outside of a manufacturing or bring-up mode, care must be taken to ensure all words are written in immediate succession. If not, then partial password unlock would need to be supported to recover.
- If the word write requires the block to be written to the copy buffer, then it is possible to fail or lose power during the subsequent operations. A control word mechanism is used to track what step the EEPROM was in if a failure occurs. If not completed, the **EESUPP** register indicates the partial completion.

After a reset and prior to writing any data to the EEPROM, software must read the **EESUPP** register and check for the presence of any error condition which may indicate that a write or erase was in progress when the system was reset due to a voltage drop. If either the PRETRY or ERETRY bits are set, the peripheral should be reset by setting and then clearing the R0 bit in the **EEPROM Software Reset (SREEPROM)** register and waiting for the WORKING bit in the **EEDONE** register to clear before again checking the **EESUPP** register for error indicators. This procedure should allow the EEPROM to recover from the write or erase error. In very isolated cases, the **EESUPP** register may continue to register an error after this operation, in which case the reset should be repeated. After recovery, the application should rewrite the data which was being programmed when the initial failure occurred.

### *Soft Reset Handling*

The following soft resets should not be asserted during an EEPROM program or erase operation:

- Software reset (SYSRESREQ)
- Software peripheral reset
- Watchdog reset (if configured as a system reset in the **RESBEHAVCTL** register)
- MOSC failure reset
- BOR reset (if configured as a system reset in the **RESBEHAVCTL** register)
- External reset (if configured as a system reset in the **RESBEHAVCTL** register)
- Writes to the **HSSR** register

The WORKING bit of the **EEDONE** register can be checked before the reset is asserted to see if an EEPROM program or erase operation is occurring. Soft resets may occur when using a debugger and should be avoided during an EEPROM operation. A reset such as the Watchdog reset can be mapped to an external reset using a GPIO, or Hibernate can be entered, if time is not a concern.

### *Endurance*

Endurance is per meta-block which is 8 blocks. Endurance is measured in two ways:

1. To the application, it is the number of writes that can be performed.
2. To the microcontroller, it is the number of erases that can be performed on the meta-block.

Because of the second measure, the number of writes depends on how the writes are performed. For example:

- One word can be written more than 500K times, but, these writes impact the meta-block that the word is within. As a result, writing one word 500K times, then trying to write a nearby word 500K times is not assured to work. To ensure success, the words should be written more in parallel.
- All words can be written in a sweep with a total of more than 500K sweeps which updates all words more than 500K times.
- Different words can be written such that any or all words can be written more than 500K times when write counts per word stay about the same. For example, offset 0 could be written 3 times, then offset 1 could be written 2 times, then offset 2 is written 4 times, then offset 1 is written twice, then offset 0 is written again. As a result, all 3 offsets would have 4 writes at the end of the sequence. This kind of balancing within 7 writes maximizes the endurance of different words within the same meta-block.

#### 8.2.4.2 EEPROM Initialization and Configuration

Before writing to any EEPROM registers, the clock to the EEPROM module must be enabled through the **EEPROM Run Mode Clock Gating Control (RCGCEEPROM)** register (see page 417) and the following initialization steps must be executed:

1. Insert delay (6 cycles plus function call overhead).
2. Poll the WORKING bit in the **EEPROM Done Status (EEDONE)** register until it is clear, indicating that the EEPROM has completed its power-on initialization. When WORKING=0, continue.
3. Read the PRETRY and ERETRY bits in the **EEPROM Support Control and Status (EESUPP)** register. If either of the bits are set, return an error, else continue.
4. Reset the EEPROM module using the **EEPROM Software Reset (SREEPROM)** register at offset 0x558 in the System Control register space.
5. Insert delay (6 cycles plus function call overhead).
6. Poll the WORKING bit in the **EEPROM Done Status (EEDONE)** register to determine when it is clear. When WORKING=0, continue.
7. Read the PRETRY and ERETRY bits in the **EESUPP** register. If either of the bits are set, return an error, else the EEPROM initialization is complete and software may use the peripheral as normal.

---

**Important:** Failure to perform these initialization steps after a reset may lead to incorrect operation or permanent data loss if the EEPROM is later written.

If the PRETRY or ERETRY bits are set in the **ESUPP** register, the EEPROM was unable to recover its state. If power is stable when this occurs, this indicates a fatal error and is likely an indication that the EEPROM memory has exceeded its specified lifetime write/erase specification. If the supply voltage is unstable when this return code is observed, retrying the operation once the voltage is stabilized may clear the error.

---

The EEPROM initialization function code is named EEPROMinit( ) in TivaWare, which can be downloaded from http://www.ti.com/tivaware.

### 8.2.5 Bus Matrix Memory Accesses

The following table identifies the Bus Masters and their access to the various memories on the bus matrix.

**Table 8-5. Master Memory Access Availability**

| Master | Flash Access | ROM Access | SRAM Access | EEPROM Access | External Memory Access (via EPI) |
|---|---|---|---|---|---|
| CPU Instruction Bus | Yes | Yes (read-only) | Yes | Yes | Yes |
| CPU Data Bus | Yes | Yes (read-only) | - | Yes | Yes |
| µDMA | Yes (read-only, Run-Mode-only) | - | Yes | Yes | Yes |
| Ethernet Module | - | - | Yes | - | - |
| USB | - | - | Yes | - | - |
| LCD | - | - | Yes | - | Yes |

## 8.3 Register Map

Table 8-6 on page 653 lists the ROM Controller register and the Flash memory control registers. The offset listed is a hexadecimal increment to the register's address. The Flash memory register offsets are relative to the Flash memory control base address of 0x400F.D000. The EEPROM registers are relative to the EEPROM base address of 0x400A.F000. The ROM control and Flash memory protection register offsets are relative to the System Control base address of 0x400F.E000.

**Table 8-6. Flash Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **Internal Memory Registers (Internal Memory Control Offset)** | | | | | |
| 0x000 | FMA | RW | 0x0000.0000 | Flash Memory Address | 656 |
| 0x004 | FMD | RW | 0x0000.0000 | Flash Memory Data | 657 |
| 0x008 | FMC | RW | 0x0000.0000 | Flash Memory Control | 658 |
| 0x00C | FCRIS | RO | 0x0000.0000 | Flash Controller Raw Interrupt Status | 661 |
| 0x010 | FCIM | RW | 0x0000.0000 | Flash Controller Interrupt Mask | 664 |
| 0x014 | FCMISC | RW1C | 0x0000.0000 | Flash Controller Masked Interrupt Status and Clear | 666 |
| 0x020 | FMC2 | RW | 0x0000.0000 | Flash Memory Control 2 | 669 |
| 0x030 | FWBVAL | RW | 0x0000.0000 | Flash Write Buffer Valid | 670 |
| 0x03C | FLPEKEY | RO | 0x0000.FFFF | Flash Program/Erase Key | 671 |
| 0x100 - 0x17C | FWBn | RW | 0x0000.0000 | Flash Write Buffer n | 672 |
| 0xFC0 | FLASHPP | RO | 0xF014.00FF | Flash Peripheral Properties | 673 |
| 0xFC4 | SSIZE | RO | 0x0000.03FF | SRAM Size | 675 |
| 0xFC8 | FLASHCONF | RW | 0x0000.0000 | Flash Configuration Register | 676 |
| 0xFCC | ROMSWMAP | RO | 0x0000.0000 | ROM Third-Party Software | 678 |

**Table 8-6. Flash Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xFD0 | FLASHDMASZ | RW | 0x0000.0000 | Flash DMA Address Size | 680 |
| 0xFD4 | FLASHDMAST | RW | 0x0000.0000 | Flash DMA Starting Address | 681 |
| **EEPROM Registers (EEPROM Control Offset)** | | | | | |
| 0x000 | EESIZE | RO | 0x0060.0600 | EEPROM Size Information | 682 |
| 0x004 | EEBLOCK | RW | 0x0000.0000 | EEPROM Current Block | 683 |
| 0x008 | EEOFFSET | RW | 0x0000.0000 | EEPROM Current Offset | 684 |
| 0x010 | EERDWR | RW | - | EEPROM Read-Write | 685 |
| 0x014 | EERDWRINC | RW | - | EEPROM Read-Write with Increment | 686 |
| 0x018 | EEDONE | RO | 0x0000.0000 | EEPROM Done Status | 687 |
| 0x01C | EESUPP | RW | - | EEPROM Support Control and Status | 689 |
| 0x020 | EEUNLOCK | RW | - | EEPROM Unlock | 690 |
| 0x030 | EEPROT | RW | 0x0000.0000 | EEPROM Protection | 691 |
| 0x034 | EEPASS0 | RW | - | EEPROM Password | 693 |
| 0x038 | EEPASS1 | RW | - | EEPROM Password | 693 |
| 0x03C | EEPASS2 | RW | - | EEPROM Password | 693 |
| 0x040 | EEINT | RW | 0x0000.0000 | EEPROM Interrupt | 694 |
| 0x050 | EEHIDE0 | RW | 0x0000.0000 | EEPROM Block Hide 0 | 695 |
| 0x054 | EEHIDE1 | RW | 0x0000.0000 | EEPROM Block Hide 1 | 696 |
| 0x058 | EEHIDE2 | RW | 0x0000.0000 | EEPROM Block Hide 2 | 696 |
| 0x080 | EEDBGME | RW | 0x0000.0000 | EEPROM Debug Mass Erase | 697 |
| 0xFC0 | EEPROMPP | RO | 0x0000.01FF | EEPROM Peripheral Properties | 698 |
| **Memory Registers (System Control Offset)** | | | | | |
| 0x0D4 | RVP | RO | 0x0101.FFF0 | Reset Vector Pointer | 699 |
| 0x1D0 | BOOTCFG | RO | 0xFFFF.FFFE | Boot Configuration | 704 |
| 0x1E0 | USER_REG0 | W0 | 0xFFFF.FFFF | User Register 0 | 707 |
| 0x1E4 | USER_REG1 | W0 | 0xFFFF.FFFF | User Register 1 | 707 |
| 0x1E8 | USER_REG2 | W0 | 0xFFFF.FFFF | User Register 2 | 707 |
| 0x1EC | USER_REG3 | W0 | 0xFFFF.FFFF | User Register 3 | 707 |
| 0x200 | FMPRE0 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 0 | 700 |
| 0x204 | FMPRE1 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 1 | 700 |
| 0x208 | FMPRE2 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 2 | 700 |
| 0x20C | FMPRE3 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 3 | 700 |

**Table 8-6. Flash Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x210 | FMPRE4 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 4 | 700 |
| 0x214 | FMPRE5 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 5 | 700 |
| 0x218 | FMPRE6 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 6 | 700 |
| 0x21C | FMPRE7 | RW | 0xFFFF.FFFF | Flash Memory Protection Read Enable 7 | 700 |
| 0x400 | FMPPE0 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 0 | 702 |
| 0x404 | FMPPE1 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 1 | 702 |
| 0x408 | FMPPE2 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 2 | 702 |
| 0x40C | FMPPE3 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 3 | 702 |
| 0x410 | FMPPE4 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 4 | 702 |
| 0x414 | FMPPE5 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 5 | 702 |
| 0x418 | FMPPE6 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 6 | 702 |
| 0x41C | FMPPE7 | RW | 0xFFFF.FFFF | Flash Memory Protection Program Enable 7 | 702 |

## 8.4 Internal Memory Register Descriptions (Internal Memory Control Offset)

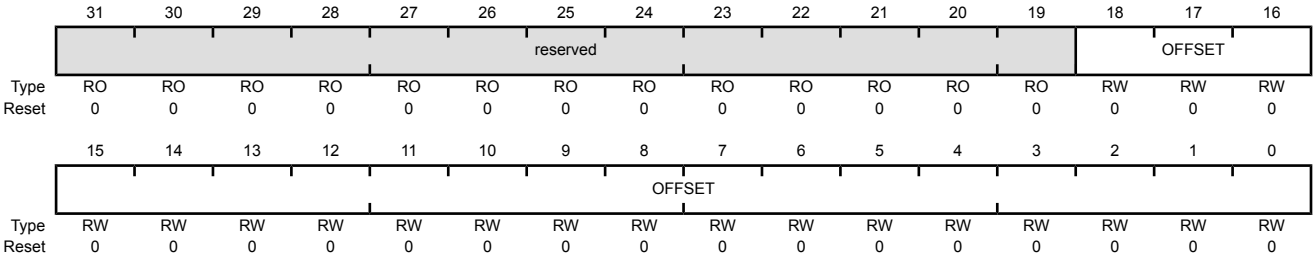
This section lists and describes the memory control registers, in numerical order by address offset. Registers in this section are relative to the memory control base address of 0x400F.D000.

## Register 1: Flash Memory Address (FMA), offset 0x000

During a write operation, this register contains a 4-byte-aligned address and specifies where the data is written. During erase operations for flash space that is not user configurable (that is, **FMPREn**, **FMPPEn**, **USER_REGn**, **BOOTCFG**), this register contains a 16 KB-aligned CPU byte address and specifies which block is erased. Note that the alignment requirements must be met by software or the results of the operation are unpredictable.

Flash Memory Address (FMA)

Base 0x400F.D000
Offset 0x000
Type RW, reset 0x0000.0000



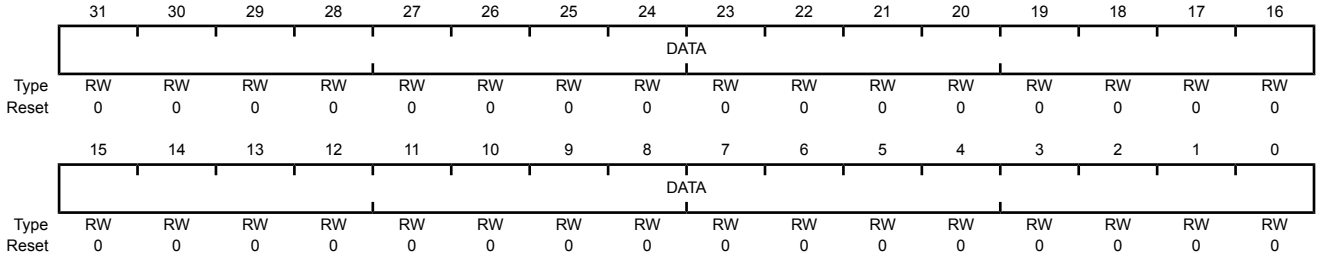

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 18:0 | OFFSET | RW | 0x0 | Address Offset<br>Address offset in Flash memory where operation is performed, except for non-volatile registers (see “Non-Volatile Register Programming--Flash Memory Resident Registers” on page 645 for details on values for this field). |

## Register 2: Flash Memory Data (FMD), offset 0x004

This register contains the data to be written during the programming cycle. Note that the contents of this register are undefined for a read access of an execute-only block. This register is not used during erase cycles.

Flash Memory Data (FMD)

Base 0x400F.D000
Offset 0x004
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RW | 0x0000.0000 | Data Value<br>Data value for write operation. |

## Register 3: Flash Memory Control (FMC), offset 0x008

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 656). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 657) is written to the specified address.

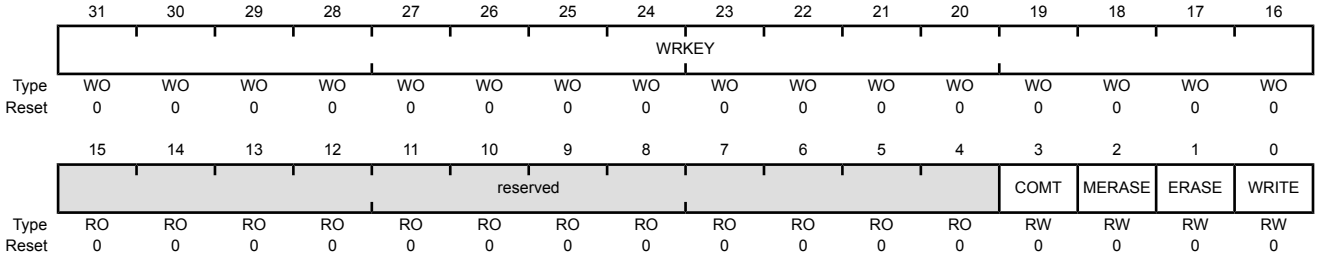
For non-volatile registers, **FMPREn**, **FMPPEn**, **USER_REGn**, and USER_REGn, the respective register is programmed with the value to be written rather than the **FMD** register.

This register must be the final register written and initiates the memory operation. The four control bits in the lower byte of this register are used to initiate memory operations.

Care must be taken not to set multiple control bits as the results of such an operation are unpredictable.

Flash Memory Control (FMC)

Base 0x400F.D000
Offset 0x008
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | WRKEY | WO | 0x0000 | Flash Memory Write Key<br>This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 or the PEKEY value in the **FLPEKEY** register must be written into this field for a Flash memory write to occur. The use of 0xA442 or PEKEY is dependent on the value of the KEY bit in the **BOOTCFG** register at 0x1D0.<br>Writes to the **FMC** register without this WRKEY value are ignored. A read of this field returns the value 0. |
| 15:4 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | COMT | RW | 0 | Commit Register Value<br>This bit is used to commit writes to Flash-memory-resident registers and to monitor the progress of that process.<br><br>Value Description<br>0 A write of 0 has no effect on the state of this bit.<br>When read, a 0 indicates that the previous commit access is complete.<br>1 Set this bit to commit (write) the register value to a Flash-memory-resident register.<br>When read, a 1 indicates that the previous commit access is not complete.<br><br>See “Non-Volatile Register Programming-- Flash Memory Resident Registers” on page 645 for more information on programming Flash-memory-resident registers. |
| 2 | MERASE | RW | 0 | Mass Erase Flash Memory<br>This bit is used to mass erase the Flash main memory and to monitor the progress of that process.<br><br>Value Description<br>0 A write of 0 has no effect on the state of this bit.<br>When read, a 0 indicates that the previous mass erase access is complete.<br>1 Set this bit to erase the Flash main memory.<br>When read, a 1 indicates that the previous mass erase access is not complete.<br><br>For information on erase time, see “Flash Memory” on page 2124. |
| 1 | ERASE | RW | 0 | Erase a Page of Flash Memory<br>This bit is used to erase a page of Flash memory and to monitor the progress of that process.<br><br>Value Description<br>0 A write of 0 has no effect on the state of this bit.<br>When read, a 0 indicates that the previous page erase access is complete.<br>1 Set this bit to erase the Flash memory page specified by the contents of the **FMA** register.<br>When read, a 1 indicates that the previous page erase access is not complete.<br><br>For information on erase time, see “Flash Memory” on page 2124. |

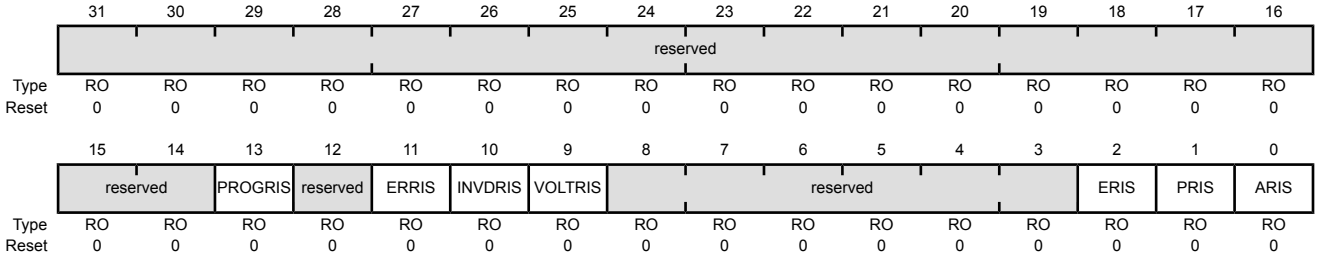
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | WRITE | RW | 0 | Write a Word into Flash Memory<br>This bit is used to write a word into Flash memory and to monitor the progress of that process.<br><br>Value Description<br>0 A write of 0 has no effect on the state of this bit.<br>When read, a 0 indicates that the previous write update access is complete.<br>1 Set this bit to write the data stored in the **FMD** register into the Flash memory location specified by the contents of the **FMA** register.<br>When read, a 1 indicates that the write update access is not complete.<br><br>For information on programming time, see “Flash Memory” on page 2124. |

## Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C

This register indicates that the Flash memory controller has an interrupt condition. An interrupt is sent to the interrupt controller only if the corresponding **FCIM** register bit is set.

Flash Controller Raw Interrupt Status (FCRIS)

Base 0x400F.D000
Offset 0x00C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000.000 | 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. |
| 13 | PROGRIS | RO | 0 | Program Verify Error Raw Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 An interrupt is pending because the verify of a PROGRAM operation failed. If this error occurs when using the Flash write buffer, software must inspect the affected words to determine where the error occurred.<br><br>This bit is cleared by writing a 1 to the PROGMISC bit in the **FCMISC** register. |
| 12 | reserved | RO | 0 | 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. |
| 11 | ERRIS | RO | 0 | Erase Verify Error Raw Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 An interrupt is pending because the verify of an ERASE operation failed. If this error occurs when using the Flash write buffer, software must inspect the affected words to determine where the error occurred.<br><br>This bit is cleared by writing a 1 to the ERMISC bit in the **FCMISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | INVDRIS | RO | 0 | Invalid Data Raw Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 An interrupt is pending because a bit that was previously programmed as a 0 is now being requested to be programmed as a 1.<br><br>This bit is cleared by writing a 1 to the INVMISC bit in the **FCMISC** register. |
| 9 | VOLTRIS | RO | 0 | Pump Voltage Raw Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 An interrupt is pending because the regulated voltage of the pump went out of spec during the Flash operation and the operation was terminated.<br><br>This bit is cleared by writing a 1 to the VOLTMISC bit in the **FCMISC** register. |
| 8:3 | reserved | RO | 0 | 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. |
| 2 | ERIS | RO | 0 | EEPROM Raw Interrupt Status<br>This bit provides status EEPROM operation.<br><br>Value Description<br>0 An EEPROM interrupt has not occurred.<br>1 An EEPROM interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the EMISC bit in the **FCMISC** register. |
| 1 | PRIS | RO | 0 | Programming Raw Interrupt Status<br>This bit provides status on programming cycles which are write or erase actions generated through the **FMC** or **FMC2** register bits (see page 658 and page 669).<br><br>Value Description<br>0 The programming or erase cycle has not completed.<br>1 The programming or erase cycle has completed.<br><br>This status is sent to the interrupt controller when the PMASK bit in the **FCIM** register is set.<br>This bit is cleared by writing a 1 to the PMISC bit in the **FCMISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ARIS | RO | 0 | Access Raw Interrupt Status |

| Value | Description |
|---|---|
| 0 | No access has tried to improperly program or erase the Flash memory. |
| 1 | A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the **FMPPEn** registers. |

This status is sent to the interrupt controller when the AMASK bit in the **FCIM** register is set.

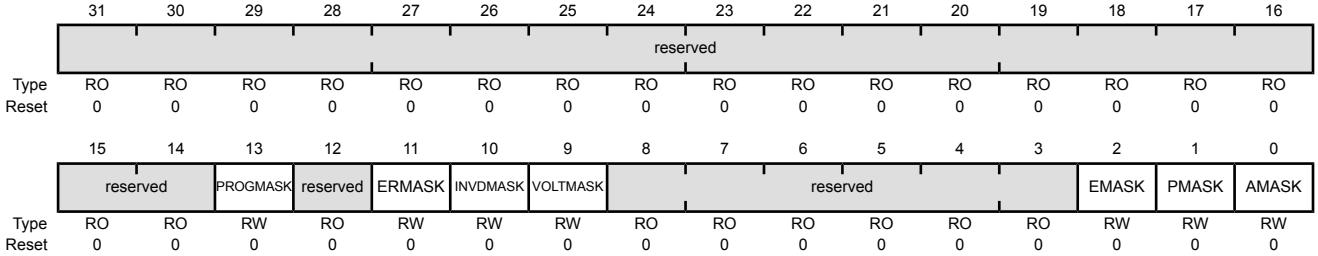
This bit is cleared by writing a 1 to the AMISC bit in the **FCMISC** register.

## Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010

This register controls whether the Flash memory controller generates interrupts to the controller.

Flash Controller Interrupt Mask (FCIM)

Base 0x400F.D000
Offset 0x010
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000.000 | 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. |
| 13 | PROGMASK | RW | 0 | Program Verify Error Interrupt Mask<br>Value Description<br>0 The PROGRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PROGRIS bit is set. |
| 12 | reserved | RO | 0 | 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. |
| 11 | ERMASK | RW | 0 | Erase Verify Error Interrupt Mask<br>Value Description<br>0 The ERRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the ERRIS bit is set. |
| 10 | INVDMASK | RW | 0 | Invalid Data Interrupt Mask<br>Value Description<br>0 The INVDRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the INVDRIS bit is set. |

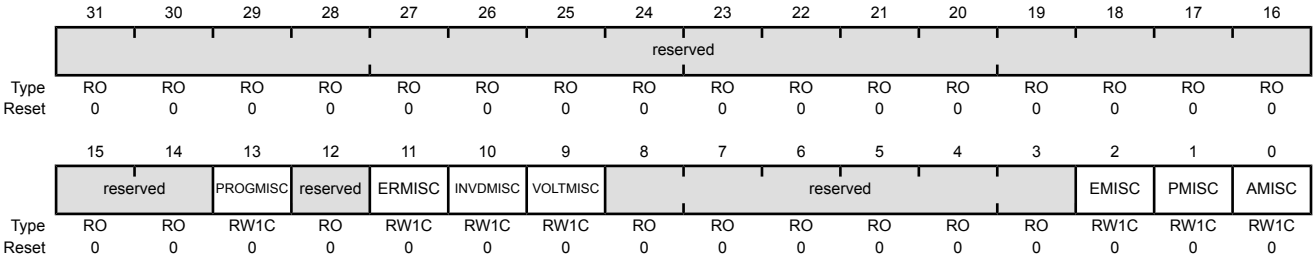
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | VOLTMASK | RW | 0 | Pump Voltage Interrupt Mask<br>Value Description<br>0 The VOLTRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the VOLTRIS bit is set. |
| 8:3 | reserved | RO | 0 | 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. |
| 2 | EMASK | RW | 0 | EEPROM Interrupt Mask<br>Value Description<br>0 The ERIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the ERIS bit is set. |
| 1 | PMASK | RW | 0 | Programming Interrupt Mask<br>This bit controls the reporting of the programming raw interrupt status to the interrupt controller.<br>Value Description<br>0 The PRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PRIS bit is set. |
| 0 | AMASK | RW | 0 | Access Interrupt Mask<br>This bit controls the reporting of the access raw interrupt status to the interrupt controller.<br>Value Description<br>0 The ARIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the ARIS bit is set. |

## Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014

This register provides two functions. First, it reports the cause of an interrupt by indicating which interrupt source or sources are signalling the interrupt. Second, it serves as the method to clear the interrupt reporting.

Flash Controller Masked Interrupt Status and Clear (FCMISC)

Base 0x400F.D000
Offset 0x014
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000.000 | 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. |
| 13 | PROGMISC | RW1C | 0 | PROGVER Masked Interrupt Status and Clear<br><br>Value Description<br>0 When read, a 0 indicates that an interrupt has not occurred.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled.<br>Writing a 1 to this bit clears `PROGMISC` and also the `PROGRIS` bit in the **FCRIS** register (see page 661). |
| 12 | reserved | RO | 0 | 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. |
| 11 | ERMISC | RW1C | 0 | ERVER Masked Interrupt Status and Clear<br><br>Value Description<br>0 When read, a 0 indicates that an interrupt has not occurred.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled.<br>Writing a 1 to this bit clears `ERMISC` and also the `ERRIS` bit in the **FCRIS** register (see page 661). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | INVDMISC | RW1C | 0 | Invalid Data Masked Interrupt Status and Clear<br><br>Value Description<br>0 When read, a 0 indicates that an interrupt has not occurred.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled.<br>Writing a 1 to this bit clears INVDMISC and also the INVDRIS bit in the **FCRIS** register (see page 661). |
| 9 | VOLTMISC | RW1C | 0 | VOLT Masked Interrupt Status and Clear<br><br>Value Description<br>0 When read, a 0 indicates that an interrupt has not occurred.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled.<br>Writing a 1 to this bit clears VOLTMISC and also the VOLTRIS bit in the **FCRIS** register (see page 661). |
| 8:3 | reserved | RO | 0 | 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. |
| 2 | EMISC | RW1C | 0 | EEPROM Masked Interrupt Status and Clear<br><br>Value Description<br>0 When read, a 0 indicates that an interrupt has not occurred.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled.<br>Writing a 1 to this bit clears EMISC and also the ERIS bit in the **FCRIS** register (see page 661). |
| 1 | PMISC | RW1C | 0 | Programming Masked Interrupt Status and Clear<br><br>Value Description<br>0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred.<br>A write of 0 has no effect on the state of this bit.<br>1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed.<br>Writing a 1 to this bit clears PMISC and also the PRIS bit in the **FCRIS** register (see page 661). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | AMISC | RW1C | 0 | Access Masked Interrupt Status and Clear |

| Value | Description |
|---|---|
| 0 | When read, a 0 indicates that no improper accesses have occurred.<br>A write of 0 has no effect on the state of this bit. |
| 1 | When read, a 1 indicates that an unmasked interrupt was signaled because a program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the **FMPPEn** registers.<br>Writing a 1 to this bit clears `AMISC` and also the `ARIS` bit in the **FCRIS** register (see page 661). |

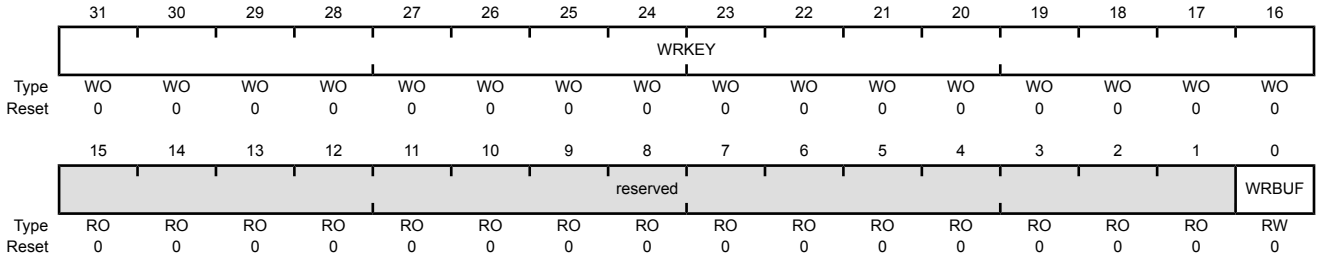
## Register 7: Flash Memory Control 2 (FMC2), offset 0x020

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 656). If the access is a write access, the data contained in the **Flash Write Buffer (FWB)** registers is written.

This register must be the final register written as it initiates the memory operation.

Flash Memory Control 2 (FMC2)

Base 0x400F.D000
Offset 0x020
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | WRKEY | WO | 0x0000 | Flash Memory Write Key<br>This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. There are two options for the WRKEY value:<br>If the KEY value in the **BOOTCFG** register is 0x1 at reset, the value 0xA442 is used as a key enable to initiate the appropriate access cycle for the location specified by the address in the **FMA** register.<br>If the KEY value in the **BOOTCFG** register is 0x0 at reset, the value programmed in the **FLPEKEY** register is used as a key enable to initiate the appropriate access cycle for the location specified by the address in the **FMA** register.<br>Writes to the **FMC2** register without this WRKEY value are ignored. A read of this field returns the value 0. |
| 15:1 | reserved | RO | 0x000 | 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. |
| 0 | WRBUF | RW | 0 | Buffered Flash Memory Write<br>This bit is used to start a buffered write to Flash memory.<br>Value Description<br>0 A write of 0 has no effect on the state of this bit.<br>When read, a 0 indicates that the previous buffered Flash memory write access is complete.<br>1 Set this bit to write the data stored in the **FWBn** registers to the location specified by the contents of the **FMA** register.<br>When read, a 1 indicates that the previous buffered Flash memory write access is not complete.<br>For information on programming time, see “Flash Memory” on page 2124. |

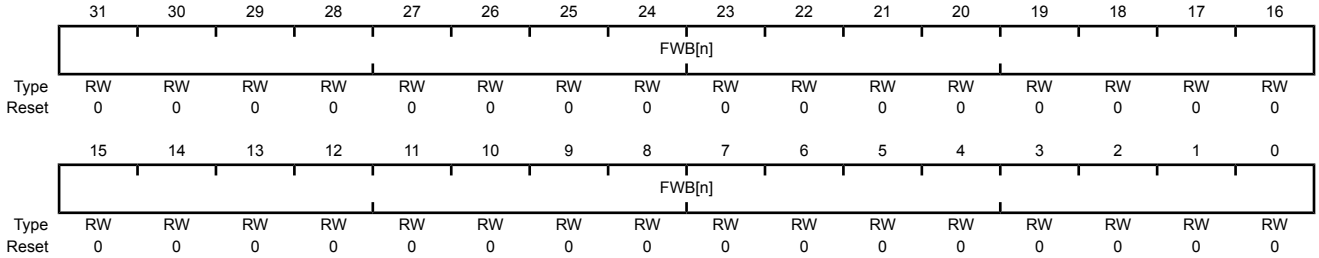
## Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030

This register provides a bitwise status of which **FWBn** registers have been written by the processor since the last write of the Flash memory write buffer. The entries with a 1 are written on the next write of the Flash memory write buffer. This register is cleared after the write operation by hardware. A protection violation on the write operation also clears this status.

Software can program the same 32 words to various Flash memory locations by setting the FWB[n] bits after they are cleared by the write operation. The next write operation then uses the same data as the previous one. In addition, if a **FWBn** register change should not be written to Flash memory, software can clear the corresponding FWB[n] bit to preserve the existing data when the next write operation occurs.

Flash Write Buffer Valid (FWBVAL)

Base 0x400F.D000
Offset 0x030
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | FWB[n] | RW | 0x0 | Flash Memory Write Buffer |

| Value | Description |
|---|---|
| 0 | The corresponding **FWBn** register has no new data to be written. |
| 1 | The corresponding **FWBn** register has been updated since the last buffer write operation and is ready to be written to Flash memory. |

Bit 0 corresponds to **FWB0**, offset 0x100, and bit 31 corresponds to **FWB31**, offset 0x13C.

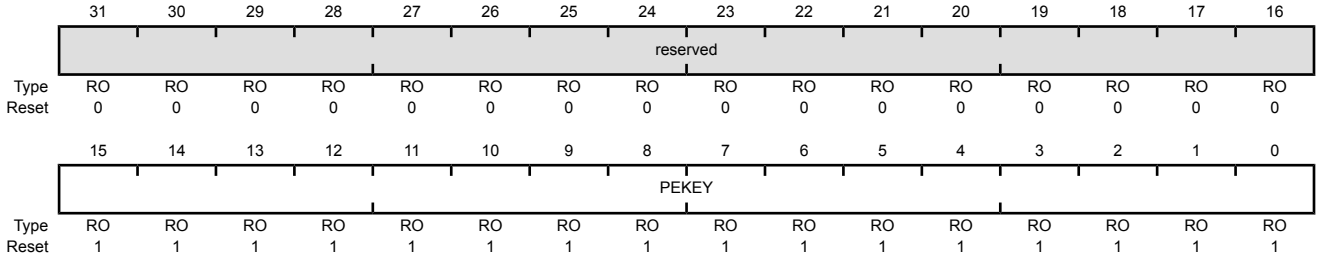
## Register 9: Flash Program/Erase Key (FLPEKEY), offset 0x03C

This register provides a mechanism for protection from inadvertent writes to flash by supplying a 16-bit key . If the KEY value in the **BOOTCFG** register is 0, then this value is used as the 16-bit key in place of 0xA442 in the **FMC/FMC2** registers for committed flash writes.

This can be used for cases where a new image is downloaded and the first word of the new image has the 16-bit key value to be used for that product. This 16-bit key is used to allow the write to **FMC** or **FMC2** to take place.

Flash Program/Erase Key (FLPEKEY)

Base 0x400F.D000
Offset 0x03C
Type RO, reset 0x0000.FFFF



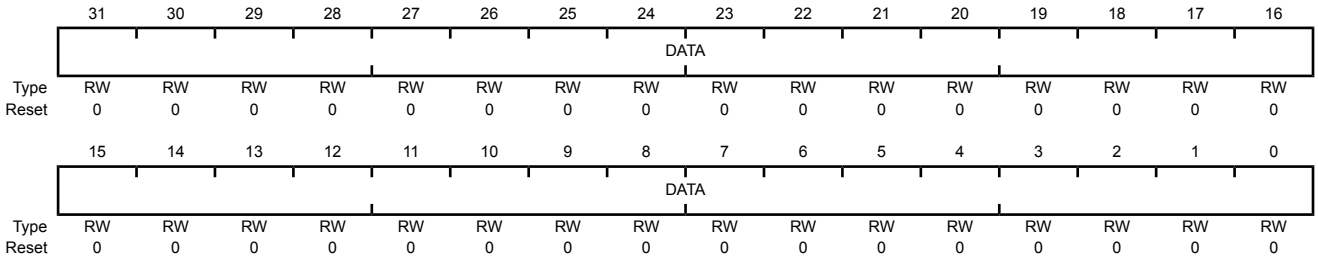

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | PEKEY | RO | 0xFFFF | Key Value<br>When a value other than all 1s or all 0s, this 16-bit value is used as the "match" for the upper 16-bits of the register **FMC** and **FMC2** keys. |

## Register 10: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C

These 32 registers hold the contents of the data to be written into the Flash memory on a buffered Flash memory write operation. The offset selects one of the 32-bit registers. Only **FWBn** registers that have been updated since the preceding buffered Flash memory write operation are written into the Flash memory, so it is not necessary to write the entire bank of registers in order to write 1 or 2 words. The **FWBn** registers are written into the Flash memory with the **FWB0** register corresponding to the address contained in **FMA**. **FWB1** is written to the address **FMA**+0x4 etc. Note that only data bits that are 0 result in the Flash memory being modified. A data bit that is 1 leaves the content of the Flash memory bit at its previous value.

Flash Write Buffer n (FWBn)

Base 0x400F.D000
Offset 0x100 - 0x17C
Type RW, reset 0x0000.0000

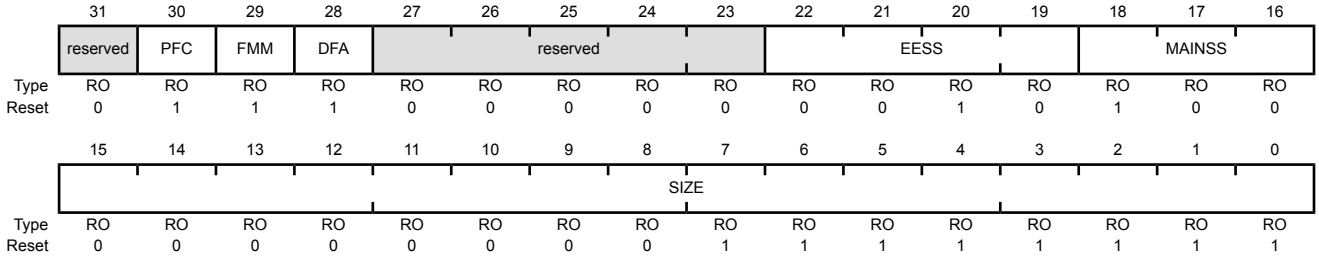



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RW | 0x0000.0000 | Data<br>Data to be written into the Flash memory. |

## Register 11: Flash Peripheral Properties (FLASHPP), offset 0xFC0

Flash Peripheral Properties (FLASHPP)

Base 0x400F.D000
Offset 0xFC0
Type RO, reset 0xF014.00FF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | reserved | RO | 0 | 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. |
| 30 | PFC | RO | 0x1 | Prefetch Buffer Mode<br><br>Value Description<br>0 Single set of 2x256-bit buffers used.<br>1 Two sets of 2x256-bit prefetch buffers are available to use and may be enabled through the **FLASHCONF** register. |
| 29 | FMM | RO | 0x1 | Flash Mirror Mode<br><br>Value Description<br>0 Mirror Mode not available.<br>1 Flash Mirror Mode is available to be enabled or disabled by user through **FLASHCONF** register. |
| 28 | DFA | RO | 0x1 | DMA Flash Access<br><br>**Note:** µDMA can only access flash in Run Mode (not available in low power modes).<br><br>Value Description<br>0 DMA cannot be used to access Flash<br>1 DMA may access the Flash memory range specified by the **FLASHDMAST** and **FLASHDMASZ** registers |
| 27:23 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 22:19 | EESS | RO | 0x2 | EEPROM Sector Size of the physical bank<br><br>Value Description<br>0x0 1 KB<br>0x1 2 KB<br>0x2 4 KB<br>0x3 8 KB<br>0x4-0x7 reserved |
| 18:16 | MAINSS | RO | 0x4 | Flash Sector Size of the physical bank<br><br>Value Description<br>0x0 1 KB<br>0x1 2 KB<br>0x2 4 KB<br>0x3 8 KB<br>0x4 16 KB<br>0x5-0x7 reserved |
| 15:0 | SIZE | RO | 0xFF | Flash Size<br>Indicates the size of the on-chip Flash memory<br><br>Value Description<br>0x00FF 512 KB of Flash |

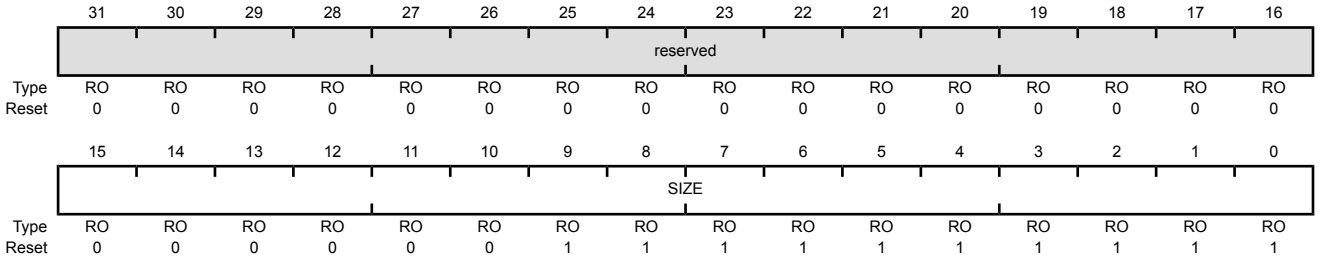
## Register 12: SRAM Size (SSIZE), offset 0xFC4

This register indicates the size of the on-chip SRAM.

**Important:** This register should be used to determine the size of the SRAM that is implemented on this microcontroller.

SRAM Size (SSIZE)

Base 0x400F.D000
Offset 0xFC4
Type RO, reset 0x0000.03FF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | 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. |
| 15:0 | SIZE | RO | 0x3FF | SRAM Size<br>Indicates the size of the on-chip SRAM.<br><br>Value Description<br>0x03FF 256 KB of SRAM |

## Register 13: Flash Configuration Register (FLASHCONF), offset 0xFC8

The **FLASHCONF** register allows the user to enable or disable various properties of the Flash. The force bits, FBF0N and FBF0FF, can be used to test code performance and execution by turning the prefetch buffers on and subsequently forcing them off.

Flash Configuration Register (FLASHCONF)

Base 0x400F.D000
Offset 0xFC8
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | FMME | SPFE | reserved | | | | | | | | CLRTV | reserved | | FPFON | FPFOFF |
| Type | RO | RW | RW | RO | RO | RO | RO | RO | RO | RO | RO | RW | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | reserved | RO | 0 | 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. |
| 30 | FMME | RW | 0x0 | Flash Mirror Mode Enable<br><br>Value Description<br>0 Flash mirror mode is disabled.<br>1 Flash mirror mode feature is enabled. Access to the lower banks is translated to upper. |
| 29 | SPFE | RW | 0x0 | Single Prefetch Mode Enable<br><br>Value Description<br>0 A 4x256-bit prefetch buffer is enabled and used.<br>1 A single 2x256-bit prefetch buffer is enabled and used. |
| 28:21 | reserved | RO | 0x000 | 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. |
| 20 | CLRTV | RW | 0 | Clear Valid Tags<br>This is a self-clearing bit.<br><br>Value Description<br>0 No effect.<br>1 Clear valid tags in the prefetch buffer. |
| 19:18 | reserved | RO | 0 | 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. |

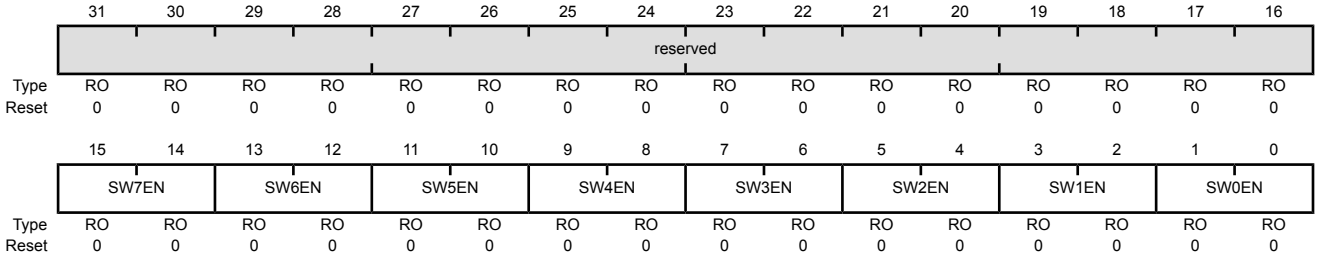
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | FPFON | RW | 0 | Force Prefetch On<br><br>Value Description<br>0 No effect<br>1 Force prefetch buffers to be enabled. |
| 16 | FPFOFF | RW | 0 | Force Prefetch Off<br><br>Value Description<br>0 No effect<br>1 Force prefetch buffers to be disabled. |
| 15:0 | reserved | RO | 0x0000 | 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. |

## Register 14: ROM Third-Party Software (ROMSWMAP), offset 0xFCC

This register indicates the presence of third-party software in the on-chip ROM. ROMSWMAP enables the ROM apertures that are available.

ROM Third-Party Software (ROMSWMAP)

Base 0x400F.D000
Offset 0xFCC
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | 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. |
| 15:14 | SW7EN | RO | 0x0 | ROM SW Region 7 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |
| 13:12 | SW6EN | RO | 0x0 | ROM SW Region 6 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |
| 11:10 | SW5EN | RO | 0x0 | ROM SW Region 5 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |
| 9:8 | SW4EN | RO | 0x0 | ROM SW Region 4 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | SW3EN | RO | 0x0 | ROM SW Region 3 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |
| 5:4 | SW2EN | RO | 0x0 | ROM SW Region 2 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |
| 3:2 | SW1EN | RO | 0x0 | ROM SW Region 1 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |
| 1:0 | SW0EN | RO | 0x0 | ROM SW Region 0 Availability<br>Value Description<br>0x0 Software region not available to the core.<br>0x1 Region available to core<br>0x2-0x3 reserved |

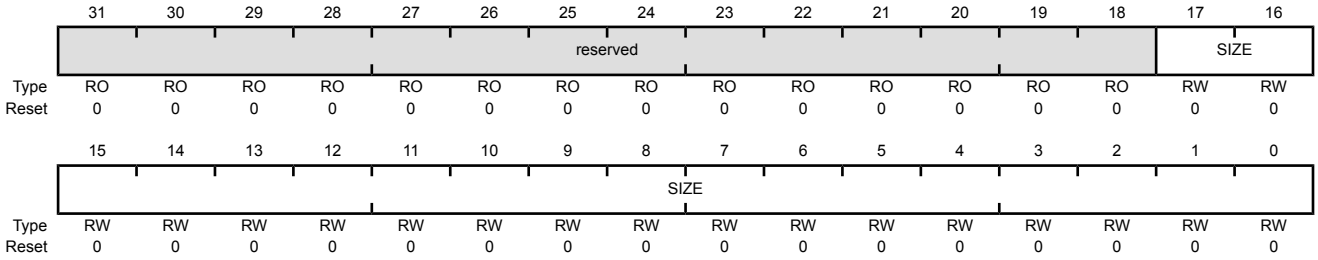
## Register 15: Flash DMA Address Size (FLASHDMASZ), offset 0xFD0

The **FLASHDMASZ** register contains the area of Flash that the µDMA can access.

**Note:** The µDMA can access Flash in Run Mode only (not available in low power modes).

Flash DMA Address Size (FLASHDMASZ)

Base 0x400F.D000
Offset 0xFD0
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0x0000 | 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. |
| 17:0 | SIZE | RW | 0x0 | µDMA-accessible Memory Size<br>The size of the region addressable by the µDMA. Note that the DFA bit must be set in the **FLASHPP** register before this value can be programmed. Size of region is defined as 2*(SIZE + 1) KB. |

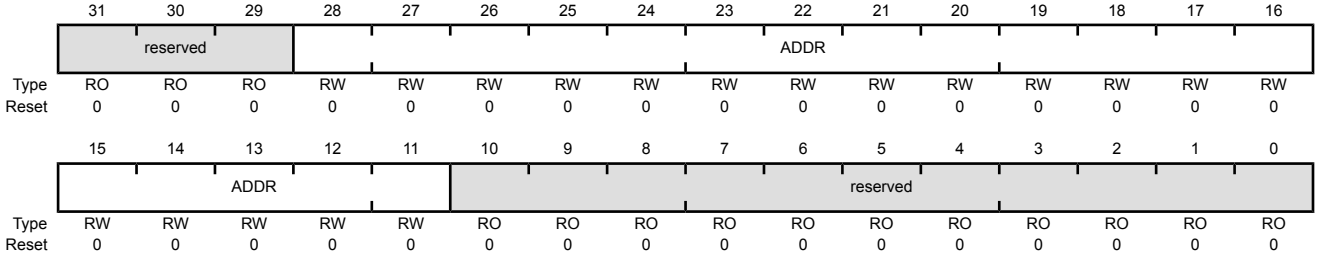
### Register 16: Flash DMA Starting Address (FLASHDMAST), offset 0xFD4

The starting address for the Flash region accessible by the µDMA is programmed in the **FLASHDMAST** register.

**Note:** The µDMA can access Flash in Run Mode only (not available in low power modes).

Flash DMA Starting Address (FLASHDMAST)

Base 0x400F.D000
Offset 0xFD4
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0x0 | 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. |
| 28:11 | ADDR | RW | 0x0 | Contains the starting address of the flash region accessible by µDMA if the **FLASHPP** register DFA bit is set |
| 10:0 | reserved | RO | 0x0 | 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. |

## 8.5 EEPROM Register Descriptions (EEPROM Offset)

This section lists and describes the EEPROM registers, in numerical order by address offset. Registers in this section are relative to the EEPROM base address of 0x400A.F000.

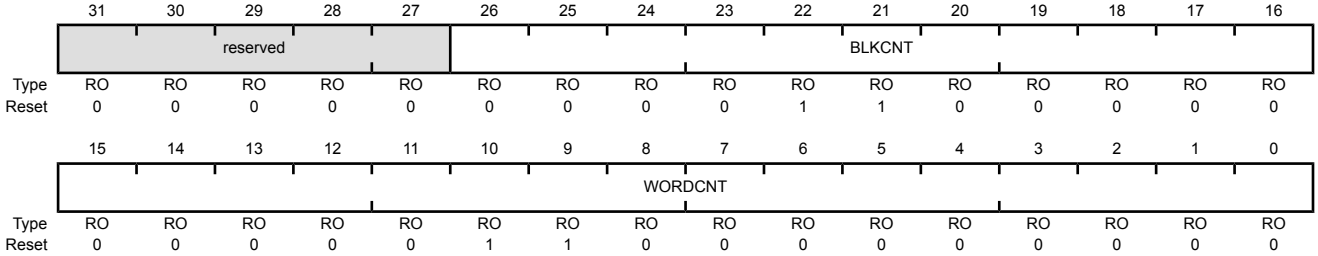
Note that the EEPROM module clock must be enabled before the registers can be programmed (see page 417). There must be a delay of 3 system clocks after the EEPROM module clock is enabled before any EEPROM module registers are accessed. In addition, after enabling or resetting the EEPROM module, software must wait until the WORKING bit in the **EEDONE** register is clear before accessing any EEPROM registers.

## Register 17: EEPROM Size Information (EESIZE), offset 0x000

The **EESIZE** register indicates the number of 16-word blocks and 32-bit words in the EEPROM.

EEPROM Size Information (EESIZE)

Base 0x400A.F000
Offset 0x000
Type RO, reset 0x0060.0600



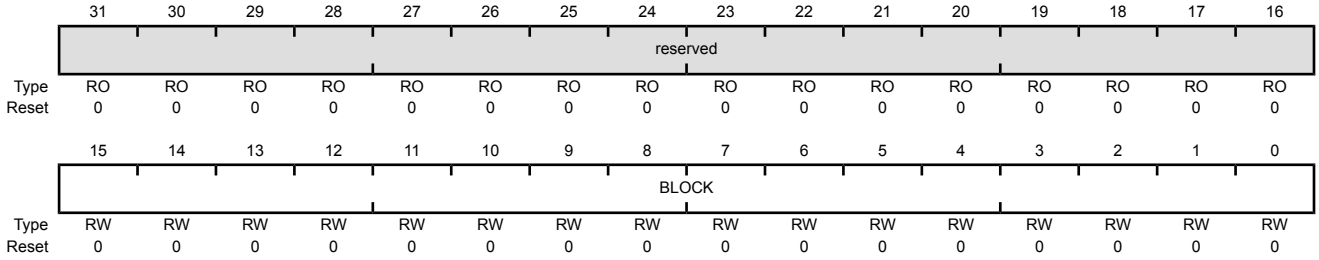

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:27 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 26:16 | BLKCNT | RO | 0x60 | Number of 16-Word Blocks<br>This value encoded in this field describes the number of 16-word blocks in the EEPROM. |
| 15:0 | WORDCNT | RO | 0x600 | Number of 32-Bit Words<br>This value encoded in this field describes the number of 32-bit words in the EEPROM. |

## Register 18: EEPROM Current Block (EEBLOCK), offset 0x004

The **EEBLOCK** register is used to select the EEPROM block for subsequent reads, writes, and protection control. The value is a page offset into the EEPROM, such that the first block is 0, then second block is 1, etc. Each block contains 16 words. Attempts to set an invalid block causes the BLOCK field to be configured to 0. To verify that the intended block is being accessed, software can read the BLOCK field after it has been written. An invalid block can be either a non-existent block or a block that has been hidden using the **EEHIDE** register. Note that block 0 cannot be hidden.

EEPROM Current Block (EEBLOCK)

Base 0x400A.F000
Offset 0x004
Type RW, reset 0x0000.0000



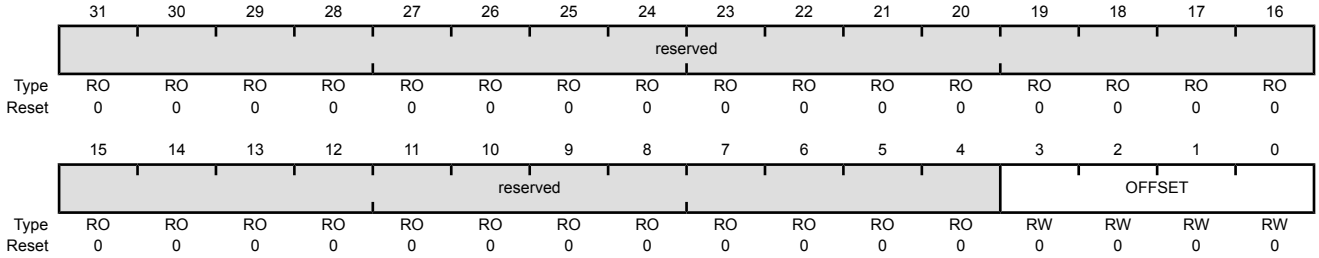

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x00000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | BLOCK | RW | 0x0000 | Current Block<br>This field specifies the block in the EEPROM that is selected for subsequent accesses. Once this field is configured, the read-write registers operate against the specified block, using the **EEOFFSET** register to select the word within the block. Additionally, the protection and unlock registers are used for the selected block. The maximum value that can be written into this register is determined by the block count, as indicated by the **EESIZE** register. Attempts to write this field larger than the maximum number of blocks or to a locked block causes this field to be configured to 0. |

## Register 19: EEPROM Current Offset (EEOFFSET), offset 0x008

The **EEOFFSET** register is used to select the EEPROM word to read or write within the block selected by the **EEBLOCK** register. The value is a word offset into the block. Because accesses to the **EERDWRINC** register change the offset, software can read the contents of this register to determine the current offset.

EEPROM Current Offset (EEOFFSET)

Base 0x400A.F000
Offset 0x008
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | OFFSET | RW | 0x0 | Current Address Offset<br>This value is the current address specified as an offset into the block selected by the **EEBLOCK** register. Once configured, the read-write registers, **EERDRWR** and **EERDWRINC**, operate against that address. The offset is automatically incremented by the **EERDWRINC** register, with wrap around within the block, which means the offset is incremented from 15 back to 0. |

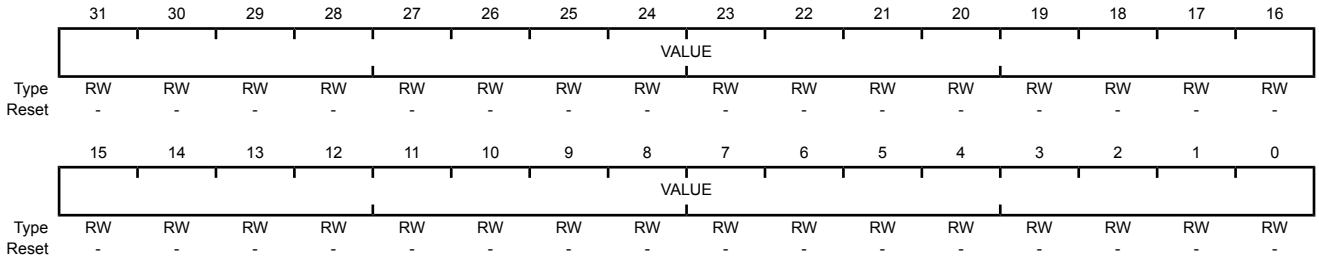
## Register 20: EEPROM Read-Write (EERDWR), offset 0x010

The **EERDWR** register is used to read or write the EEPROM word at the address pointed to by the **EEBLOCK** and **EEOFFSET** registers. If the protection or access rules do not permit access, the operation is handled as follows: if reading is not allowed, the value 0xFFFF.FFFF is returned in all cases; if writing is not allowed, the **EEDONE** register is configured to indicate an error.

**Note:** A read of the **EERDWR** register during the EEPROM initialization sequence is only valid when the `WORKING` bit is 0 in **EEDONE** register:

EEPROM Read-Write (EERDWR)

Base 0x400A.F000
Offset 0x010
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | VALUE | RW | - | EEPROM Read or Write Data<br>On a read, this field contains the value at the word pointed to by **EEOFFSET**. On a write, this field contains the data to be stored at the word pointed to by **EEOFFSET**. For writes, configuring this field starts the write process. If protection and access rules do not permit reads, all 1s are returned. If protection and access rules do not permit writes, the write fails and the **EEDONE** register indicates failure. |

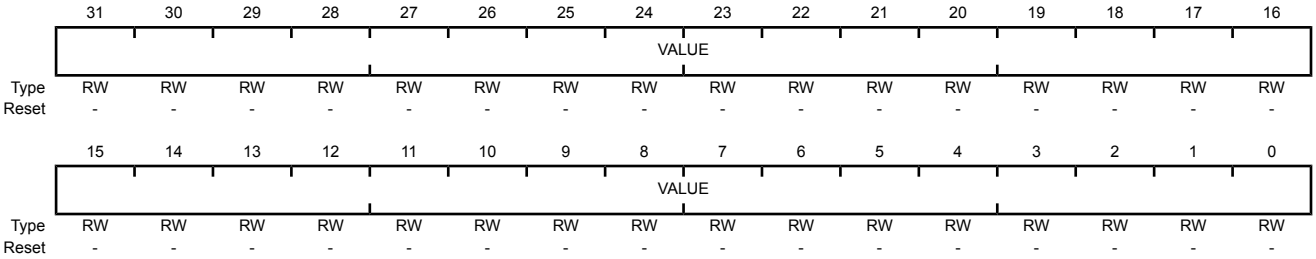
## Register 21: EEPROM Read-Write with Increment (EERDWRINC), offset 0x014

The **EERDWRINC** register is used to read or write the EEPROM word at the address pointed to by the **EEBLOCK** and **EEOFFSET** registers, and then increment the OFFSET field in the **EEOFFSET** register. If the protection or access rules do not permit access, the operation is handled as follows: if reading is not allowed, the value 0xFFFF.FFFF is returned in all cases; if writing is not allowed, the **EEDONE** register is configured to indicate an error. In any case, the OFFSET field is incremented. If the last value is reached, OFFSET wraps around to 0 and points to the first word.

**Note:** A read of the **EERDWRINC** register during the EEPROM initialization sequence is only valid when the WORKING bit is 0 in **EEDONE** register:

EEPROM Read-Write with Increment (EERDWRINC)

Base 0x400A.F000
Offset 0x014
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | VALUE | RW | - | EEPROM Read or Write Data with Increment<br>On a read, this field contains the value at the word pointed to by **EEOFFSET**. On a write, this field contains the data to be stored at the word pointed to by **EEOFFSET**. For writes, configuring this field starts the write process. If protection and access rules do not permit reads, all 1s are returned. If protection and access rules do not permit writes, the write fails and the **EEDONE** register indicates failure.<br>Regardless of error, the OFFSET field in the **EEOFFSET** register is incremented by 1, and the value wraps around if the last word is reached. |

## Register 22: EEPROM Done Status (EEDONE), offset 0x018

The **EEDONE** register indicates completion status of a write to the following registers:

- **EERDWR** or **EERDWRINC** register (for writes to the EEPROM memory)
- **EEPROT** register (for setting read and protection of the current block)
- **EEPASSn** registers (for configuring a password for a block)
- **EEDBGME** register (for mass erase of an EEPROM block)

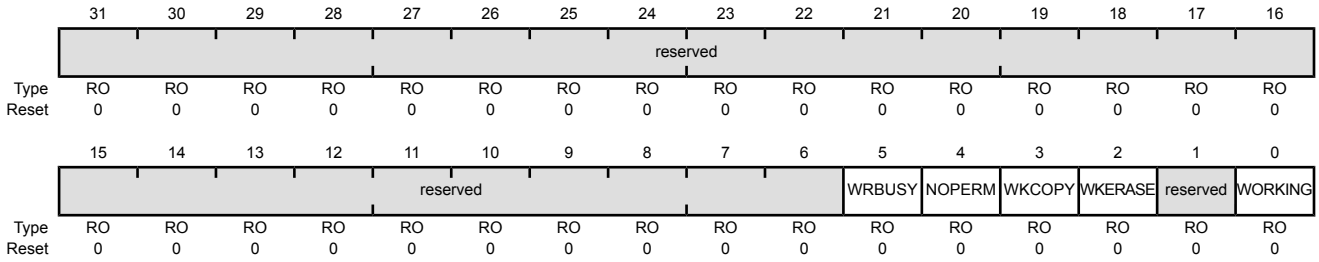
This register can indicate if the write ended in an error or not. The **EEDONE** register can be used in conjunction with the **EEINT** register to be indicate completion. The register can be **EEDONE** polled or read after an **EEINT** register interrupt fires. If any of the bit values in the **EEDONE** register are 1 after completion, then an error has occurred for that register write. If all of the bits are clear then the writes completed with success.

**Note:** Reads of the following registers during the EEPROM initialization sequence are only valid when the `WORKING` bit is 0 in **EEDONE** register:

- **EERDWR** or **EERDWRINC**
- **EEPROT**
- **EEPASSn**

EEPROM Done Status (EEDONE)

Base 0x400A.F000
Offset 0x018
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.0 | 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 | WRBUSY | RO | 0 | Write Busy<br>Value Description<br>0 No error<br>1 An attempt to access the EEPROM was made while a write was in progress. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | NOPERM | RO | 0 | Write Without Permission<br><br>Value Description<br>0 No error<br>1 An attempt was made to write without permission. This error can result because the block is locked, the write violates the programmed access protection, or when an attempt is made to write a password when the password has already been written. |
| 3 | WKCOPY | RO | 0 | Working on a Copy<br><br>Value Description<br>0 The EEPROM is not copying.<br>1 A write is in progress and is waiting for the EEPROM to copy to or from the copy buffer. |
| 2 | WKERASE | RO | 0 | Working on an Erase<br><br>Value Description<br>0 The EEPROM is not erasing.<br>1 A write is in progress and the original block is being erased after being copied. |
| 1 | reserved | RO | 0 | 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. |
| 0 | WORKING | RO | 0 | EEPROM Working<br><br>Value Description<br>0 The EEPROM is not working.<br>1 The EEPROM is performing the requested operation. |

## Register 23: EEPROM Support Control and Status (EESUPP), offset 0x01C

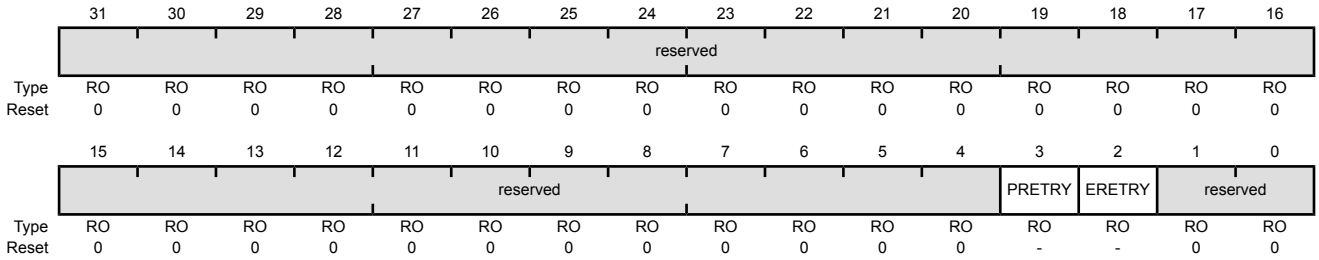
The **EESUPP** register indicates if internal operations are required because an internal copy buffer must be erased or a programming failure has occurred and the operation must be completed. These conditions are explained below as well as in more detail in the section called “Error During Programming” on page 650.

- If either PRETRY or ERETRY is set indicating that an operation must be completed, setting the START bit causes the operation to be performed again
- The PRETRY and ERETRY bits are cleared automatically after the failed operation has been successfully completed.

These bits are not changed by reset, so any condition that occurred before a reset is still indicated after a reset.

EEPROM Support Control and Status (EESUPP)

Base 0x400A.F000
Offset 0x01C
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | PRETRY | RO | - | Programming Must Be Retried<br>Value Description<br>0 Programming has not failed.<br>1 Programming from a copy in either direction failed to complete. |
| 2 | ERETRY | RO | - | Erase Must Be Retried<br>Value Description<br>0 Erasing has not failed.<br>1 Erasing failed to complete. If the failed erase is due to the erase of a main buffer, the copy is performed after the erase completes successfully. |
| 1:0 | reserved | RO | 0 | 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. |

## Register 24: EEPROM Unlock (EEUNLOCK), offset 0x020

The **EEUNLOCK** register can be used to unlock the whole EEPROM or a single block using a password. Unlocking is only required if a password is registered using the **EEPASSn** registers for the block that is selected by the **EEBLOCK** register. If block 0 has a password, it locks the remaining blocks from any type of access, but uses its own protection mechanism, for example readable, but not writable when locked. In addition, if block 0 has a password, it must be unlocked before unlocking any other block.

The **EEUNLOCK** register is written between 1 and 3 times to form the 32-bit, 64-bit, or 96-bit password registered using the **EEPASSn** registers. The value used to configure the **EEPASS0** register must always be written last. For example, for a 96-bit password, the value used to configure the **EEPASS2** register must be written first followed by the **EEPASS1** and **EEPASS0** register values. The block or the whole EEPROM can be re-locked by writing 0xFFFF.FFFF to this register.

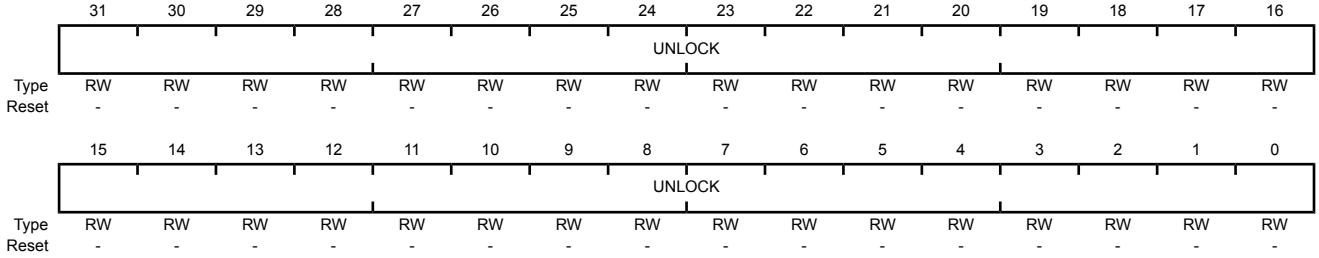
In the event that an invalid value is written to this register, the block remains locked. The state of the EEPROM lock can be determined by reading back the **EEUNLOCK** register. If a multi-word password is set and the number of words written is incorrect, writing 0xFFFF.FFFF to this register reverts the EEPROM lock to the locked state, and the proper unlock sequence can be retried.

Note that the internal logic is balanced to prevent any electrical or time-based attack being used to find the correct password or its length.

**Note:** A read of the **EEUNLOCK** register during the EEPROM initialization sequence is only valid when the `WORKING` bit is 0 in **EEDONE** register:

EEPROM Unlock (EEUNLOCK)

Base 0x400A.F000
Offset 0x020
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | UNLOCK | RW | - | EEPROM Unlock |

| Value | Description |
|---|---|
| 0 | The EEPROM is locked. |
| 1 | The EEPROM is unlocked. |

The EEPROM is locked if the block referenced by the **EEBLOCK** register has a password registered, or if the master block (block 0) has a password. Unlocking is performed by writing the password to this register. The block or the EEPROM stays unlocked until it is locked again or until the next reset. It can be locked again by writing 0xFFFF.FFFF to this register.

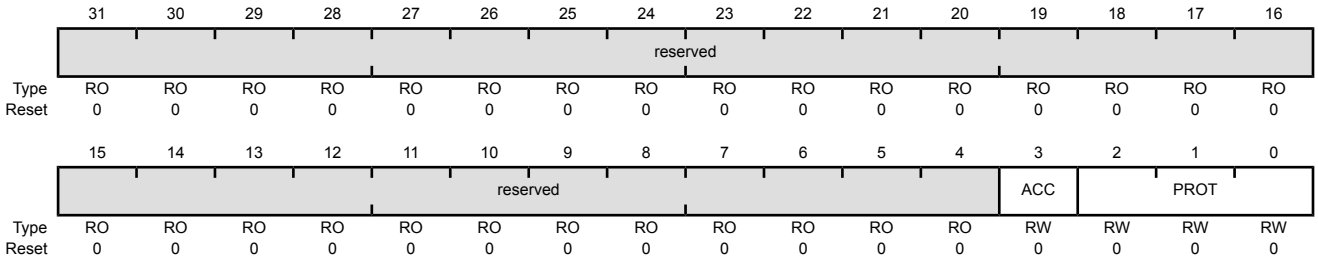
## Register 25: EEPROM Protection (EEPROT), offset 0x030

The **EEPROT** register is used to set or read the protection for the current block, as selected by the **EEBLOCK** register. Protection and access control is used to determine when a block's contents can be read or written.

**Note:** A read of the **EEPROT** register during the EEPROM initialization sequence is only valid when the `WORKING` bit is 0 in **EEDONE** register:

EEPROM Protection (EEPROT)

Base 0x400A.F000
Offset 0x030
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | ACC | RW | 0 | Access Control<br><br>Value Description<br>0 Both user and supervisor code may access this block of the EEPROM.<br>1 Only supervisor code may access this block of the EEPROM. µDMA and Debug are also prevented from accessing the EEPROM.<br><br>If this bit is set for block 0, then the whole EEPROM may only be accessed by supervisor code. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2:0 | PROT | RW | 0x0 | Protection Control<br>The Protection bits control what context is needed for reading and writing the block selected by the **EEBLOCK** register, or if block 0 is selected, all blocks. The following values are allowed:<br><br>Value Description<br>0x0 This setting is the default.<br>Without password: the block is not protected and is readable and writable at any time.<br>With password: the block is readable, but only writable when unlocked.<br>0x1 With password: the block is readable or writable only when unlocked.<br>This value has no meaning when there is no password.<br>0x2 Without password: the block is readable, not writable.<br>With password: the block is readable only when unlocked, but is not writable under any conditions.<br>0x3 Reserved |

## Register 26: EEPROM Password (EEPASS0), offset 0x034

## Register 27: EEPROM Password (EEPASS1), offset 0x038

## Register 28: EEPROM Password (EEPASS2), offset 0x03C

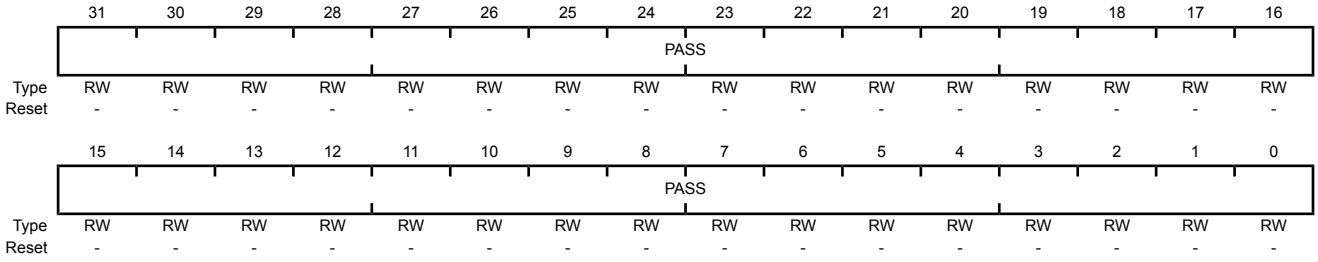
The **EEPASSn** registers are used to configure a password for a block. A password may only be set once and cannot be changed. The password may be 32-bits, 64-bits, or 96-bits. Each word of the password can be any 32-bit value other than 0xFFFF.FFFF (all 1s). To set a password, the **EEPASS0** register is written to with a value other than 0xFFFF.FFFF. When the write completes, as indicated in the **EEDONE** register, the application may choose to write to the **EEPASS1** register with a value other than 0xFFFF.FFFF. When that write completes, the application may choose to write to the **EEPASS2** register with a value other than 0xFFFF.FFFF to create a 96-bit password. The registers do not have to be written consecutively, and the **EEPASS1** and **EEPASS2** registers may be written at a later date. Based on whether 1, 2, or all 3 registers have been written, the unlock code also requires the same number of words to unlock.

**Note:** Once the password is written, the block is not actually locked until either a reset occurs or 0xFFFF.FFFF is written to **EEUNLOCK**.

**Note:** A read of the **EEPASSn** register during the EEPROM initialization sequence is only valid when the WORKING bit is 0 in **EEDONE** register:

EEPROM Password (EEPASSn)

Base 0x400A.F000
Offset 0x034
Type RW, reset -



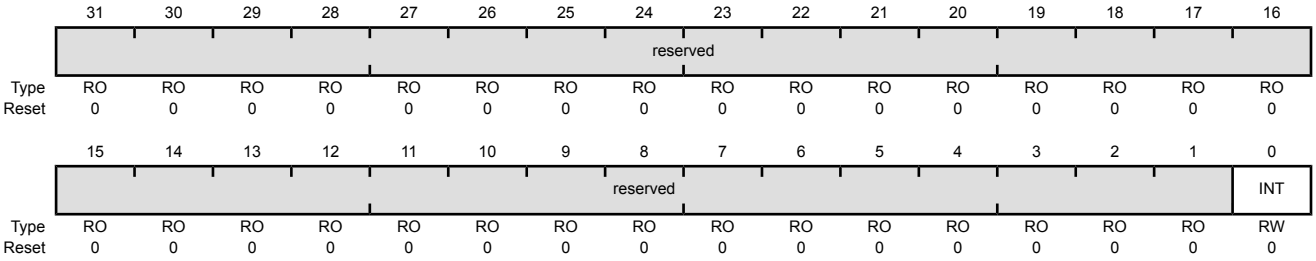

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PASS | RW | - | Password<br>This register reads as 0x1 if a password is registered for this block and 0x0 if no password is registered. A write to this register if it reads as 0x0 sets the password. If an attempt is made to write to this register when it reads as 0x1, the write is ignored and the NOPERM bit in the **EEDONE** register is set. |

## Register 29: EEPROM Interrupt (EEINT), offset 0x040

The **EEINT** register is used to control whether an interrupt should be generated when a write to EEPROM completes as indicated by the **EEDONE** register value changing from 0x1 to any other value. If the INT bit in this register is set, the ERIS bit in the **Flash Controller Raw Interrupt Status (FCRIS)** register is set whenever the **EEDONE** register value changes from 0x1 as the Flash memory and the EEPROM share an interrupt vector.

EEPROM Interrupt (EEINT)

Base 0x400A.F000
Offset 0x040
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | INT | RW | 0 | Interrupt Enable |

| Value | Description |
|---|---|
| 0 | No interrupt is generated. |
| 1 | An interrupt is generated when the **EEDONE** register transitions from 1 to 0 or an error occurs. The **EEDONE** register provides status after a write to an offset location as well as a write to the password and protection bits. |

## Register 30: EEPROM Block Hide 0 (EEHIDE0), offset 0x050

The **EEHIDE0** register is used to hide one or more blocks other than EEPROM block 0. Bits 1 through 31 of this register correspond to EEPROM blocks 1 through 31. Once hidden, the block is not accessible until the next reset. This model allows initialization code to have access to data which is not visible to the rest of the application. This register also provides for additional security in that there is no password to search for in the code or data.

EEPROM Block Hide 0 (EEHIDE0)

Base 0x400A.F000
Offset 0x050
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hn | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hn | | | | | | | | | | | | | | | reserved |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | Hn | RW | 0x0000.000 | Hide Block |

| Value | Description |
|---|---|
| 0 | The corresponding block is not hidden. |
| 1 | The block number that corresponds to the bit number is hidden. A hidden block cannot be accessed, and the OFFSET value in the **EEBLOCK** register cannot be set to that block number. If an attempt is made to configure the OFFSET field to a hidden block, the **EEBLOCK** register is cleared. |
| | Any attempt to clear a bit in this register that is set is ignored. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | reserved | RO | 0 | 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. |

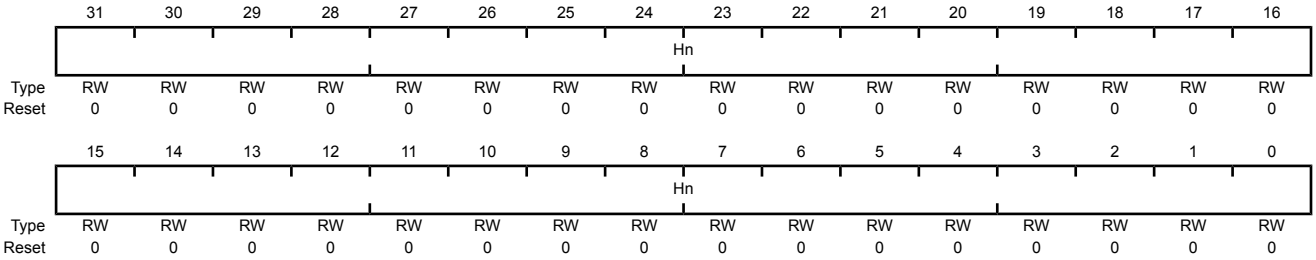
## Register 31: EEPROM Block Hide 1 (EEHIDE1), offset 0x054

## Register 32: EEPROM Block Hide 2 (EEHIDE2), offset 0x058

The **EEHIDE** register is used to hide one or more blocks. Bits 0 through 31 of the EEHIDE1 register correspond to EEPROM blocks 32 through 63. Bits 0 through 31 of the EEHIDE2 register correspond to EEPROM blocks 64 through 95. Once hidden, the block is not accessible until the next reset. This model allows initialization code to have access to data which is not visible to the rest of the application. This register also provides for additional security in that there is no password to search for in the code or data.

EEPROM Block Hide n (EEHIDEn)

Base 0x400A.F000
Offset 0x054
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | Hn | RW | 0x0000.000 | Hide Block |

| Value | Description |
|---|---|
| 0 | The corresponding block is not hidden. |
| 1 | The block number that corresponds to the bit number is hidden. A hidden block cannot be accessed, and the OFFSET value in the **EEBLOCK** register cannot be set to that block number. If an attempt is made to configure the OFFSET field to a hidden block, the **EEBLOCK** register is cleared.<br>Any attempt to clear a bit in this register that is set is ignored. |

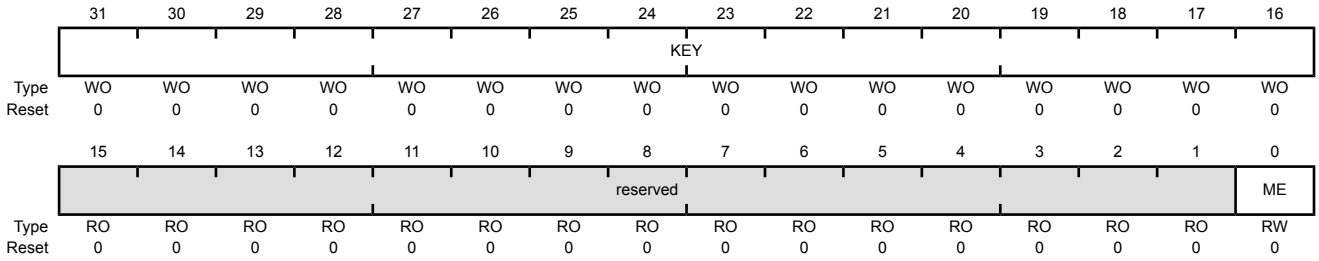
## Register 33: EEPROM Debug Mass Erase (EEDBGME), offset 0x080

The **EEDBGME** register is used to mass erase the EEPROM block back to its default state from the factory. This register is intended to be used only for debug and test purposes, not in production environments. The erase takes place in such a way as to be secure. It first erases all data and then erases the protection mechanism. This register can only be written from supervisor mode by the core, and can also be written by the Tiva™ C Series debug controller when enabled. A key is used to avoid accidental use of this mechanism. Note that if a power down takes place while erasing, the mechanism should be used again to complete the operation. Powering off prematurely does not expose secured data.

To start a mass erase, the whole register must be written as 0xE37B.0001. The register reads back as 0x1 until the erase is fully completed at which time it reads as 0x0. The **EEDONE** register is set to 0x1 when the erase is started and changes to 0x0 or an error when the mass erase is complete.

EEPROM Debug Mass Erase (EEDBGME)

Base 0x400A.F000
Offset 0x080
Type RW, reset 0x0000.0000



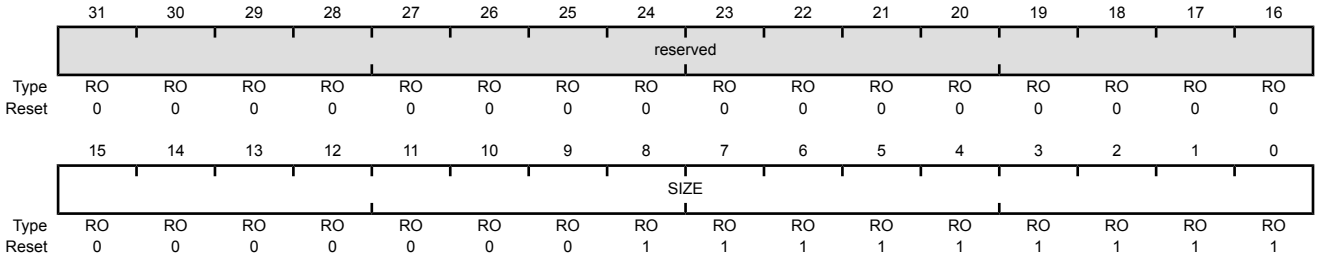

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | KEY | WO | 0x0000 | Erase Key<br>This field must be written with 0xE37B for the ME field to be effective. |
| 15:1 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | ME | RW | 0 | Mass Erase<br>Value Description<br>0 No action.<br>1 When written as a 1, the EEPROM is mass erased. This bit continues to read as 1 until the EEPROM is fully erased. |

### Register 34: EEPROM Peripheral Properties (EEPROMPP), offset 0xFC0

The **EEPROMPP** register indicates the size of the EEPROM for this part.

EEPROM Peripheral Properties (EEPROMPP)

Base 0x400A.F000
Offset 0xFC0
Type RO, reset 0x0000.01FF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | 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. |
| 15:0 | SIZE | RO | 0x1FF | EEPROM Size<br>Indicates the size of the on-chip EEPROM. Any values not shown are reserved. |

| Value | Description |
|---|---|
| 0x0000 | 64 bytes of EEPROM |
| 0x0001 | 128 bytes of EEPROM |
| 0x0003 | 256 bytes of EEPROM |
| 0x0007 | 512 bytes of EEPROM |
| 0x000F | 1 KB of EEPROM |
| 0x001F | 2 KB of EEPROM |
| 0x003F | 3 KB of EEPROM |
| 0x007F | 4 KB of EEPROM |
| 0x00FF | 5 KB of EEPROM |
| 0x01FF | 6 KB of EEPROM |

## 8.6 Memory Register Descriptions (System Control Offset)

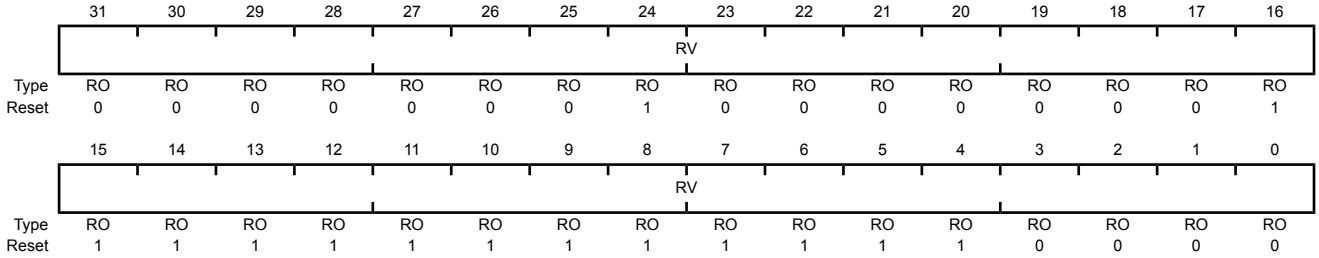
The remainder of this section lists and describes the registers that reside in the System Control address space, in numerical order by address offset. Registers in this section are relative to the System Control base address of 0x400F.E000.

## Register 35: Reset Vector Pointer (RVP), offset 0x0D4

The **Reset Vector Pointer (RVP)** register contains the address of the reset vector of the software module that is to be executed after boot loader execution. The **RVP** register is initialized by a power-on reset.

Reset Vector Pointer (RVP)

Base 0x400F.E000
Offset 0x0D4
Type RO, reset 0x0101.FFF0




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RV | RO | 0x0101.FFF0 | Reset Vector Pointer Address |

### Register 36: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x200

### Register 37: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

### Register 38: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

### Register 39: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

### Register 40: Flash Memory Protection Read Enable 4 (FMPRE4), offset 0x210

### Register 41: Flash Memory Protection Read Enable 5 (FMPRE5), offset 0x214

### Register 42: Flash Memory Protection Read Enable 6 (FMPRE6), offset 0x218

### Register 43: Flash Memory Protection Read Enable 7 (FMPRE7), offset 0x21C

**Note:** The **FMPRE0** register is aliased for backwards compatibility.

**Note:** Offset is relative to System Control base address of 0x400F.E000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Note that for protecting sectors, eight bits need to be cleared to create a 16-KB read-protected sector.

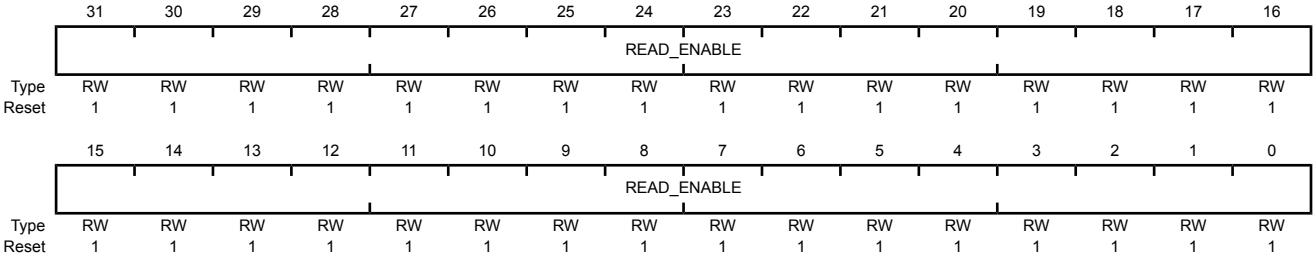
This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is RW0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter.

Each **FMPREn** register controls a 64K block. For additional information, see “Protected Flash Memory Registers” on page 641.

- **FMPRE0**: 0 to 64 KB
- **FMPRE1**: 65 to 128 KB
- **FMPRE2**: 129 to 192 KB
- **FMPRE3**: 193 to 256 KB
- **FMPRE4**: 257 to 320 KB
- **FMPRE5**: 321 to 384 KB
- **FMPRE6**: 385 to 448 KB
- **FMPRE7**: 449 to 512 KB

### Flash Memory Protection Read Enable n (FMPREn)

Base 0x400F.E000
Offset 0x200
Type RW, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | RW | 0xFFFF.FFFF | Flash Read Enable<br>Each bit configures a 2-KB flash block to be read only. Note that for read-protection of sectors, eight bits need to be cleared to create a 16-KB read-protected sector.<br>The policies may be combined as shown in the table "Flash Protection Policy Combinations". |

### Register 44: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x400

### Register 45: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

### Register 46: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

### Register 47: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

### Register 48: Flash Memory Protection Program Enable 4 (FMPPE4), offset 0x410

### Register 49: Flash Memory Protection Program Enable 5 (FMPPE5), offset 0x414

### Register 50: Flash Memory Protection Program Enable 6 (FMPPE6), offset 0x418

### Register 51: Flash Memory Protection Program Enable 7 (FMPPE7), offset 0x41C

**Note:** The **FMPPE0** register is aliased for backwards compatibility.

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the read-only protection bits). Since the memory is two-way interleaved and each bank individually is an 8-KB sector, read-only protection must occur across a block size of 16-KB. No smaller block size is supported. Note that the **Flash Memory Protection Read (FMPREn)** registers do allow read-protection of a block as small as 2 KB, unlike the **FMPPEn** registers.

Thus, in order to execute-only protect a 16-KB block, a user must program the entire eight bits of the byte to the same value. For example, to protect the first 16-KB block, bits [7:0] of the **FMPPE0** register need to be cleared to all 0s.

This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. This register is RW0; the user can only change the protection byte from all 1s to all 0s (and may NOT change from all 0 to all 1). The changes are not permanent until the register is committed (saved), at which point the byte change is permanent. If a byte is changed from all 1s to all 0s and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. For additional information, see “Protected Flash Memory Registers” on page 641.

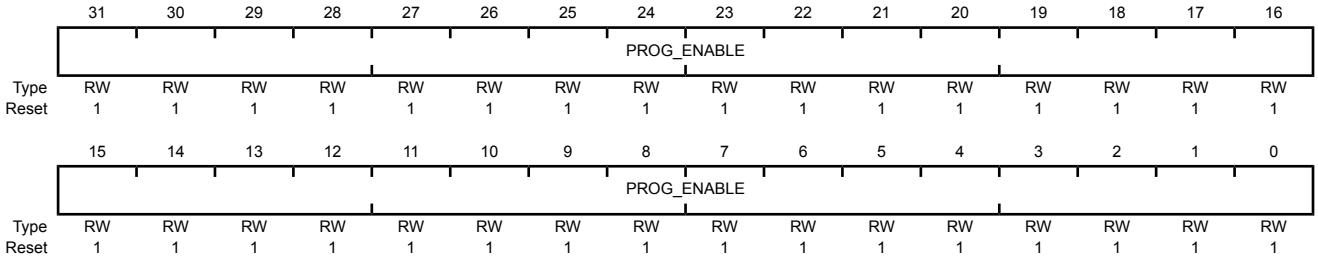
Each **FMPPEn** register controls a 64K block. For additional information, see “Protected Flash Memory Registers” on page 641.

- **FMPPE0**: 0 to 64 KB
- **FMPPE1**: 65 to 128 KB
- **FMPPE2**: 129 to 192 KB
- **FMPPE3**: 193 to 256 KB

- **FMPPE4**: 257 to 320 KB
- **FMPPE5**: 321 to 384 KB
- **FMPPE6**: 385 to 448 KB
- **FMPPE7**: 449 to 512 KB

Flash Memory Protection Program Enable n (FMPPEn)

Base 0x400F.E000
Offset 0x400
Type RW, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PROG_ENABLE | RW | 0xFFFF.FFFF | Flash Programming Enable<br>Every eighth bit programs an 16-KB flash sector to be execute only.<br>The policies may be combined as shown in Table 8-2 on page 642. |

## Register 52: Boot Configuration (BOOTCFG), offset 0x1D0

**Note:** Offset is relative to System Control base address of 0x400F.E000.

**Note:** The **Boot Configuration (BOOTCFG)** register requires a POR before the committed changes take effect.

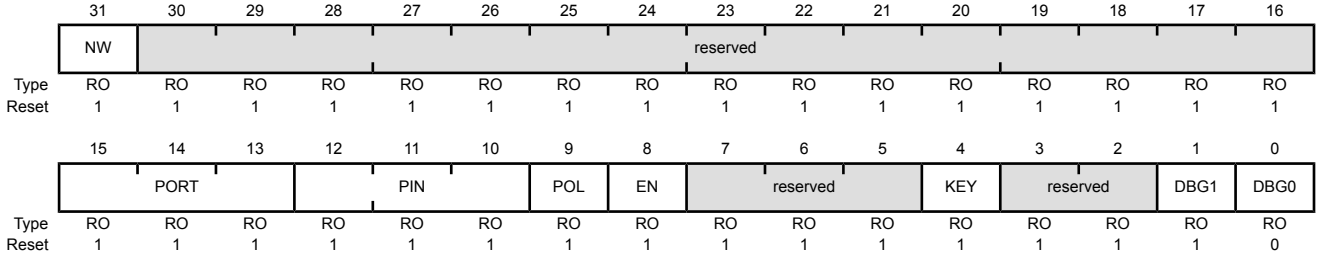
This register is not written directly, but instead uses the **FMD** register as explained in “Non-Volatile Register Programming-- Flash Memory Resident Registers” on page 645. When this register is committed, the new value cannot be read back until after the power cycle. This register provides configuration of a GPIO pin to enable the ROM Boot Loader as well as a write-once mechanism to disable external debugger access to the device. At reset, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal from Ports A through H as configured by the bits in this register. At reset, the following sequence is performed:

1. The **BOOTCFG** register is read. If the EN bit is clear, the ROM Boot Loader is executed.
2. In the ROM Boot Loader, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
3. If the EN bit is set or the status doesn't match the specified polarity, the data at address 0x0000.0004 is read, and if the data at this address is 0xFFFF.FFFF, the ROM is mapped to address 0x0000.0000 and execution continues out of the ROM Boot Loader.
4. If there is data at address 0x0000.0004 that is not 0xFFFF.FFFF, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

The DBG0 bit is cleared by the factory and the DBG1 bit is set, which enables external debuggers. Clearing the DBG1 bit disables any external debugger access to the device, starting with the next power-up cycle of the device. The NW bit indicates that bits in the register can be changed from 1 to 0.

By committing the register values using the COMT bit in the **FMC** register, the register contents become non-volatile and are therefore retained following power cycling. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset when the register is not yet committed; any other type of reset does not affect this register. Once committed, the register retains its value through power-on reset. Once committed, the only way to restore the factory default value of this register is to perform the sequence detailed in “Recovering a "Locked" Microcontroller” on page 224.

Boot Configuration (BOOTCFG)
Base 0x400F.E000
Offset 0x1D0
Type RO, reset 0xFFFF.FFFE



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NW | RO | 1 | Not Written<br>When set, this bit indicates that the values in this register can be changed from 1 to 0. When clear, this bit specifies that the contents of this register cannot be changed. |
| 30:16 | reserved | RO | 0xFFFF | 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. |
| 15:13 | PORT | RO | 0x7 | Boot GPIO Port<br>This field selects the port of the GPIO port pin that enables the ROM boot loader at reset.<br>**Note:** The selected port can be reprogrammed for a different function after reset.<br>Value Description<br>0x0 Port A<br>0x1 Port B<br>0x2 Port C<br>0x3 Port D<br>0x4 Port E<br>0x5 Port F<br>0x6 Port G<br>0x7 Port H |
| 12:10 | PIN | RO | 0x7 | Boot GPIO Pin<br>This field selects the pin number of the GPIO port pin that enables the ROM boot loader at reset.<br>Value Description<br>0x0 Pin 0<br>0x1 Pin 1<br>0x2 Pin 2<br>0x3 Pin 3<br>0x4 Pin 4<br>0x5 Pin 5<br>0x6 Pin 6<br>0x7 Pin 7 |
| 9 | POL | RO | 1 | Boot GPIO Polarity<br>When set, this bit selects a high level for the GPIO port pin to enable the ROM boot loader at reset. When clear, this bit selects a low level for the GPIO port pin. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | EN | RO | 1 | Boot GPIO Enable<br>Clearing this bit enables the use of a GPIO pin to enable the ROM Boot Loader at reset. When this bit is set, the contents of address 0x0000.0004 are checked to see if the Flash memory has been programmed. If the contents are not 0xFFFF.FFFF, the core executes out of Flash memory. If the Flash has not been programmed, the core executes out of ROM. |
| 7:5 | reserved | RO | 0x7 | 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. |
| 4 | KEY | RO | 1 | KEY Select<br>This bit chooses between using the value 0xA442 or the PEKEY value in the **FLPEKEY** register as the WRKEY value in the **FMC/FMC2** register.<br>Value Description<br>0 The PEKEY value in the **FLPEKEY** register is committed by user and used as the WRKEY in the **FMC/FMC2** register. Writes to **FMC/FMC2** register with a 0xA442 key are ignored.<br>1 0xA442 is used as key |
| 3:2 | reserved | RO | 0x3 | 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. |
| 1 | DBG1 | RO | 1 | Debug Control 1<br>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available. |
| 0 | DBG0 | RO | 0 | Debug Control 0<br>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available. |

### Register 53: User Register 0 (USER_REG0), offset 0x1E0

### Register 54: User Register 1 (USER_REG1), offset 0x1E4

### Register 55: User Register 2 (USER_REG2), offset 0x1E8

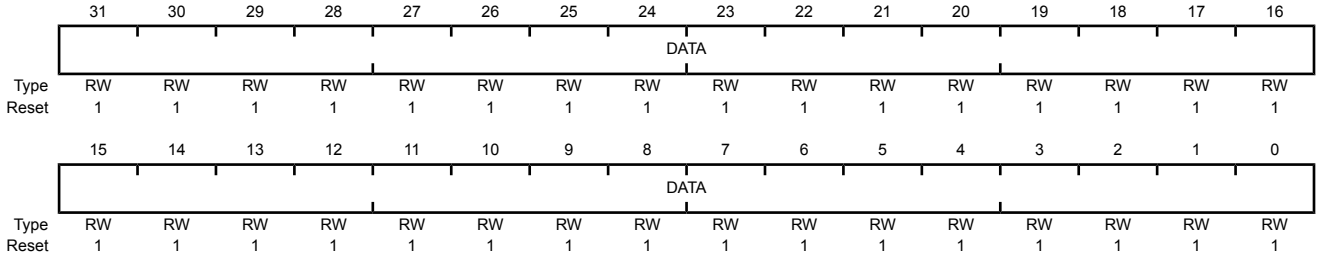
### Register 56: User Register 3 (USER_REG3), offset 0x1EC

**Note:** Offset is relative to System Control base address of 0x400F.E000.

These registers each provide 32 bits of user-defined data that is non-volatile. Bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset when the register is not yet committed; any other type of reset does not affect this register. Once committed, the register retains its value through power-on reset. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG section.

User Register n (USER_REGn)

Base 0x400F.E000
Offset 0x1E0
Type W0, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RW | 0xFFFF.FFFF | User Data<br>Contains the user data value. This field is initialized to all 1s and once committed, retains its value through power-on reset. |

# 9 Micro Direct Memory Access (µDMA)

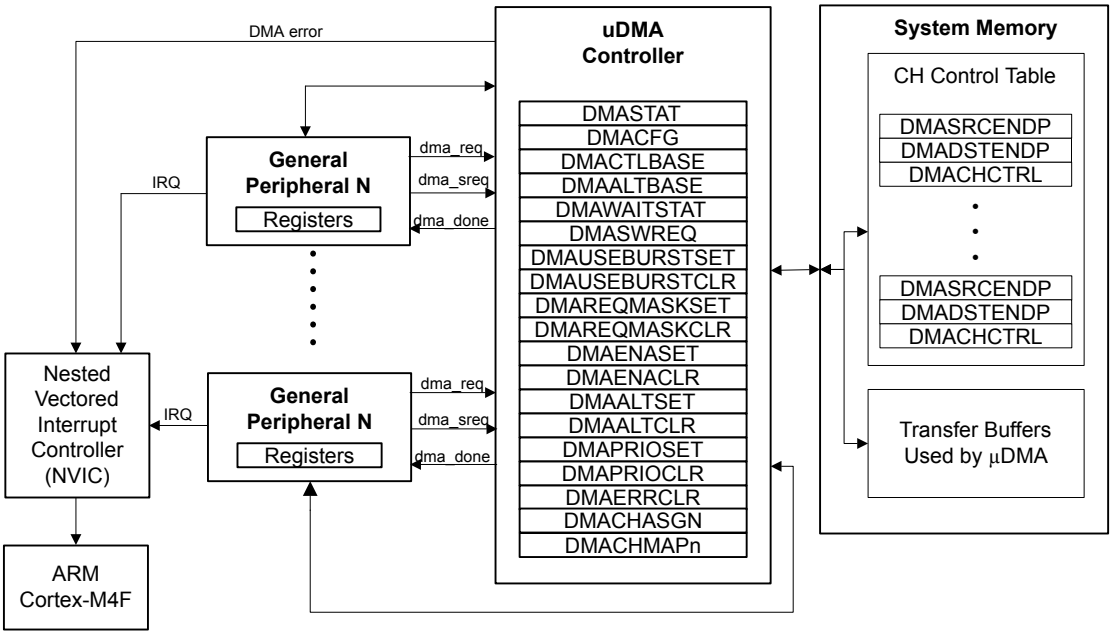
The TM4C129XKCZAD microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the Cortex™-M4F processor, allowing for more efficient use of the processor and the available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. It 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 in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of up to 256 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
  - Flexible channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable priority scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between µDMA controller and the processor core
  - µDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- 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, half-word, word, or no increment

- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

## 9.1 Block Diagram

**Figure 9-1. µDMA Block Diagram**




## 9.2 Functional Description

The µDMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the microcontroller's Cortex-M4F processor core. It 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. The µDMA controller's usage of the bus is always subordinate to the processor core, so it never holds up a bus transaction by the processor. Because the µDMA controller is only using otherwise-idle bus cycles, the data transfer bandwidth it provides is essentially free, with no impact on the rest of the system. The bus architecture has been optimized to greatly enhance the ability of the processor core and the µDMA controller to efficiently share the on-chip bus, thus improving performance. The optimizations include RAM striping and peripheral bus segmentation, which in many cases allow both the processor core and the µDMA controller to access the bus and perform simultaneous data transfers.

Each peripheral function that is supported has a dedicated channel on the µDMA controller that can be configured independently. The µDMA controller implements a unique configuration method using channel control structures that are 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 each time it makes a µDMA service request.

### 9.2.1 Channel Assignments

Each DMA channel has up to nine possible assignments which are selected using the **DMA Channel Map Select n (DMACHMAPn)** registers with 4-bit assignment fields for each µDMA channel.

Table 9-1 on page 710 shows the µDMA channel mapping. The Enc. column shows the encoding for the respective **DMACHMAPn** bit field. Encodings 0x9-0xF are reserved. To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, Enc. 0 is equivalent to a **DMACHASGN** bit being clear, and Enc. 1 is equivalent to a **DMACHASGN** bit being set. If the **DMACHASGN** register is read, bit fields return 0 if the corresponding **DMACHMAPn** register field value are equal to 0, otherwise they return 1 if the corresponding **DMACHMAPn** register field values are not equal to 0. The Type indication in the table indicates if a particular peripheral uses a single request (S), burst request (B) or either (SB).

**Note:** Channels or encodings marked as reserved cannot be used for µDMA transfers. Channels designated in the table as only "Software" are dedicated software channels. When only one software request is required in an application, dedicated software channels can be used. If multiple software requests in code are required, then peripheral channel software requests should be used for proper µDMA completion acknowledgement.

**Table 9-1. µDMA Channel Assignments**

| Channel | Encoding | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | |
| | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type |
| 0 | Reserved | - | UART2 RX | SB | Reserved | - | GPTimer 4A | B | Reserved | - | Reserved | - | I2C0 RX | SB B | Reserved | - | Reserved | - |
| 1 | Reserved | - | UART2 TX | SB | Reserved | - | GPTimer 4B | B | Reserved | - | Reserved | - | I2C0 TX | SB B | Reserved | - | Reserved | - |
| 2 | Reserved | - | GPTimer 3A | B | Reserved | - | Reserved | - | Reserved | - | Reserved | - | I2C1RX | SB B | Reserved | - | Reserved | - |
| 3 | Reserved | - | GPTimer 3B | B | Reserved | - | Software | S | Reserved | - | Reserved | - | I2C1 TX | SB B | Reserved | - | Reserved | - |
| 4 | Reserved | - | GPTimer 2A | B | Reserved | - | GPIO A | B | Reserved | - | SHA/MD5 0 Cin | B | I2C2 RX | SB B | Reserved | - | Reserved | - |
| 5 | Reserved | - | GPTimer 2B | B | Reserved | - | GPIO B | B | Reserved | - | SHA/MD5 0 Din | B | I2C2 TX | SB B | Reserved | - | Reserved | - |
| 6 | Reserved | - | GPTimer 2A | B | UART5 RX | SB | GPIO C | B | I2C0 RX | SB B | SHA/MD5 0 Cout | B | Reserved | - | Reserved | - | Reserved | - |
| 7 | Reserved | - | GPTimer 2B | B | UART5 TX | SB | GPIO D | B | I2C0 TX | SB B | Reserved | - | Reserved | - | Reserved | - | Reserved | - |
| 8 | UART0 RX | SB | UART1 RX | SB | Reserved | - | GPTimer 5A | B | I2C1RX | SB B | Reserved | - | Reserved | - | Reserved | - | Reserved | - |
| 9 | UART0 TX | SB | UART1 TX | SB | Reserved | - | GPTimer 5B | B | I2C1 TX | SB B | Reserved | - | Reserved | - | Reserved | - | Reserved | - |
| 10 | SSI0 RX | SB | SSI1 RX | SB | UART6 RX | SB | Reserved | - | I2C2 RX | SB B | Reserved | - | Reserved | - | GPTimer 6A | B | Reserved | - |

**Table 9-1. µDMA Channel Assignments *(continued)***

| Channel | Encoding | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | |
| | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type | Peripheral | Type |
| 11 | SSI0 TX | SB | SSI1 TX | SB | UART6 TX | SB | Reserved | - | I2C2 TX | SB B | Reserved | - | Reserved | - | GPTimer 6B | B | Reserved | - |
| 12 | Reserved | - | UART2 RX | SB | SSI2 RX | SB | Reserved | - | GPIO K | B | AES0 Cin | B | Reserved | - | GPTimer 7A | B | Reserved | - |
| 13 | Reserved | - | UART2 TX | SB | SSI2 TX | SB | Reserved | - | GPIO L | B | AES0 Cout | B | Reserved | - | GPTimer 7B | B | Reserved | - |
| 14 | ADC0 SS0 | SB | GPTimer 2A | B | SSI3 RX | SB | GPIO E | B | GPIO M | B | AES0 Din | B | Reserved | - | Reserved | - | Reserved | - |
| 15 | ADC0 SS1 | SB | GPTimer 2B | B | SSI3 TX | SB | GPIO F | B | GPIO N | B | AES0 Dout | B | Reserved | - | Reserved | - | Reserved | - |
| 16 | ADC0 SS2 | SB | Reserved | - | UART3 RX | SB | Reserved | - | GPIO P | B | Reserved | - | Reserved | - | Reserved | - | Reserved | - |
| 17 | ADC0 SS3 | SB | Reserved | - | UART3 TX | SB | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - |
| 18 | GPTimer 0A | B | GPTimer 1A | B | UART4 RX | SB | GPIO B | B | I2C3 RX | SB B | Reserved | - | Reserved | - | Reserved | - | Reserved | - |
| 19 | GPTimer 0B | B | GPTimer 1B | B | UART4 TX | SB | GPIO G | B | I2C3 TX | SB B | Reserved | - | Reserved | - | Reserved | - | Reserved | - |
| 20 | GPTimer 1A | B | EPI 0 RX Software | B | UART7 RX | SB | GPIO H | B | I2C4 RX | SB B | DES0 Cin | B | Reserved | - | Reserved | - | Reserved | - |
| 21 | GPTimer 1B | B | EPI 0 TX Software | B | UART7 TX | SB | GPIO J | B | I2C4 TX | SB B | DES0 Din | B | Reserved | - | Reserved | - | Reserved | - |
| 22 | UART1 RX | SB | Software | B | Reserved | - | Software | B | I2C5 RX | SB B | DES0 Dout | B | Reserved | - | Reserved | - | I2C8 RX | B |
| 23 | UART1 TX | SB | Software | B | Reserved | - | Software | B | I2C5 TX | SB B | Reserved | - | Reserved | - | Reserved | - | I2C8 TX | B |
| 24 | SSI1 RX | SB | ADC1 SS0 | SB | Reserved | - | Reserved | - | GPIO Q | B | Reserved | - | Reserved | - | Reserved | - | I2C9 RX | B |
| 25 | SSI1 TX | SB | ADC1 SS1 | SB | Reserved | - | Reserved | - | GPIO R | B | Reserved | - | Reserved | - | Reserved | - | I2C9 TX | B |
| 26 | Software | B | ADC1 SS2 | SB | Reserved | - | Reserved | - | GPIO S | B | Reserved | - | Reserved | - | Reserved | - | I2C6 RX | B |
| 27 | Software | B | ADC1 SS3 | SB | Reserved | - | Reserved | - | Reserved | - | Reserved | - | GPIO T | B | Reserved | - | I2C6 TX | B |
| 28 | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | I2C7 RX | B |
| 29 | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | Reserved | - | I2C7 TX | B |
| 30 | Software | B | Software | B | Reserved | - | Software | B | Reserved | - | Reserved | - | Reserved | - | EPI0 RX | B | 1Wire0 | B |
| 31 | Reserved | - | Reserved | - | Reserved | - | Reserved | B | Reserved | - | Reserved | - | Reserved | - | EPI0 TX | B | Reserved | - |

## 9.2.2 Priority

The µDMA controller assigns priority to each channel based on the channel number and the priority level bit for the channel. Channel number 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 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 **DMA Channel Priority Set (DMAPRIOSET)** register and cleared with the **DMA Channel Priority Clear (DMAPRIOCLR)** register.

**Note:** If one peripheral is mapped to two different channels, then the application should either use the default mapping for that peripheral or change the default mapping to another source. For example, if UART1 channels 8 and 9 are enabled for use, then even if channels 22 and 23 are disabled, they must be mapped to software or another peripheral (if available).

### 9.2.3 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 again. 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, it 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 should not use a large arbitration size for best response on high priority channels.

The arbitration size can also be thought of as a burst size. It is the maximum number of items that are transferred at any one time in a burst. Here, the term arbitration refers to determination of 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 held off whenever the processor must perform a bus transaction on the same bus, even in the middle of a burst transfer.

### 9.2.4 Request Types

The µDMA controller responds to two types of requests from a peripheral: single or burst. 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 are asserted, and the µDMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 9-2 on page 712, which shows how each peripheral supports the two request types.

**Table 9-2. Request Type Support**

| Peripheral | Event that generates Single Request | Event that generates Burst Request |
|---|---|---|
| ADC | FIFO not empty | FIFO half full |
| EPI WFIFO | None | WFIFO Level (configurable) |
| EPI NBRFIFO | None | NBRFIFO Level (configurable) |
| General-Purpose Timer | None | Trigger event |
| GPIO | None | Trigger event |
| $I^2C$ TX | TX Buffer Not Full | TX FIFO Level (configurable) |
| $I^2C$ RX | RX Buffer Not Empty | RX FIFO Level (configurable) |

**Table 9-2. Request Type Support *(continued)***

| Peripheral | Event that generates Single Request | Event that generates Burst Request |
|---|---|---|
| SSI TX | TX FIFO Not Full | TX FIFO Level (fixed at 4) |
| SSI RX | RX FIFO Not Empty | RX FIFO Level (fixed at 4) |
| UART TX | TX FIFO Not Full | TX FIFO Level (configurable) |
| UART RX | RX FIFO Not Empty | RX FIFO Level (configurable) |
| SHA/MD5 | None | Context In DMA request (SHA/MD5 0 Cin)<br>Context Out DMA request (SHA/MD5 0 Cout)<br>Data In DMA request (SHA/MD5 0 Din) |
| AES | None | Context In DMA request (AES0 Cin)<br>Context Out DMA request (AES0 Cout)<br>Data In DMA request (AES0 Din)<br>Data Out DMA request (AES0 Dout) |
| DES | None | Context In DMA request (DES0 Cin)<br>Data In DMA request (DES0 Din)<br>Data Out DMA request (DES0 Dout) |

#### 9.2.4.1 Single Request

When a single request is detected, and not a burst request, the µDMA controller transfers one item and then stops to wait for another request.

#### 9.2.4.2 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 should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART generates a burst request based on the FIFO trigger level. In this case, the arbitration size should 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 is started, and cannot be interrupted, even by a higher priority channel. Burst transfers complete in a shorter time than the same number of non-burst transfers.

It may be desirable to use only burst transfers and not allow single transfers. For example, perhaps 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 by using the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register. By setting the bit for a channel in this register, the µDMA controller only responds to burst requests for that channel.

### 9.2.5 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 9-3 on page 714 shows the layout in memory 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 so that all of the primary entries are in the first half of the table, and all the alternate structures are 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 is complete. In this case, the alternate control structures are not used and therefore only the first half of the table must be allocated in memory; the second half of the control table is not necessary, and that 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 should be allocated for the entire table.

Any unused memory in the control table may be used by the application. This includes the control structures for any channels that are unused by the application as well as the unused control word for each channel.

**Table 9-3. 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 9-4 shows an individual control structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four long words: the source end pointer, the destination end pointer, the control word, and an unused entry. The end pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

**Table 9-4. Channel Control Structure**

| Offset | Description |
|---|---|
| 0x000 | Source End Pointer |
| 0x004 | Destination End Pointer |
| 0x008 | Control Word |
| 0x00C | Unused |

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 word and each field are described in detail in “µDMA Channel Control Structure” on page 732. 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.

Prior to starting a transfer, a µDMA channel must be enabled by setting the appropriate bit in the **DMA Channel Enable Set (DMAENASET)** register. A channel can be disabled by setting the channel bit in the **DMA Channel Enable Clear (DMAENACLR)** register. At the end of a complete µDMA transfer, the controller automatically disables the channel.

### 9.2.6 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.

#### 9.2.6.1 Stop Mode

While Stop is not actually a transfer mode, it is a valid value for the mode field of the control word. When the mode field has this value, the µDMA controller does not perform any transfers and disables the channel if it is enabled. At the end of a transfer, the µDMA controller updates the control word to set the mode to Stop.

#### 9.2.6.2 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 should not be used in any situation where the request is momentary even though the entire transfer should be completed. For example, a software-initiated transfer creates a momentary request, and in Basic mode, only the number of transfers specified by the ARBSIZE field in the **DMA Channel Control Word (DMACHCTL)** register is transferred on a software request, even if there is more data to transfer.

When all of the items have been transferred using Basic mode, the µDMA controller sets the mode for that channel to Stop.

BASIC mode can be programmed to ignore when XFERSIZE reaches 0x000 and continue copying on request until the channel is stopped manually. If the NXTUSEBURST bit in the **uDMA Channel Control Word (DMACHCTL)** register is set while in BASIC mode and the XFERSIZE reaches 0x000 and is not written back, transfers continue until the request is deasserted by the peripheral.

#### 9.2.6.3 Auto Mode

Auto mode is similar to Basic mode, except that once a transfer request is received, the transfer runs to completion, even if the µDMA request is removed. This mode is suitable for software-triggered transfers. Generally, Auto mode is not used with a peripheral.

When all the items have been transferred using Auto mode, the µDMA controller sets the mode for that channel to Stop.

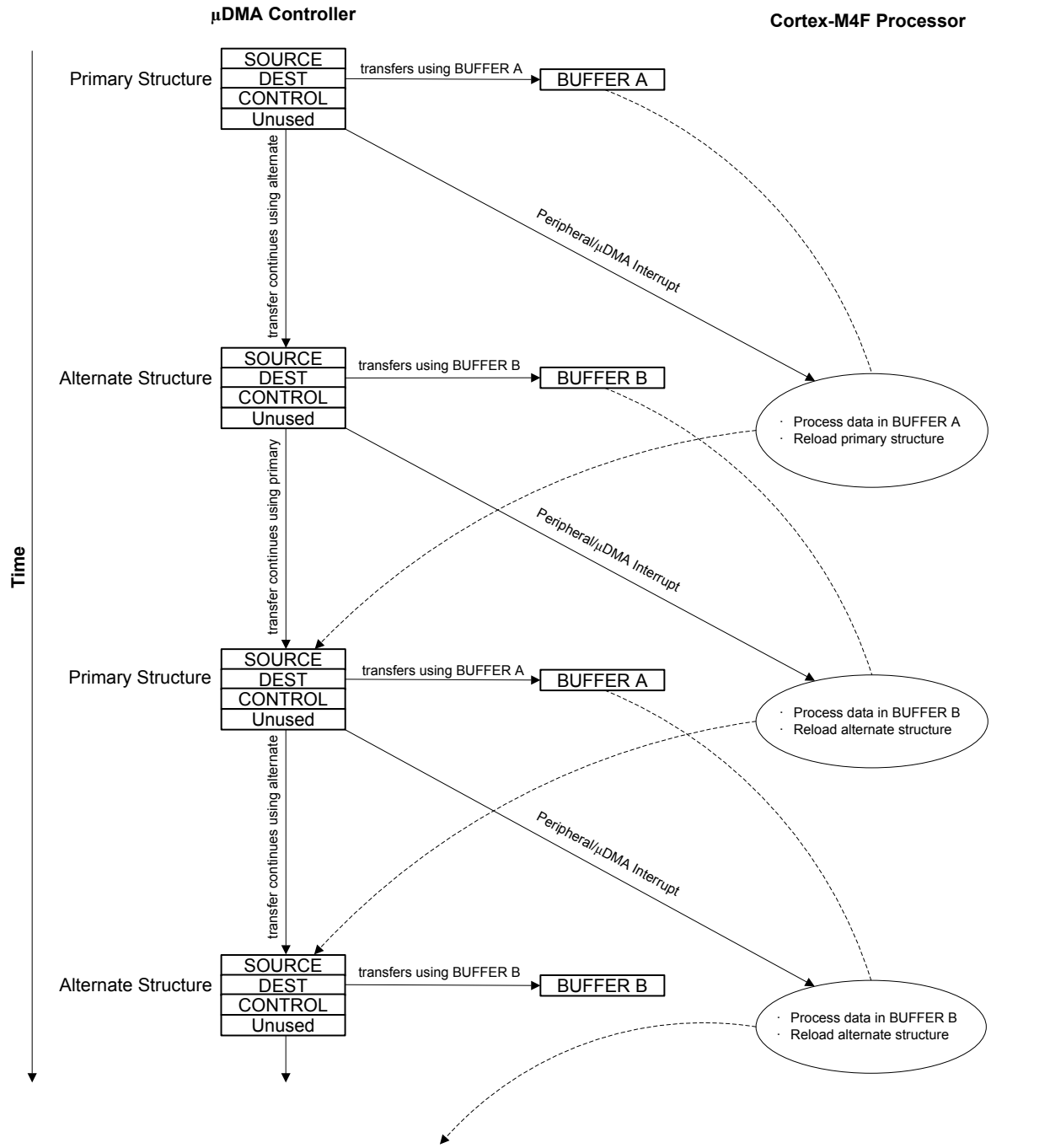
#### 9.2.6.4 Ping-Pong

Ping-Pong mode is used to support a continuous data flow to or from a peripheral. To use Ping-Pong mode, both the primary and alternate data structures must be implemented. 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 is complete, the µDMA controller reads the alternate control structure for that channel to continue the transfer. Each time this happens, 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 back and forth between buffers as the data flows to or from the peripheral.

Refer to Figure 9-2 on page 716 for an example showing operation in Ping-Pong mode.

**Figure 9-2. Example of Ping-Pong µDMA Transaction**



#### 9.2.6.5 Memory Scatter-Gather

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.

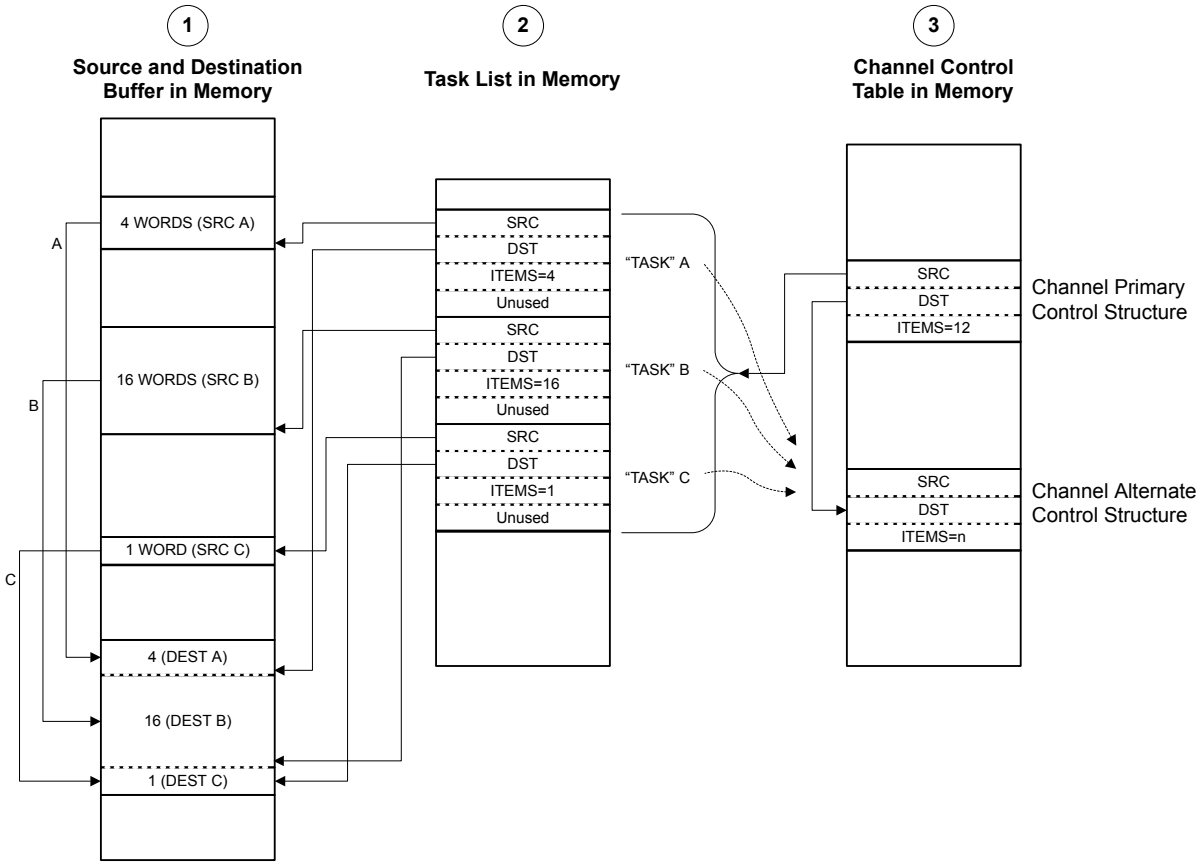
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to Scatter-Gather mode. Each entry in the table is copied in turn 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. Once 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 up to 256 arbitrary transfers can be performed based on a single µDMA request.

Refer to Figure 9-3 on page 718 and Figure 9-4 on page 719, which show 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 9-3 on page 718 shows how the application sets up a µDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

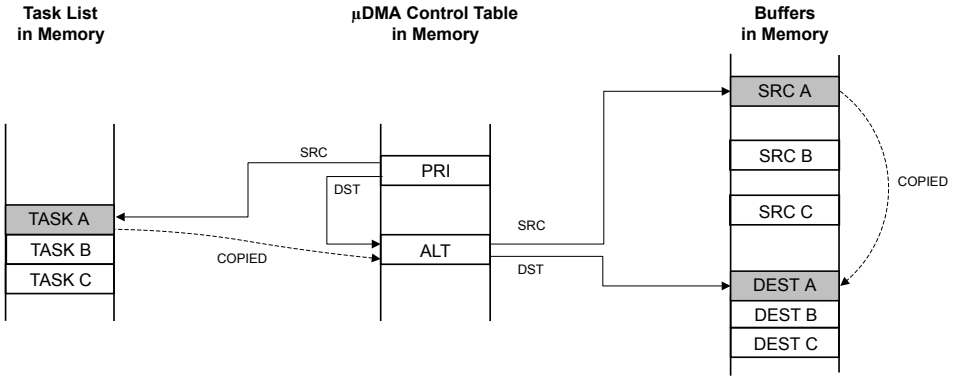
Figure 9-4 on page 719 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from 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.

**Figure 9-3. Memory Scatter-Gather, Setup and Configuration**



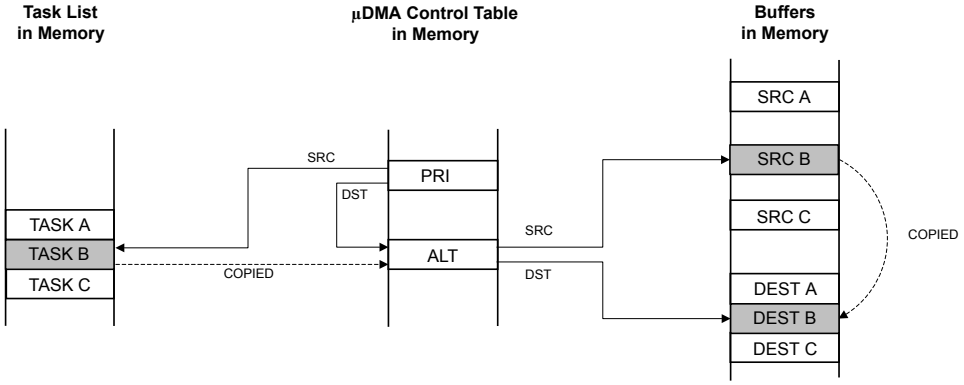

NOTES:
1. Application has a need to copy data items from three separate locations in memory into one combined buffer.
2. Application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.
4. The SRC and DST pointers in the task list must point to the last location in the corresponding buffer.

## Figure 9-4. Memory Scatter-Gather, µDMA Copy Sequence



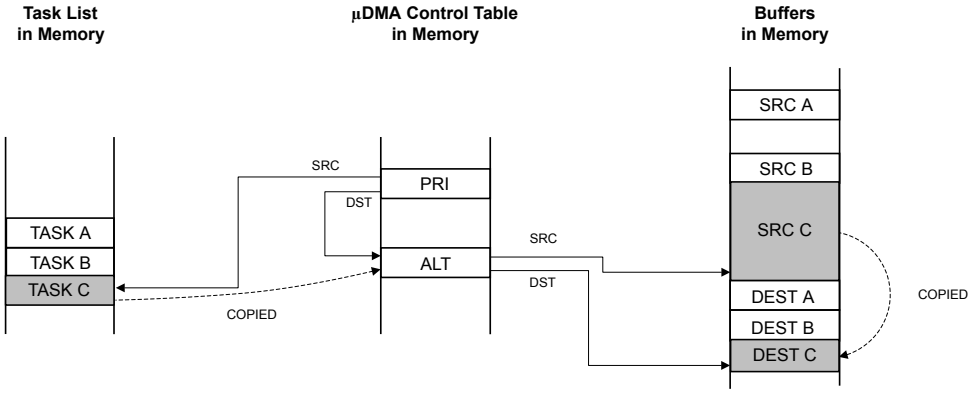

Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer A to the destination buffer.




Using the channel's primary control structure, the µDMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer B to the destination buffer.




Using the channel's primary control structure, the µDMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer C to the destination buffer.

#### 9.2.6.6 Peripheral Scatter-Gather

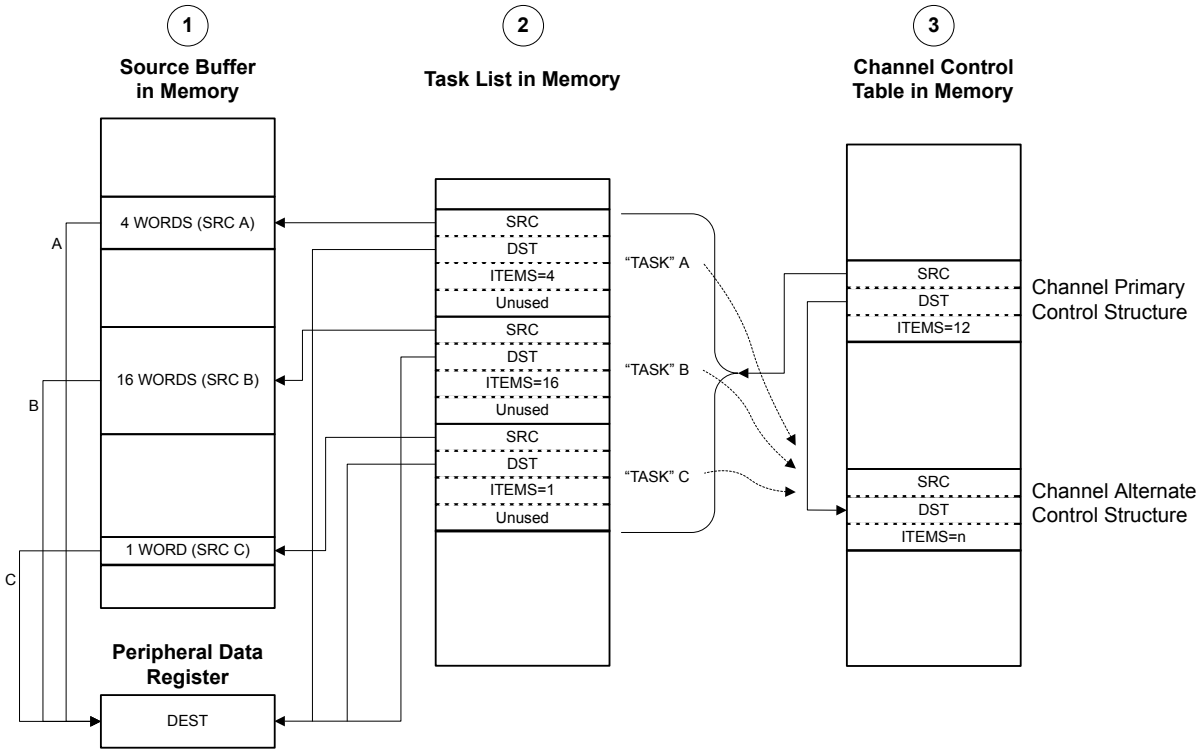
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a µDMA request. Upon detecting a request from the peripheral, the µDMA controller uses the primary control structure to copy one entry from the list to the alternate control structure and then performs the transfer. At the end of this transfer, the primary control structure will copy the next task to the alternate control structure . If the next task is a memory-to-memory transfer, execution will start immediately and run to completion; if the next task is a peripheral-type transfer, the µDMA will wait for a peripheral request to begin.

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.

Refer to Figure 9-5 on page 721 and Figure 9-6 on page 722, which show 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 9-5 on page 721 shows how the application sets up a µDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

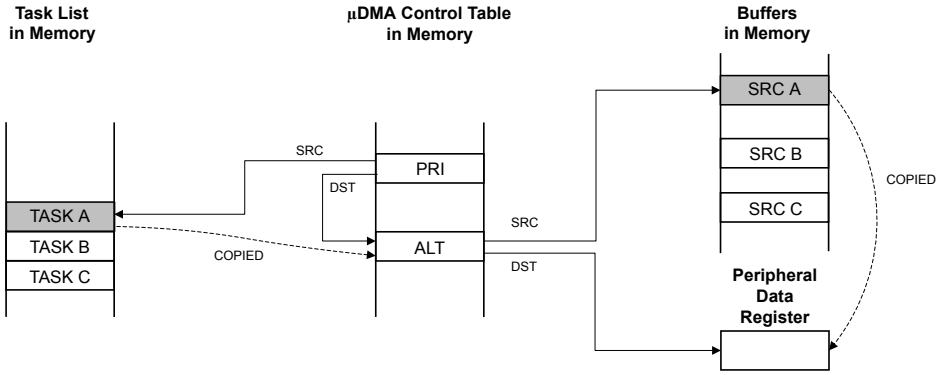
Figure 9-6 on page 722 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from 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.

**Figure 9-5. Peripheral Scatter-Gather, Setup and Configuration**



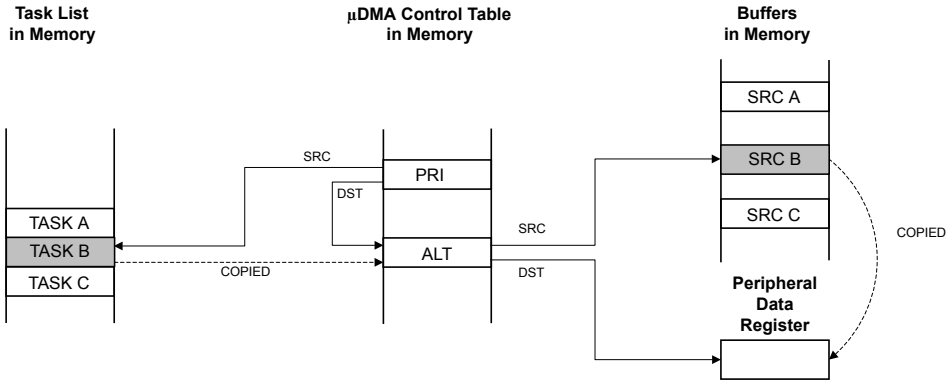

NOTES:
1. Application has a need to copy data items from three separate locations in memory into a peripheral data register.
2. Application sets up μDMA "task list" in memory, which contains the pointers and control configuration for three μDMA copy "tasks."
3. Application sets up the channel primary 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 9-6. Peripheral Scatter-Gather, µDMA Copy Sequence



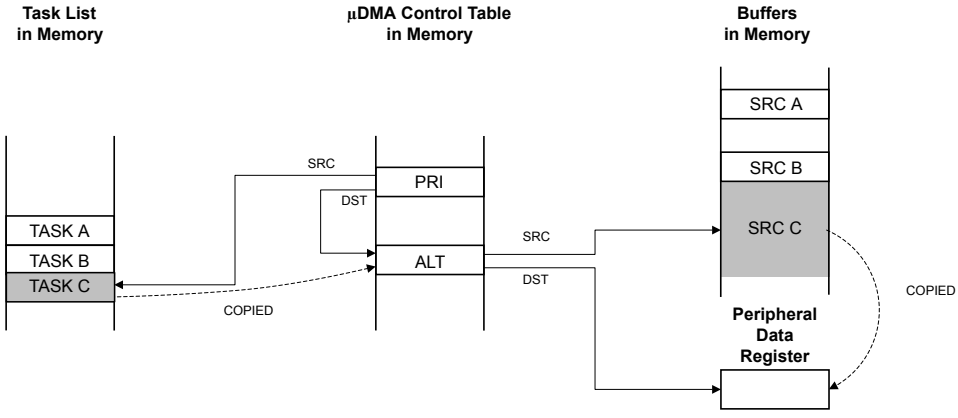

Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer A to the peripheral data register.




Using the channel's primary control structure, the µDMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer B to the peripheral data register.




Using the channel's primary control structure, the µDMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer C to the peripheral data register.

### 9.2.7 Transfer Size and Increment

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 auto-incremented by bytes, half-words, or words, or can be set to no increment. The source and destination address increment values can be set independently, and 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, but 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 9-5 shows the configuration to read from a peripheral that supplies 8-bit data.

**Table 9-5. 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 |

### 9.2.8 Peripheral Interface

There are three main classes of uDMA-connected peripherals:

- Peripherals with FIFOs serviced by the uDMA to transmit or receive data.
- Peripherals that provide trigger inputs to the uDMA

#### 9.2.8.1 FIFO Peripherals

FIFO peripherals contain a FIFO of data to be sent and a FIFO of data that has been received. The uDMA controller is used to transfer data between these FIFOs and system memory. For example, when a UART FIFO contains one or more entries, a single transfer request is sent to the uDMA for processing. If this request has not been processed and the UART FIFO reaches the interrupt FIFO level set in the **UART Interrupt FIFO Level Select (UARTIFLS)** register, another interrupt is sent to the uDMA which is higher priority than the single-transfer request. In this instance, an ARBSIZ transfer is performed as configured in the **DMACHCTL** register. After the transfer is complete, the DMA sends a receive or transmit complete interrupt to the **UART Raw Interrupt Status (UARTRIS)** register.

If the FIFO peripheral's SETn bit is set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register, then the uDMA will only perform transfers defined by the ARBSIZ bit field in the **DMACHCTL** register for better bus utilization. For peripherals that tend to transmit and receive in bursts, such as the UART, we recommend against the use of this configuration since it could cause the tail end of transmissions to stick in the FIFO.

#### 9.2.8.2 Trigger Peripherals

Certain peripherals, such as the general purpose timer, trigger an interrupt to the uDMA controller when a programmed event occurs. When a trigger event occurs, the uDMA executes a transfer defined by the ARBSIZ bit field in the **DMACHCTL** register. If only a single transfer is needed for a uDMA trigger, then the ARBSIZ field is set to 0x1.

If the trigger peripheral generates another uDMA request while the prior one is being serviced and that particular channel is the highest priority asserted channel, the second request will be processed as soon as the handling of the first is complete. If two additional trigger peripheral uDMA requests are generated prior to the completion of the first, the third request is lost.

### 9.2.9 Software Request

A transfer is initiated by software by first configuring and enabling the transfer, and then issuing a software request using the **DMA Channel Software Request (DMASWREQ)** register. For software-based transfers, the Auto transfer mode should be used.

It is possible to initiate a transfer on any available software channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral µDMA channel, then the completion interrupt occurs on the interrupt vector for the peripheral instead of the software interrupt vector. Any peripheral channel may be used for software requests as long as the corresponding peripheral is not using µDMA for data transfer.

**Note:** Channels designated in the table as only "Software" are dedicated software channels. When only one software request is required in an application, dedicated software channels can be used. If multiple software requests in code are required, then peripheral channel software requests should be used for proper µDMA completion acknowledgement.

### 9.2.10 Interrupts and Errors

Depending on the peripheral, the µDMA can indicate transfer completion at the end of an entire transfer or when a FIFO or buffer reaches a certain level (see Table 9-2 on page 712 and the individual peripheral chapters). When a µDMA transfer is complete, a dma_done signal is sent to the peripheral that initiated the µDMA event. Interrupts can be enabled within the peripheral to trigger on µDMA transfer completion. Please refer to the individual peripheral chapters for more information on peripheral µDMA interrupts. If the transfer uses the software µDMA channel, then the completion interrupt occurs on the dedicated software µDMA interrupt vector (see Table 9-6 on page 724).

If the µDMA controller encounters a bus or memory protection error as it attempts to perform a data transfer, it disables the µDMA channel that caused the error and generates an interrupt on the µDMA error interrupt vector. The processor can read the **DMA Bus Error Clear (DMAERRCLR)** register to determine if an error is pending. The ERRCLR bit is set if an error occurred. The error can be cleared by writing a 1 to the ERRCLR bit.

Table 9-6 shows the dedicated interrupt assignments for the µDMA controller.

**Table 9-6. µDMA Interrupt Assignments**

| Interrupt | Assignment |
|---|---|
| 44 | µDMA Software Channel Transfer |
| 45 | µDMA Error |

## 9.3 Initialization and Configuration

### 9.3.1 Module Initialization

Before the µDMA controller can be used, it must be enabled in the System Control block and in the peripheral. The location of the channel control structure must also be programmed.

The following steps should be performed one time during system initialization:

1. Enable the µDMA clock using the **RCGCDMA** register (see page 402).

2. Enable the µDMA controller by setting the MASTEREN bit of the **DMA Configuration (DMACFG)** register.

3. Program the location of the channel control table by writing the base address of the table to the **DMA Channel Control Base Pointer (DMACTLBASE)** register. The base address must be aligned on a 1024-byte boundary.

## 9.3.2 Configuring a Memory-to-Memory Transfer

µDMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

### 9.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Program bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.

2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.

3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.

4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

### 9.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example transfers 256 words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in Table 9-7.

**Table 9-7. Channel Control Structure Offsets for Channel 30**

| Offset | Description |
|---|---|
| Control Table Base + 0x1E0 | Channel 30 Source End Pointer |
| Control Table Base + 0x1E4 | Channel 30 Destination End Pointer |
| Control Table Base + 0x1E8 | Channel 30 Control Word |

***Configure the Source and Destination***

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Program the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.

2. Program the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 9-8.

**Table 9-8. Channel Control Word Configuration for Memory Transfer Example**

| Field in DMACHCTL | Bits | Value | Description |
|---|---|---|---|
| DSTINC | 31:30 | 2 | 32-bit destination address increment |
| DSTSIZE | 29:28 | 2 | 32-bit destination data size |
| SRCINC | 27:26 | 2 | 32-bit source address increment |
| SRCSIZE | 25:24 | 2 | 32-bit source data size |
| reserved | 23:22 | 0 | Reserved |
| DSTPROT0[a] | 21 | 0 | Privileged access protection for destination data writes |
| reserved | 20:19 | 0 | Reserved |
| SRCPROT0[a] | 18 | 0 | Privileged access protection for source data reads |
| ARBSIZE | 17:14 | 3 | Arbitrates after 8 transfers |
| XFERSIZE | 13:4 | 255 | Transfer 256 items |
| NXTUSEBURST | 3 | 0 | N/A for this transfer type |
| XFERMODE | 2:0 | 2 | Use Auto-request transfer mode |

a. The value of this bit must be 1 (privileged) for AES, DES, or SHA accesses.

***Configure Peripheral Interrupts***

For memory-to-memory transfers, the peripheral involved must be configured to generate an interrupt when the µDMA has completed its transfer. Upon completion, the µDMA will send a `dma_done` signal to the peripheral.

#### 9.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the **DMA Channel Enable Set (DMAENASET)** register.
2. Issue a transfer request by setting bit 30 of the **DMA Channel Software Request (DMASWREQ)** register.

The µDMA transfer begins. If the interrupt is enabled, then the processor is notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the **DMAENASET** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the `XFERMODE` field of the channel control word at offset 0x1E8. This field is automatically cleared at the end of the transfer.

### 9.3.3 Configuring a Peripheral for Simple Transmit

This example configures the µDMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral uses µDMA channel 7.

#### 9.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.

2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.

3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.

4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

#### 9.3.3.2 Configure the Channel Control Structure

This example transfers 64 bytes from a memory buffer to the peripheral's transmit FIFO register using µDMA channel 7. The control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 9-9.

**Table 9-9. Channel Control Structure Offsets for Channel 7**

| Offset | Description |
|---|---|
| Control Table Base + 0x070 | Channel 7 Source End Pointer |
| Control Table Base + 0x074 | Channel 7 Destination End Pointer |
| Control Table Base + 0x078 | Channel 7 Control Word |

***Configure the Source and Destination***

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Program the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.

2. Program the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to Table 9-10.

**Table 9-10. Channel Control Word Configuration for Peripheral Transmit Example**

| Field in DMACHCTL | Bits | Value | Description |
|---|---|---|---|
| DSTINC | 31:30 | 3 | Destination address does not increment |
| DSTSIZE | 29:28 | 0 | 8-bit destination data size |
| SRCINC | 27:26 | 0 | 8-bit source address increment |
| SRCSIZE | 25:24 | 0 | 8-bit source data size |
| reserved | 23:22 | 0 | Reserved |
| DSTPROT0[a] | 21 | 0 | Privileged access protection for destination data writes |
| reserved | 20:19 | 0 | Reserved |
| SRCPROT0[a] | 18 | 0 | Privileged access protection for source data reads |
| ARBSIZE | 17:14 | 2 | Arbitrates after 4 transfers |
| XFERSIZE | 13:4 | 63 | Transfer 64 items |
| NXTUSEBURST | 3 | 0 | N/A for this transfer type |
| XFERMODE | 2:0 | 1 | Use Basic transfer mode |

a. The value of this bit must be 1 (privileged) for AES, DES, or SHA accesses.

**Note:** In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst SET[7] bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

#### 9.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the **DMA Channel Enable Set (DMAENASET)** register.

The µDMA controller is now configured for transfer on channel 7. The controller makes transfers to the peripheral whenever the peripheral asserts a µDMA request. The transfers continue until the entire buffer of 64 bytes has been transferred. When that happens, the µDMA controller disables the channel and sets the XFERMODE field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the **DMA Channel Enable Set (DMAENASET)** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the XFERMODE field of the channel control word at offset 0x078. This field is automatically cleared at the end of the transfer.

If peripheral interrupts are enabled, then the peripheral generates an interrupt when the entire transfer is complete.

### 9.3.4 Configuring a Peripheral for Ping-Pong Receive

This example configures the µDMA controller to continuously receive 8-bit data from a peripheral into a pair of 64-byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral uses µDMA channel 8.

#### 9.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 8 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 8 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 8 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.
4. Set bit 8 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

#### 9.3.4.2 Configure the Channel Control Structure

This example transfers bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the µDMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the

alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 9-11.

**Table 9-11. Primary and Alternate Channel Control Structure Offsets for Channel 8**

| Offset | Description |
|---|---|
| Control Table Base + 0x080 | Channel 8 Primary Source End Pointer |
| Control Table Base + 0x084 | Channel 8 Primary Destination End Pointer |
| Control Table Base + 0x088 | Channel 8 Primary Control Word |
| Control Table Base + 0x280 | Channel 8 Alternate Source End Pointer |
| Control Table Base + 0x284 | Channel 8 Alternate Destination End Pointer |
| Control Table Base + 0x288 | Channel 8 Alternate Control Word |

### *Configure the Source and Destination*

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Program the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.
2. Program the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.
3. Program the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.
4. Program the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088 and the alternate control word at offset 0x288 are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 9-12.
2. Program the alternate channel control word at offset 0x288 according to Table 9-12.

**Table 9-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example**

| Field in DMACHCTL | Bits | Value | Description |
|---|---|---|---|
| DSTINC | 31:30 | 0 | 8-bit destination address increment |
| DSTSIZE | 29:28 | 0 | 8-bit destination data size |
| SRCINC | 27:26 | 3 | Source address does not increment |
| SRCSIZE | 25:24 | 0 | 8-bit source data size |
| reserved | 23:22 | 0 | Reserved |
| DSTPROT0 | 21 | 0 | Privileged access protection for destination data writes |
| reserved | 20:19 | 0 | Reserved |
| SRCPROT0 | 18 | 0 | Privileged access protection for source data reads |
| ARBSIZE | 17:14 | 3 | Arbitrates after 8 transfers |

**Table 9-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example *(continued)***

| Field in DMACHCTL | Bits | Value | Description |
|---|---|---|---|
| XFERSIZE | 13:4 | 63 | Transfer 64 items |
| NXTUSEBURST | 3 | 0 | N/A for this transfer type |
| XFERMODE | 2:0 | 3 | Use Ping-Pong transfer mode |

**Note:** In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst SET[8] bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

#### 9.3.4.3 Configure the Peripheral Interrupt

An interrupt handler should be configured when using µDMA Ping-Pong mode, it is best to use an interrupt handler. However, the Ping-Pong mode can be configured without interrupts by polling. The interrupt handler is triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

#### 9.3.4.4 Enable the µDMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the **DMA Channel Enable Set (DMAENASET)** register.

#### 9.3.4.5 Process Interrupts

The µDMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the µDMA request signal, the µDMA controller makes transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it switches to the alternate channel control structure and makes transfers into buffer B. At the same time, the primary channel control word mode field is configured to indicate Stopped, and an interrupt is generated in the peripheral's raw interrupt status register.

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data or set a flag that the data must be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the XFERMODE field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:

   a. Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.

   b. Reprogram the primary channel control word at offset 0x88 according to Table 9-12 on page 729.

2. Read the alternate channel control word at offset 0x288 and check the XFERMODE field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:

**a.** Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.

**b.** Reprogram the alternate channel control word at offset 0x288 according to Table 9-12 on page 729.

### 9.3.5 Configuring Channel Assignments

Channel assignments for each µDMA channel can be changed using the **DMACHMAPn** registers. Each 4-bit field represents a µDMA channel.

Refer to Table 9-1 on page 710 for channel assignments.

For example, to use UART1 RX on channel 8, configure the CH8SEL bit in the **DMACHMAP1** register to be 0x1. If a peripheral is enabled on two different channels, the µDMA channel that has the highest priority for that peripheral takes precedence. Thus, if UART 1 RX is enabled on both channel 8 and channel 22, the UART1 RX channel 22 priority needs to be lowered before channel 8 UART1 RX can be accessed by the µDMA.

## 9.4 Register Map

Table 9-13 on page 731 lists the µDMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, thus the base address is n/a (not applicable) and noted as such above the register descriptions. In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See “Channel Configuration” on page 713 and Table 9-3 on page 714 for a description of how the entries in the channel control table are located in memory. The µDMA register addresses are given as a hexadecimal increment, relative to the µDMA base address of 0x400F.F000. Note that the µDMA module clock must be enabled before the registers can be programmed (see page 402). There must be a delay of 3 system clocks after the µDMA module clock is enabled before any µDMA module registers are accessed.

**Table 9-13. µDMA Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **µDMA Channel Control Structure (Offset from Channel Control Table Base)** | | | | | |
| 0x000 | DMASRCENDP | RW | - | DMA Channel Source Address End Pointer | 733 |
| 0x004 | DMADSTENDP | RW | - | DMA Channel Destination Address End Pointer | 734 |
| 0x008 | DMACHCTL | RW | - | DMA Channel Control Word | 735 |
| **µDMA Registers (Offset from µDMA Base Address)** | | | | | |
| 0x000 | DMASTAT | RO | 0x001F.0000 | DMA Status | 740 |
| 0x004 | DMACFG | WO | - | DMA Configuration | 742 |
| 0x008 | DMACTLBASE | RW | 0x0000.0000 | DMA Channel Control Base Pointer | 743 |
| 0x00C | DMAALTBASE | RO | 0x0000.0200 | DMA Alternate Channel Control Base Pointer | 744 |
| 0x010 | DMAWAITSTAT | RO | 0x03C3.CF00 | DMA Channel Wait-on-Request Status | 745 |
| 0x014 | DMASWREQ | WO | - | DMA Channel Software Request | 746 |

**Table 9-13. µDMA Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x018 | DMAUSEBURSTSET | RW | 0x0000.0000 | DMA Channel Useburst Set | 747 |
| 0x01C | DMAUSEBURSTCLR | WO | - | DMA Channel Useburst Clear | 748 |
| 0x020 | DMAREQMASKSET | RW | 0x0000.0000 | DMA Channel Request Mask Set | 749 |
| 0x024 | DMAREQMASKCLR | WO | - | DMA Channel Request Mask Clear | 750 |
| 0x028 | DMAENASET | RW | 0x0000.0000 | DMA Channel Enable Set | 751 |
| 0x02C | DMAENACLR | WO | - | DMA Channel Enable Clear | 752 |
| 0x030 | DMAALTSET | RW | 0x0000.0000 | DMA Channel Primary Alternate Set | 753 |
| 0x034 | DMAALTCLR | WO | - | DMA Channel Primary Alternate Clear | 754 |
| 0x038 | DMAPRIOSET | RW | 0x0000.0000 | DMA Channel Priority Set | 755 |
| 0x03C | DMAPRIOCLR | WO | - | DMA Channel Priority Clear | 756 |
| 0x04C | DMAERRCLR | RW | 0x0000.0000 | DMA Bus Error Clear | 757 |
| 0x500 | DMACHASGN | RW | 0x0000.0000 | DMA Channel Assignment | 758 |
| 0x510 | DMACHMAP0 | RW | 0x0000.0000 | DMA Channel Map Select 0 | 759 |
| 0x514 | DMACHMAP1 | RW | 0x0000.0000 | DMA Channel Map Select 1 | 760 |
| 0x518 | DMACHMAP2 | RW | 0x0000.0000 | DMA Channel Map Select 2 | 761 |
| 0x51C | DMACHMAP3 | RW | 0x0000.0000 | DMA Channel Map Select 3 | 762 |
| 0xFD0 | DMAPeriphID4 | RO | 0x0000.0004 | DMA Peripheral Identification 4 | 767 |
| 0xFE0 | DMAPeriphID0 | RO | 0x0000.0030 | DMA Peripheral Identification 0 | 763 |
| 0xFE4 | DMAPeriphID1 | RO | 0x0000.00B2 | DMA Peripheral Identification 1 | 764 |
| 0xFE8 | DMAPeriphID2 | RO | 0x0000.000B | DMA Peripheral Identification 2 | 765 |
| 0xFEC | DMAPeriphID3 | RO | 0x0000.0000 | DMA Peripheral Identification 3 | 766 |
| 0xFF0 | DMAPCellID0 | RO | 0x0000.000D | DMA PrimeCell Identification 0 | 768 |
| 0xFF4 | DMAPCellID1 | RO | 0x0000.00F0 | DMA PrimeCell Identification 1 | 769 |
| 0xFF8 | DMAPCellID2 | RO | 0x0000.0005 | DMA PrimeCell Identification 2 | 770 |
| 0xFFC | DMAPCellID3 | RO | 0x0000.00B1 | DMA PrimeCell Identification 3 | 771 |

## 9.5 µDMA Channel Control Structure

The µDMA Channel Control Structure holds the transfer settings for a µDMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 713 for an explanation of the Channel Control Table and the Channel Control Structure.

The channel control structure is one entry in the channel control table. Each channel has a primary and alternate structure. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

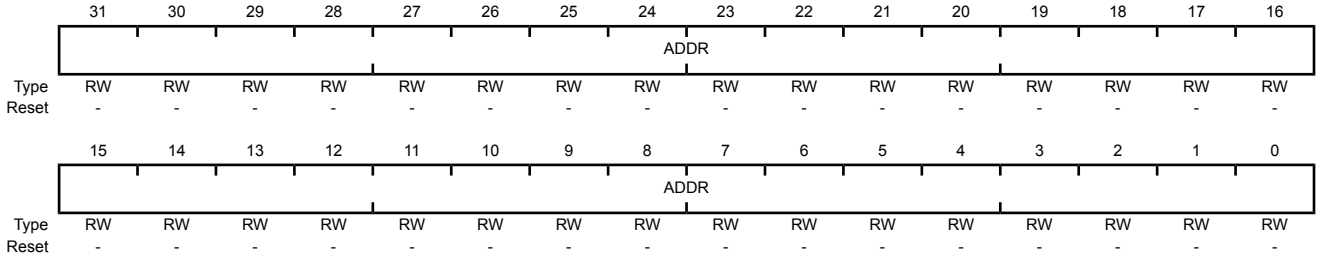
## Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

**DMA Channel Source Address End Pointer (DMASRCENDP)** is part of the Channel Control Structure and is used to specify the source address for a µDMA transfer.

**Note:** The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Source Address End Pointer (DMASRCENDP)

Base n/a
Offset 0x000
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | RW | - | Source Address End Pointer<br>This field points to the last address of the µDMA transfer source (inclusive). If the source address is not incrementing (the SRCINC field in the **DMACHCTL** register is 0x3), then this field points at the source location itself (such as a peripheral data register). |

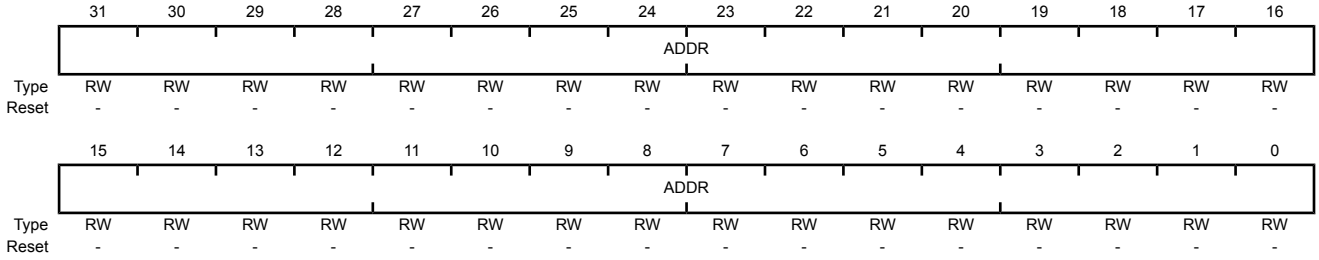
## Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

**DMA Channel Destination Address End Pointer (DMADSTENDP)** is part of the Channel Control Structure and is used to specify the destination address for a µDMA transfer.

**Note:** The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a
Offset 0x004
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | RW | - | Destination Address End Pointer<br>This field points to the last address of the µDMA transfer destination (inclusive). If the destination address is not incrementing (the DSTINC field in the **DMACHCTL** register is 0x3), then this field points at the destination location itself (such as a peripheral data register). |

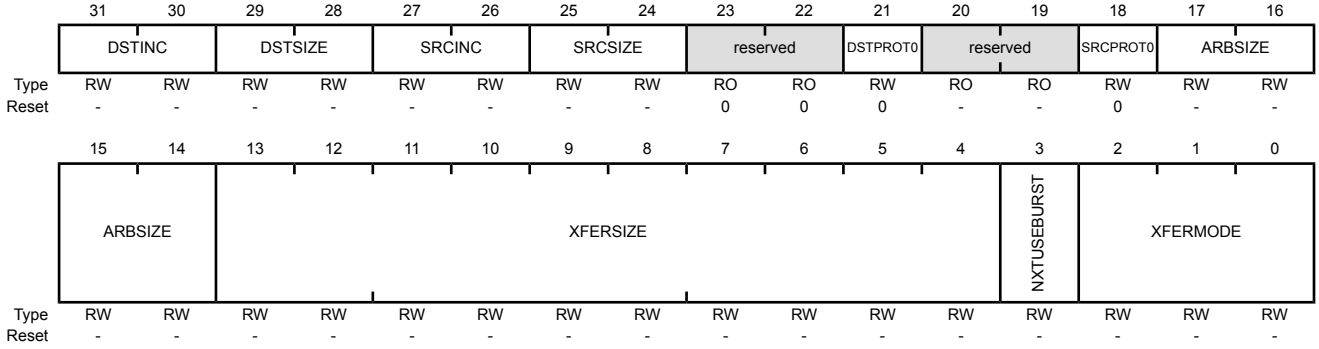
## Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

**DMA Channel Control Word (DMACHCTL)** is part of the Channel Control Structure and is used to specify parameters of a µDMA transfer.

**Note:** The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Control Word (DMACHCTL)

Base n/a
Offset 0x008
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:30 | DSTINC | RW | - | Destination Address Increment<br>This field configures the destination address increment.<br>The address increment value must be equal or greater than the value of the destination size (DSTSIZE).<br><br>Value Description<br>0x0 Byte<br>Increment by 8-bit locations<br>0x1 Half-word<br>Increment by 16-bit locations<br>0x2 Word<br>Increment by 32-bit locations<br>0x3 No increment<br>Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 29:28 | DSTSIZE | RW | - | Destination Data Size<br>This field configures the destination item data size.<br>**Note:** DSTSIZE must be the same as SRCSIZE.<br>Value Description<br>0x0 Byte<br>8-bit data size<br>0x1 Half-word<br>16-bit data size<br>0x2 Word<br>32-bit data size<br>0x3 Reserved |
| 27:26 | SRCINC | RW | - | Source Address Increment<br>This field configures the source address increment.<br>The address increment value must be equal or greater than the value of the source size (SRCSIZE).<br>Value Description<br>0x0 Byte<br>Increment by 8-bit locations<br>0x1 Half-word<br>Increment by 16-bit locations<br>0x2 Word<br>Increment by 32-bit locations<br>0x3 No increment<br>Address remains set to the value of the Source Address End Pointer (DMASRCENDP) for the channel |
| 25:24 | SRCSIZE | RW | - | Source Data Size<br>This field configures the source item data size.<br>**Note:** DSTSIZE must be the same as SRCSIZE.<br>Value Description<br>0x0 Byte<br>8-bit data size.<br>0x1 Half-word<br>16-bit data size.<br>0x2 Word<br>32-bit data size.<br>0x3 Reserved |
| 23:22 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 21 | DSTPROT0 | RW | 0 | Destination Privilege Access<br>This bit controls the privilege access protection for destination data writes.<br>**Note:** For AES,DES, or SHA accesses, this bit must be set to 1.<br>Value Description<br>0 The access is non-privileged.<br>1 The access is privileged. |
| 20:19 | reserved | RO | - | 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. |
| 18 | SRCPROT0 | RW | 0 | Source Privilege Access<br>This bit controls the privilege access protection for source data reads.<br>**Note:** For AES,DES, or SHA accesses, this bit must be set to 1.<br>Value Description<br>0 The access is non-privileged.<br>1 The access is privileged. |
| 17:14 | ARBSIZE | RW | - | Arbitration Size<br>This field configures the number of transfers that can occur before the µDMA controller re-arbitrates. The possible arbitration rate configurations represent powers of 2 and are shown below.<br>Value Description<br>0x0 1 Transfer<br>Arbitrates after each µDMA transfer<br>0x1 2 Transfers<br>0x2 4 Transfers<br>0x3 8 Transfers<br>0x4 16 Transfers<br>0x5 32 Transfers<br>0x6 64 Transfers<br>0x7 128 Transfers<br>0x8 256 Transfers<br>0x9 512 Transfers<br>0xA-0xF 1024 Transfers<br>In this configuration, no arbitration occurs during the µDMA transfer because the maximum transfer size is 1024. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13:4 | XFERSIZE | RW | - | Transfer Size (minus 1)<br>This field configures the total number of items to transfer. The value of this field is 1 less than the number to transfer (value 0 means transfer 1 item). The maximum value for this 10-bit field is 1023 which represents a transfer size of 1024 items.<br>The transfer size is the number of items, not the number of bytes. If the data size is 32 bits, then this value is the number of 32-bit words to transfer.<br>The µDMA controller updates this field immediately prior to entering the arbitration process, so it contains the number of outstanding items that is necessary to complete the µDMA cycle. |
| 3 | NXTUSEBURST | RW | - | Next Useburst<br>This field controls whether the Useburst SET[n] bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the µDMA controller uses single transfers to complete the transaction. If this bit is set, then the controller uses a burst transfer to complete the last transfer. |
| 2:0 | XFERMODE | RW | - | µDMA Transfer Mode<br>This field configures the operating mode of the µDMA cycle. Refer to “Transfer Modes” on page 715 for a detailed explanation of transfer modes.<br>Because this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled.<br>Value Description<br>0x0 Stop<br>0x1 Basic<br>0x2 Auto-Request<br>0x3 Ping-Pong<br>0x4 Memory Scatter-Gather<br>0x5 Alternate Memory Scatter-Gather<br>0x6 Peripheral Scatter-Gather<br>0x7 Alternate Peripheral Scatter-Gather |

**XFERMODE Bit Field Values.**

Stop
Channel is stopped or configuration data is invalid. No more transfers can occur.

Basic
For each trigger (whether from a peripheral or a software request), the µDMA controller performs the number of transfers specified by the ARBSIZE field.

Auto-Request
The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of XFERSIZE items without any further requests.

Ping-Pong
: This mode uses both the primary and alternate control structures for this channel. When the number of transfers specified by the XFERSIZE field have completed for the current control structure (primary or alternate), the µDMA controller switches to the other one. These switches continue until one of the control structures is not set to ping-pong mode. At that point, the µDMA controller stops. An interrupt is generated on completion of the transfers configured by each control structure. See “Ping-Pong” on page 715.

Memory Scatter-Gather
: When using this mode, the primary control structure for the channel is configured to allow a list of operations (tasks) to be performed. The source address pointer specifies the start of a table of tasks to be copied to the alternate control structure for this channel. The XFERMODE field for the alternate control structure should be configured to 0x5 (Alternate memory scatter-gather) to perform the task. When the task completes, the µDMA switches back to the primary channel control structure, which then copies the next task to the alternate control structure. This process continues until the table of tasks is empty. The last task must have an XFERMODE value other than 0x5. Note that for continuous operation, the last task can update the primary channel control structure back to the start of the list or to another list. See “Memory Scatter-Gather” on page 717.

Alternate Memory Scatter-Gather
: This value must be used in the alternate channel control data structure when the µDMA controller operates in Memory Scatter-Gather mode.

Peripheral Scatter-Gather
: This value must be used in the primary channel control data structure when the µDMA controller operates in Peripheral Scatter-Gather mode. In this mode, the µDMA controller operates exactly the same as in Memory Scatter-Gather mode, except that instead of performing the number of transfers specified by the XFERSIZE field in the alternate control structure at one time, the µDMA controller only performs the number of transfers specified by the ARBSIZE field per trigger; see Basic mode for details. See “Peripheral Scatter-Gather” on page 720.

Alternate Peripheral Scatter-Gather
: This value must be used in the alternate channel control data structure when the µDMA controller operates in Peripheral Scatter-Gather mode.

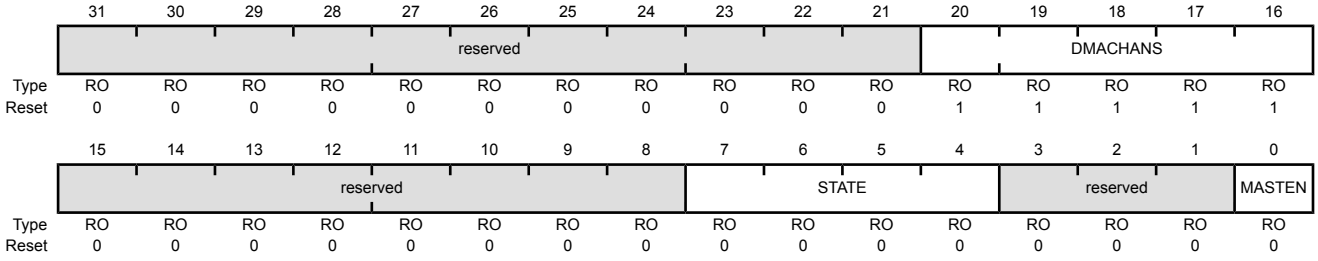
## 9.6 µDMA Register Descriptions

The register addresses given are relative to the µDMA base address of 0x400F.F000.

## Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the µDMA controller. You cannot read this register when the µDMA controller is in the reset state.

DMA Status (DMASTAT)
Base 0x400F.F000
Offset 0x000
Type RO, reset 0x001F.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:21 | reserved | RO | 0x000 | 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. |
| 20:16 | DMACHANS | RO | 0x1F | Available µDMA Channels Minus 1<br>This field contains a value equal to the number of µDMA channels the µDMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 µDMA channels. |
| 15:8 | reserved | RO | 0x00 | 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. |
| 7:4 | STATE | RO | 0x0 | Control State Machine Status<br>This field shows the current status of the control state machine. Status can be one of the following. |
| 3:1 | reserved | RO | 0x0 | 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. |

| Value | Description |
|---|---|
| 0x0 | Idle |
| 0x1 | Reading channel controller data. |
| 0x2 | Reading source end pointer. |
| 0x3 | Reading destination end pointer. |
| 0x4 | Reading source data. |
| 0x5 | Writing destination data. |
| 0x6 | Waiting for µDMA request to clear. |
| 0x7 | Writing channel controller data. |
| 0x8 | Stalled |
| 0x9 | Done |
| 0xA-0xF | Undefined |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | MASTEN | RO | 0 | Master Enable Status |

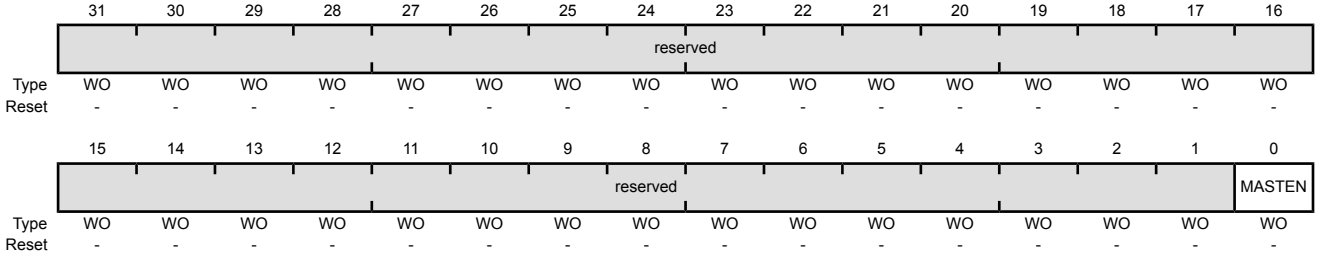
| Value | Description |
|---|---|
| 0 | The µDMA controller is disabled. |
| 1 | The µDMA controller is enabled. |

## Register 5: DMA Configuration (DMACFG), offset 0x004

The **DMACFG** register controls the configuration of the µDMA controller.

DMA Configuration (DMACFG)

Base 0x400F.F000
Offset 0x004
Type WO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | WO | - | 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. |
| 0 | MASTEN | WO | - | Controller Master Enable<br><br>Value Description<br>0 Disables the µDMA controller.<br>1 Enables µDMA controller. |

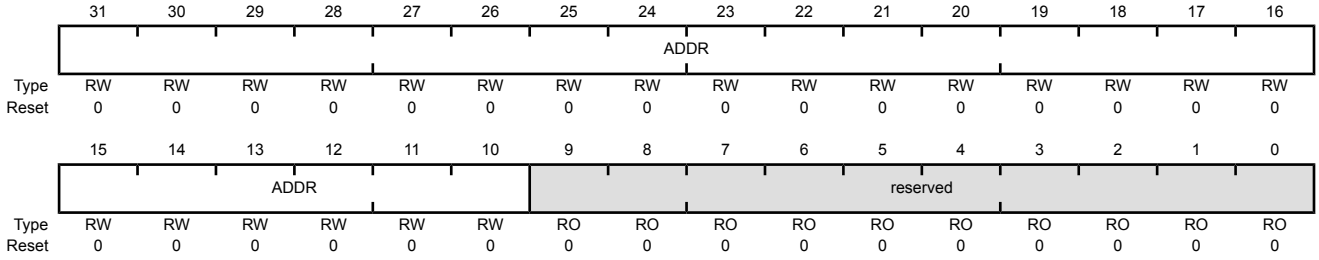
## Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

The **DMACTLBASE** register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that must be assigned to the µDMA controller depends on the number of µDMA channels used and whether the alternate channel control data structure is used. See “Channel Configuration” on page 713 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. This register cannot be read when the µDMA controller is in the reset state.

DMA Channel Control Base Pointer (DMACTLBASE)

Base 0x400F.F000
Offset 0x008
Type RW, reset 0x0000.0000



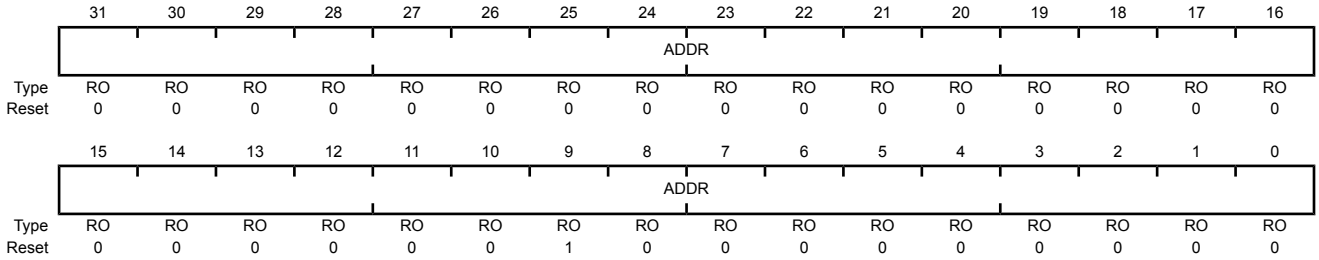

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | ADDR | RW | 0x0000.00 | Channel Control Base Address<br>This field contains the pointer to the base address of the channel control table. The base address must be 1024-byte aligned. |
| 9:0 | reserved | RO | 0x00 | Software 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. |

## Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C

The **DMAALTBASE** register returns the base address of the alternate channel control data. This register removes the necessity for application software to calculate the base address of the alternate channel control structures. This register cannot be read when the µDMA controller is in the reset state.

DMA Alternate Channel Control Base Pointer (DMAALTBASE)

Base 0x400F.F000
Offset 0x00C
Type RO, reset 0x0000.0200



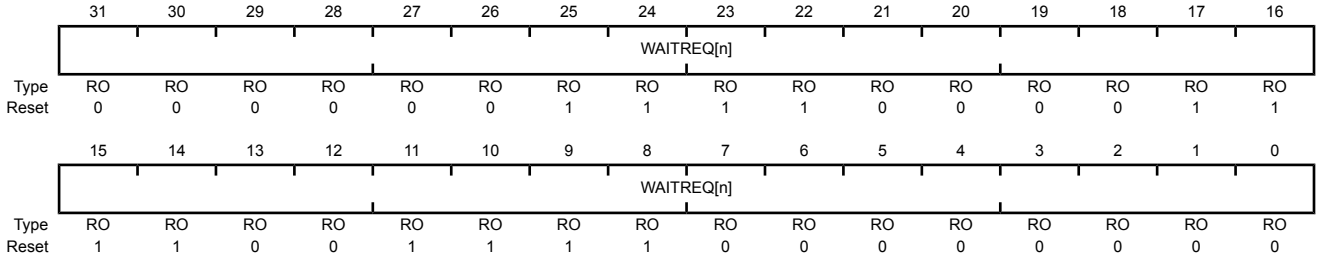

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | RO | 0x0000.0200 | Alternate Channel Address Pointer<br>This field provides the base address of the alternate channel control structures. |

## Register 8: DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010

This read-only register indicates that the µDMA channel is waiting on a request. A peripheral can hold off the µDMA from performing a single request until the peripheral is ready for a burst request to enhance the µDMA performance. The use of this feature is dependent on the design of the peripheral and is not controllable by software in any way. This register cannot be read when the µDMA controller is in the reset state.

DMA Channel Wait-on-Request Status (DMAWAITSTAT)

Base 0x400F.F000
Offset 0x010
Type RO, reset 0x03C3.CF00



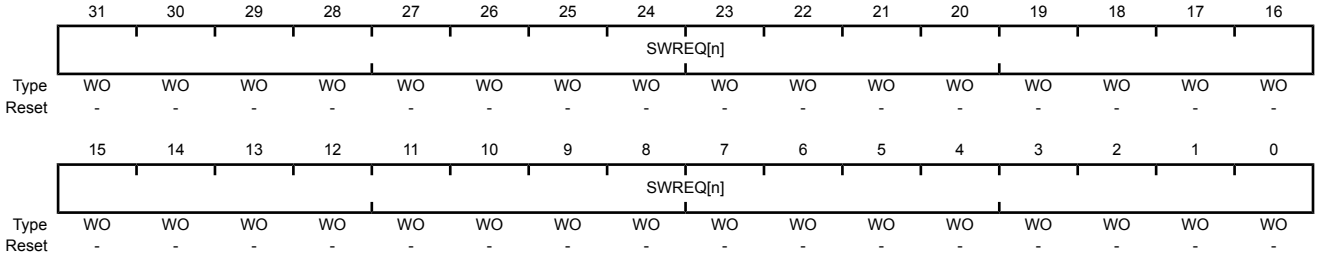

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WAITREQ[n] | RO | 0x03C3.CF00 | Channel [n] Wait Status<br>These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0.<br><br>Value Description<br>0 The corresponding channel is not waiting on a request.<br>1 The corresponding channel is waiting on a request. |

## Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014

Each bit of the **DMASWREQ** register represents the corresponding µDMA channel. Setting a bit generates a request for the specified µDMA channel.

DMA Channel Software Request (DMASWREQ)

Base 0x400F.F000
Offset 0x014
Type WO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SWREQ[n] | WO | - | Channel [n] Software Request<br>These bits generate software requests. Bit 0 corresponds to channel 0.<br><br>Value Description<br>0 No request generated.<br>1 Generate a software request for the corresponding channel.<br><br>These bits are automatically cleared when the software request has been completed. |

## Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

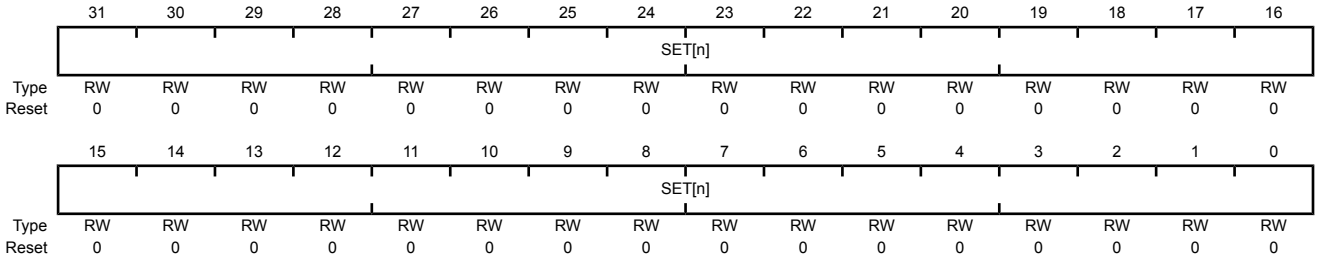
Each bit of the **DMAUSEBURSTSET** register represents the corresponding µDMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

If the amount of data to transfer is a multiple of the arbitration (burst) size, the corresponding SET[n] bit is cleared after completing the final transfer. If there are fewer items remaining to transfer than the arbitration (burst) size, the µDMA controller automatically clears the corresponding SET[n] bit, allowing the remaining items to transfer using single requests. In order to resume transfers using burst requests, the corresponding bit must be set again. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to “Request Types” on page 712 for more details about request types.

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | RW | 0x0000.0000 | Channel [n] Useburst Set |

| Value | Description |
|---|---|
| 0 | µDMA channel [n] responds to single or burst requests. |
| 1 | µDMA channel [n] responds only to burst requests. |

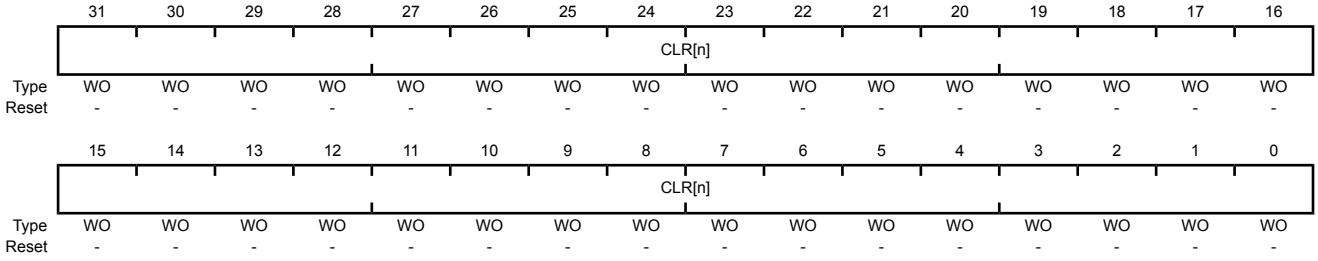
Bit 0 corresponds to channel 0. This bit is automatically cleared as described above. A bit can also be manually cleared by setting the corresponding CLR[n] bit in the **DMAUSEBURSTCLR** register.

## Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C

Each bit of the **DMAUSEBURSTCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAUSEBURSTSET** register.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000
Offset 0x01C
Type WO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CLR[n] | WO | - | Channel [n] Useburst Clear |

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAUSEBURSTSET** register meaning that µDMA channel [n] responds to single and burst requests. |

## Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020

Each bit of the **DMAREQMASKSET** register represents the corresponding µDMA channel. Setting a bit disables µDMA requests for the channel. Reading the register returns the request mask status. When a µDMA channel's request is masked, that means the peripheral can no longer request µDMA transfers. The channel can then be used for software-initiated transfers.

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
Offset 0x020
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SET[n] | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SET[n] | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | RW | 0x0000.0000 | Channel [n] Request Mask Set |

| Value | Description |
|---|---|
| 0 | The peripheral associated with channel [n] is enabled to request µDMA transfers. |
| 1 | The peripheral associated with channel [n] is not able to request µDMA transfers. Channel [n] may be used for software-initiated transfers. |

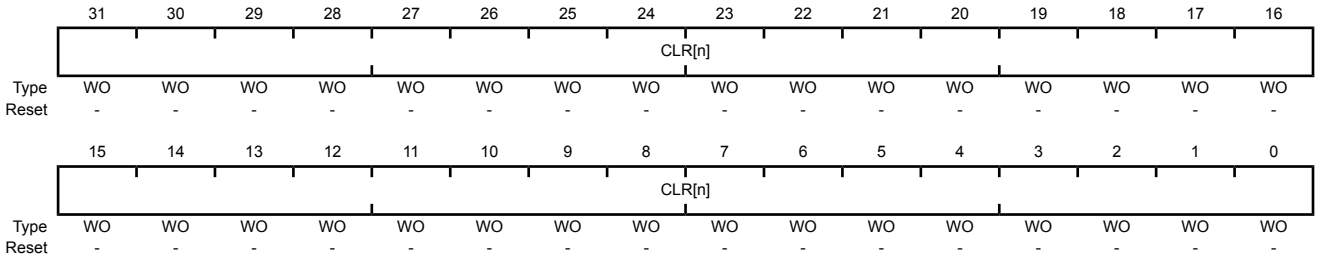
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAREQMASKCLR** register.

## Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

Each bit of the **DMAREQMASKCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAREQMASKSET** register.

DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000
Offset 0x024
Type WO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CLR[n] | WO | - | Channel [n] Request Mask Clear |

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAREQMASKSET** register meaning that the peripheral associated with channel [n] is enabled to request µDMA transfers. |

## Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding µDMA channel. Setting a bit enables the corresponding µDMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
Offset 0x028
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SET[n] | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SET[n] | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | RW | 0x0000.0000 | Channel [n] Enable Set |

| Value | Description |
|---|---|
| 0 | µDMA Channel [n] is disabled. |
| 1 | µDMA Channel [n] is enabled. |

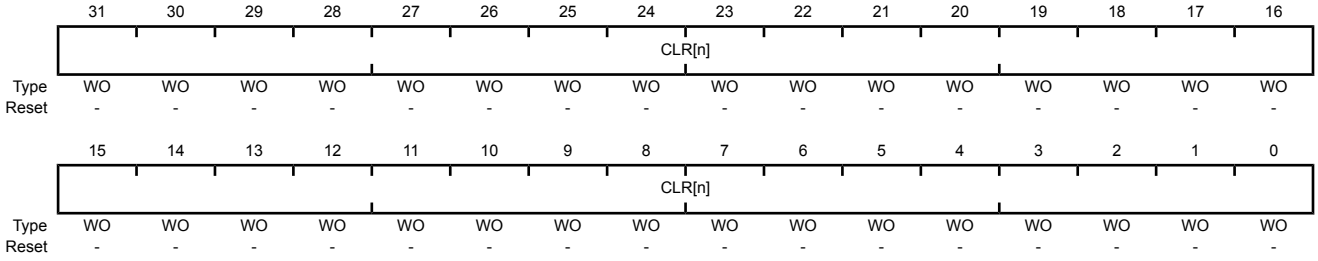
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAENACLR** register or when the end of a µDMA transfer occurs.

## Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C

Each bit of the **DMAENACLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAENASET** register.

DMA Channel Enable Clear (DMAENACLR)

Base 0x400F.F000
Offset 0x02C
Type WO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CLR[n] | WO | - | Clear Channel [n] Enable Clear |

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAENASET** register meaning that channel [n] is disabled for µDMA transfers. |

**Note:** The controller disables a channel when it completes the µDMA cycle.

## Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding µDMA channel. Setting a bit configures the µDMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding µDMA channel.

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
Offset 0x030
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SET[n] | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SET[n] | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | RW | 0x0000.0000 | Channel [n] Alternate Set |

| Value | Description |
|---|---|
| 0 | µDMA channel [n] is using the primary control structure. |
| 1 | µDMA channel [n] is using the alternate control structure. |

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAALTCLR** register.

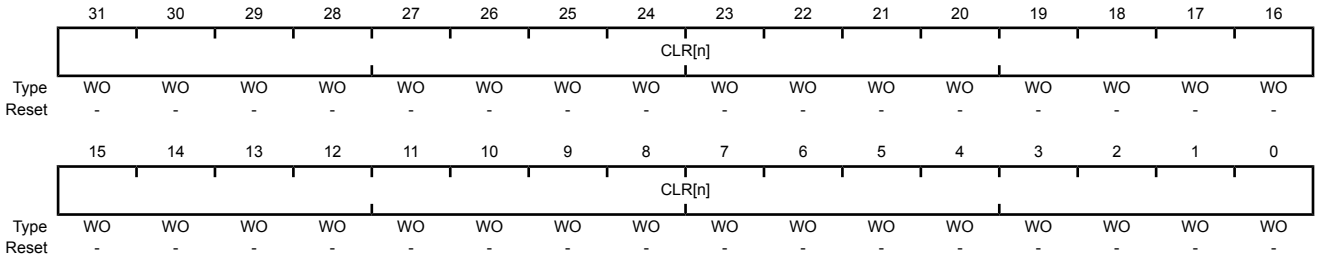
**Note:** For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.

## Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

Each bit of the **DMAALTCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAALTSET** register.

DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000
Offset 0x034
Type WO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CLR[n] | WO | - | Channel [n] Alternate Clear |

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAALTSET** register meaning that channel [n] is using the primary control structure. |

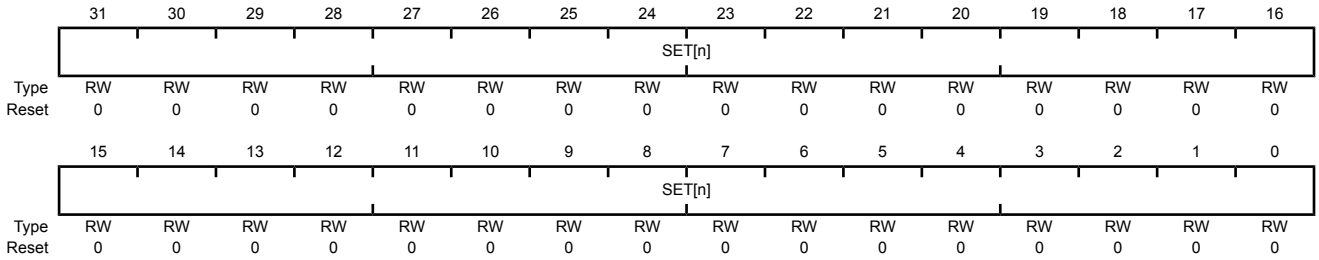
**Note:** For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.

## Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the **DMAPRIOSET** register represents the corresponding µDMA channel. Setting a bit configures the µDMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
Offset 0x038
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | RW | 0x0000.0000 | Channel [n] Priority Set |

| Value | Description |
|---|---|
| 0 | µDMA channel [n] is using the default priority level. |
| 1 | µDMA channel [n] is using a high priority level. |

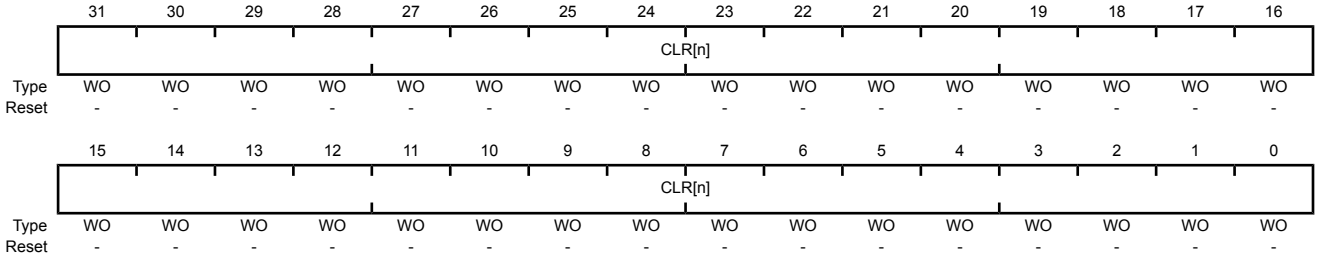
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAPRIOCLR** register.

## Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

Each bit of the **DMAPRIOCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding SET[n] bit in the **DMAPRIOSET** register.

DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000
Offset 0x03C
Type WO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CLR[n] | WO | - | Channel [n] Priority Clear |

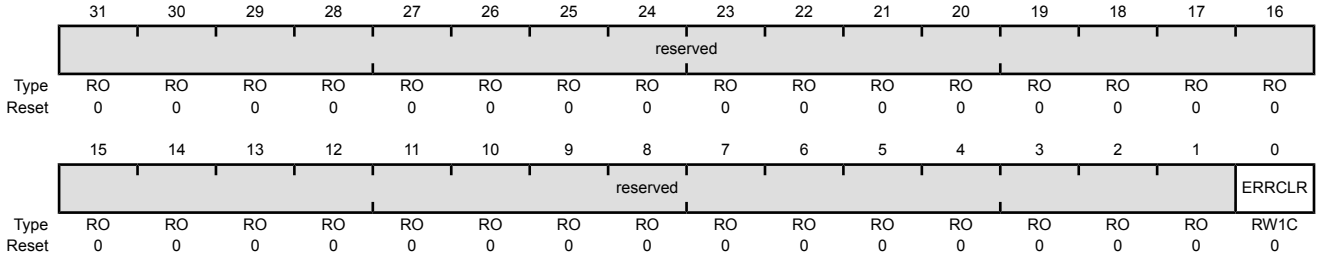
| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAPRIOSET** register meaning that channel [n] is using the default priority level. |

## Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the µDMA bus error status. The error status is set if the µDMA controller encountered a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the µDMA controller. The other channels are unaffected.

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
Offset 0x04C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | ERRCLR | RW1C | 0 | µDMA Bus Error Status<br><br>Value Description<br>0 No bus error is pending.<br>1 A bus error is pending.<br><br>This bit is cleared by writing a 1 to it. |

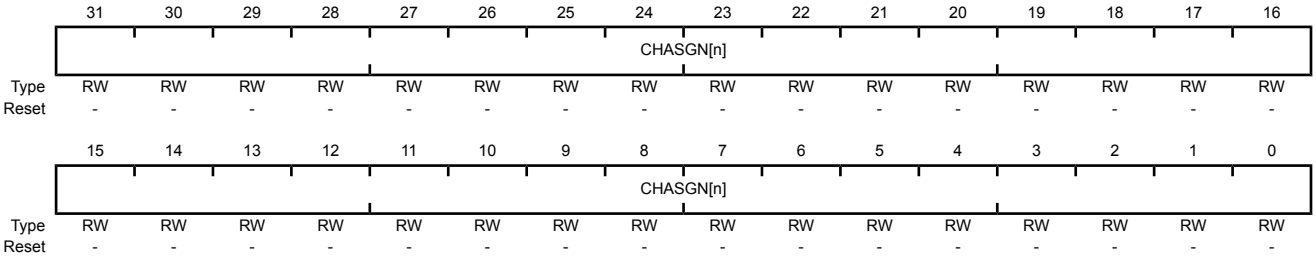
## Register 21: DMA Channel Assignment (DMACHASGN), offset 0x500

Each bit of the **DMACHASGN** register represents the corresponding µDMA channel. Setting a bit selects the secondary channel assignment as specified in Table 9-1 on page 710.

**Note:** This register is provided to support legacy software. New software should use the **DMACHMAPn** registers. If a bit is clear in this register, the corresponding field in the **DMACHMAPn** registers is configured to 0x0. If a bit is set in this register, the corresponding field is configured to 0x1. If this register is read, a bit reads as 0 if the corresponding **DMACHMAPn** register field value is equal to 0, otherwise it reads as 1 if the corresponding **DMACHMAPn** register field value is not equal to 0.

DMA Channel Assignment (DMACHASGN)

Base 0x400F.F000
Offset 0x500
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CHASGN[n] | RW | - | Channel [n] Assignment Select |

| Value | Description |
|---|---|
| 0 | Use the primary channel assignment. |
| 1 | Use the secondary channel assignment. |

## Register 22: DMA Channel Map Select 0 (DMACHMAP0), offset 0x510

Each 4-bit field of the **DMACHMAP0** register configures the µDMA channel assignment as specified in Table 9-1 on page 710.

**Note:** To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 0 (DMACHMAP0)

Base 0x400F.F000
Offset 0x510
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH7SEL | | | | CH6SEL | | | | CH5SEL | | | | CH4SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH3SEL | | | | CH2SEL | | | | CH1SEL | | | | CH0SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | CH7SEL | RW | 0x00 | µDMA Channel 7 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 27:24 | CH6SEL | RW | 0x00 | µDMA Channel 6 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 23:20 | CH5SEL | RW | 0x00 | µDMA Channel 5 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 19:16 | CH4SEL | RW | 0x00 | µDMA Channel 4 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 15:12 | CH3SEL | RW | 0x00 | µDMA Channel 3 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 11:8 | CH2SEL | RW | 0x00 | µDMA Channel 2 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 7:4 | CH1SEL | RW | 0x00 | µDMA Channel 1 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 3:0 | CH0SEL | RW | 0x00 | µDMA Channel 0 Source Select<br>See Table 9-1 on page 710 for channel assignments. |

## Register 23: DMA Channel Map Select 1 (DMACHMAP1), offset 0x514

Each 4-bit field of the **DMACHMAP1** register configures the µDMA channel assignment as specified in Table 9-1 on page 710.

**Note:** To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 1 (DMACHMAP1)

Base 0x400F.F000
Offset 0x514
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH15SEL | | | | CH14SEL | | | | CH13SEL | | | | CH12SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH11SEL | | | | CH10SEL | | | | CH9SEL | | | | CH8SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | CH15SEL | RW | 0x00 | µDMA Channel 15 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 27:24 | CH14SEL | RW | 0x00 | µDMA Channel 14 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 23:20 | CH13SEL | RW | 0x00 | µDMA Channel 13 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 19:16 | CH12SEL | RW | 0x00 | µDMA Channel 12 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 15:12 | CH11SEL | RW | 0x00 | µDMA Channel 11 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 11:8 | CH10SEL | RW | 0x00 | µDMA Channel 10 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 7:4 | CH9SEL | RW | 0x00 | µDMA Channel 9 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 3:0 | CH8SEL | RW | 0x00 | µDMA Channel 8 Source Select<br>See Table 9-1 on page 710 for channel assignments. |

## Register 24: DMA Channel Map Select 2 (DMACHMAP2), offset 0x518

Each 4-bit field of the **DMACHMAP2** register configures the µDMA channel assignment as specified in Table 9-1 on page 710.

**Note:** To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 2 (DMACHMAP2)

Base 0x400F.F000
Offset 0x518
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH23SEL | | | | CH22SEL | | | | CH21SEL | | | | CH20SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH19SEL | | | | CH18SEL | | | | CH17SEL | | | | CH16SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | CH23SEL | RW | 0x00 | µDMA Channel 23 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 27:24 | CH22SEL | RW | 0x00 | µDMA Channel 22 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 23:20 | CH21SEL | RW | 0x00 | µDMA Channel 21 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 19:16 | CH20SEL | RW | 0x00 | µDMA Channel 20 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 15:12 | CH19SEL | RW | 0x00 | µDMA Channel 19 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 11:8 | CH18SEL | RW | 0x00 | µDMA Channel 18 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 7:4 | CH17SEL | RW | 0x00 | µDMA Channel 17 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 3:0 | CH16SEL | RW | 0x00 | µDMA Channel 16 Source Select<br>See Table 9-1 on page 710 for channel assignments. |

## Register 25: DMA Channel Map Select 3 (DMACHMAP3), offset 0x51C

Each 4-bit field of the **DMACHMAP3** register configures the µDMA channel assignment as specified in Table 9-1 on page 710.

**Note:** To support legacy software which uses the **DMA Channel Assignment (DMACHASGN)** register, a value of 0x0 is equivalent to a **DMACHASGN** bit being clear, and a value of 0x1 is equivalent to a **DMACHASGN** bit being set.

DMA Channel Map Select 3 (DMACHMAP3)

Base 0x400F.F000
Offset 0x51C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH31SEL | | | | CH30SEL | | | | CH29SEL | | | | CH28SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CH27SEL | | | | CH26SEL | | | | CH25SEL | | | | CH24SEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

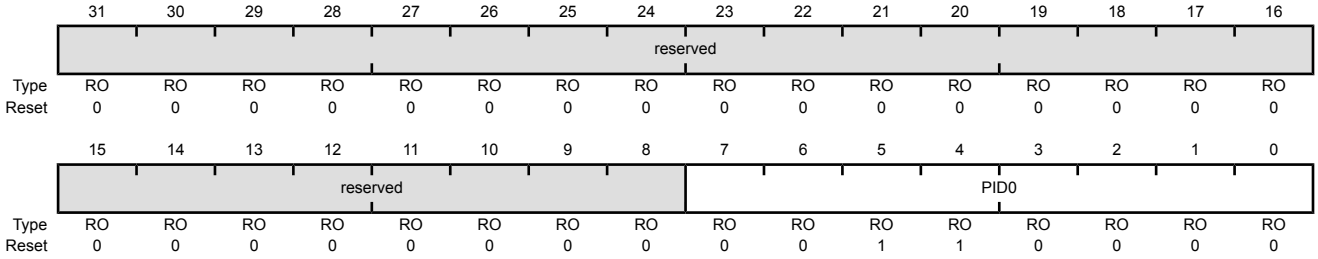
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | CH31SEL | RW | 0x00 | µDMA Channel 31 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 27:24 | CH30SEL | RW | 0x00 | µDMA Channel 30 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 23:20 | CH29SEL | RW | 0x00 | µDMA Channel 29 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 19:16 | CH28SEL | RW | 0x00 | µDMA Channel 28 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 15:12 | CH27SEL | RW | 0x00 | µDMA Channel 27 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 11:8 | CH26SEL | RW | 0x00 | µDMA Channel 26 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 7:4 | CH25SEL | RW | 0x00 | µDMA Channel 25 Source Select<br>See Table 9-1 on page 710 for channel assignments. |
| 3:0 | CH24SEL | RW | 0x00 | µDMA Channel 24 Source Select<br>See Table 9-1 on page 710 for channel assignments. |

## Register 26: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 0 (DMAPeriphID0)

Base 0x400F.F000
Offset 0xFE0
Type RO, reset 0x0000.0030



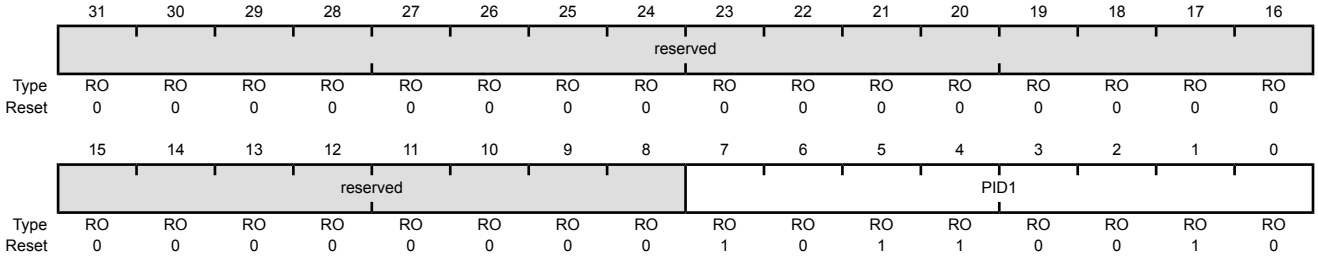

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x30 | µDMA Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 27: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 1 (DMAPeriphID1)

Base 0x400F.F000
Offset 0xFE4
Type RO, reset 0x0000.00B2



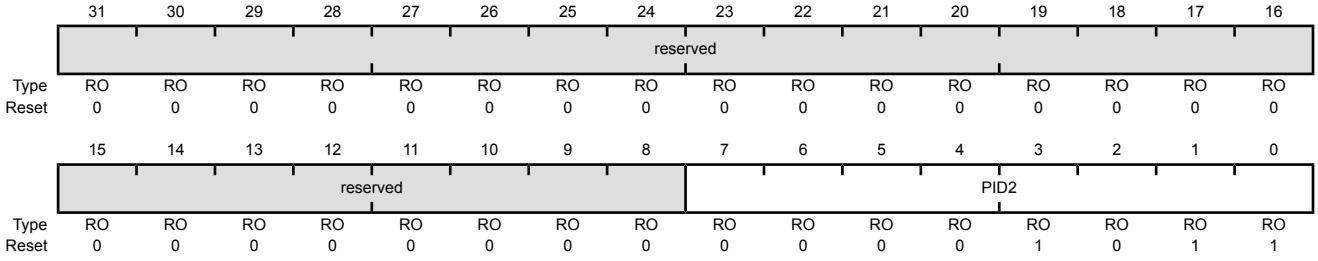

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0xB2 | µDMA Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 28: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000
Offset 0xFE8
Type RO, reset 0x0000.000B



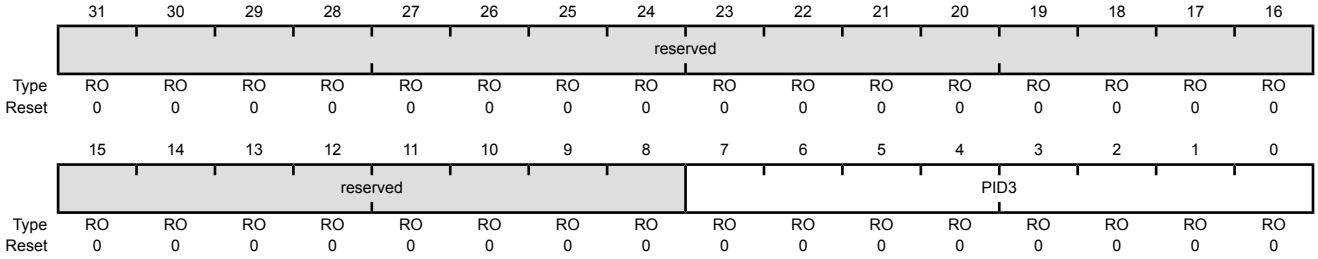

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x0B | µDMA Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 29: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 3 (DMAPeriphID3)

Base 0x400F.F000
Offset 0xFEC
Type RO, reset 0x0000.0000



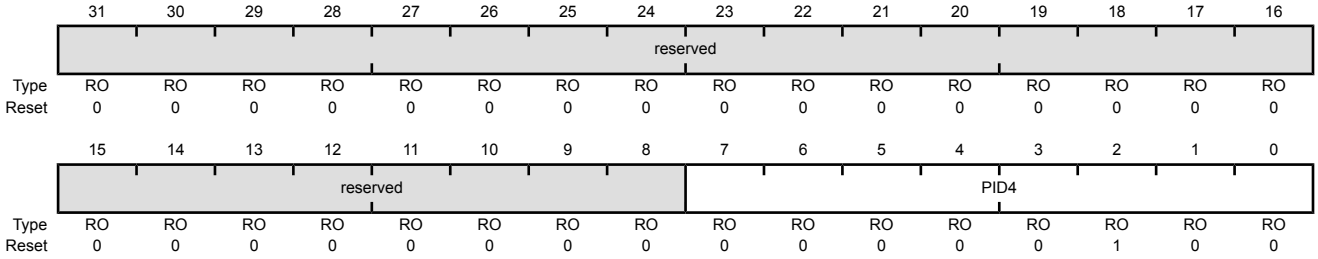

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x00 | µDMA Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 30: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 4 (DMAPeriphID4)

Base 0x400F.F000
Offset 0xFD0
Type RO, reset 0x0000.0004



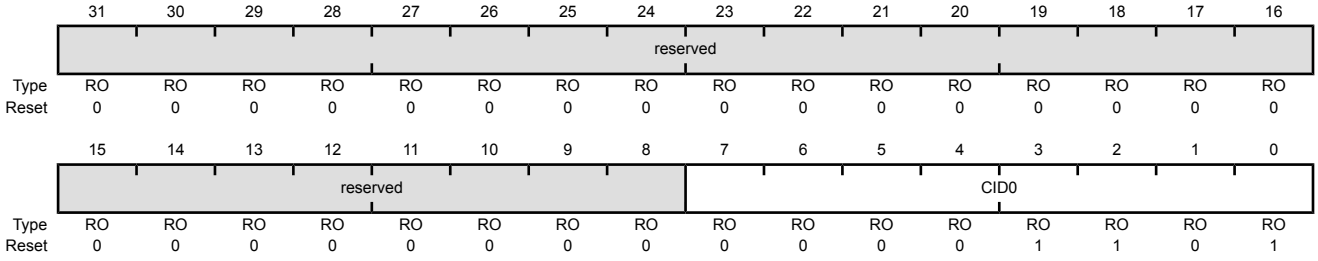

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x04 | µDMA Peripheral ID Register<br>Can be used by software to identify the presence of this peripheral. |

## Register 31: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 0 (DMAPCellID0)

Base 0x400F.F000
Offset 0xFF0
Type RO, reset 0x0000.000D



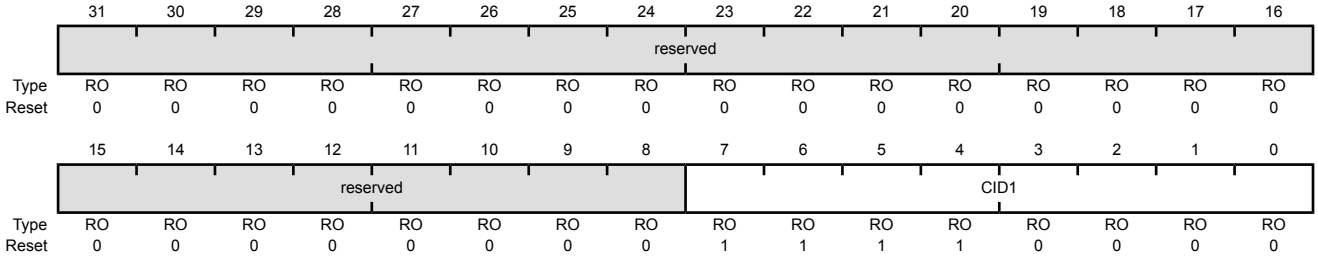

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | µDMA PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 32: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 1 (DMAPCellID1)

Base 0x400F.F000
Offset 0xFF4
Type RO, reset 0x0000.00F0



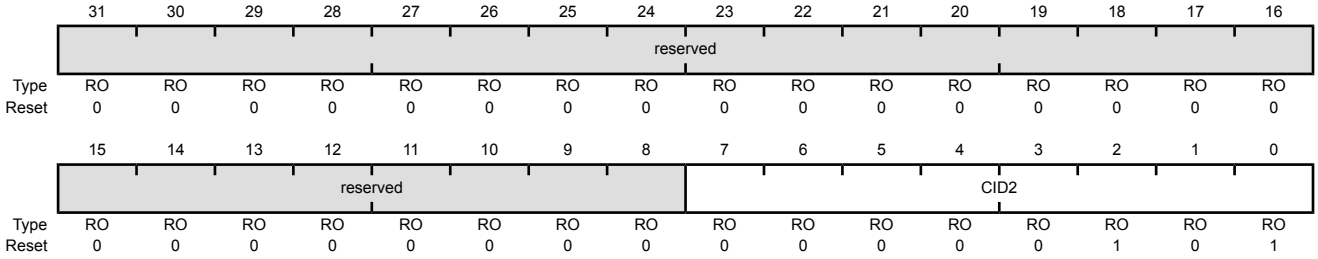

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | µDMA PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 33: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 2 (DMAPCellID2)

Base 0x400F.F000
Offset 0xFF8
Type RO, reset 0x0000.0005



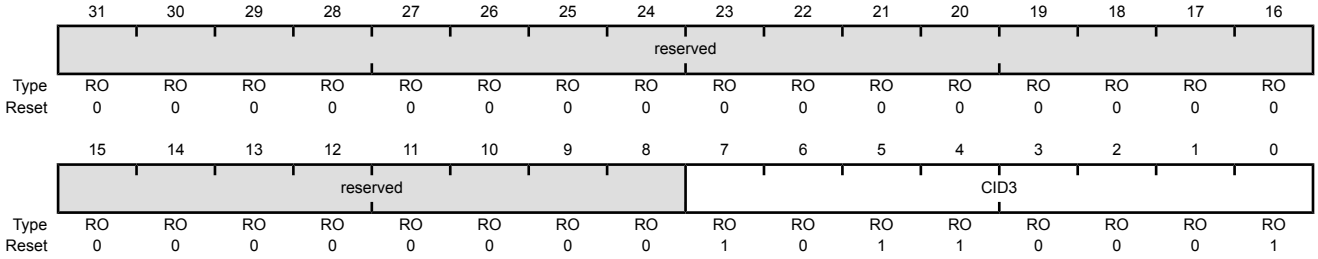

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | µDMA PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 34: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 3 (DMAPCellID3)

Base 0x400F.F000
Offset 0xFFC
Type RO, reset 0x0000.00B1




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x00 | 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. |
| 7:0 | CID3 | RO | 0xB1 | µDMA PrimeCell ID Register [31:24]<br>Provides software a standard cross-peripheral identification system. |

# 10 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G, Port H, Port J, Port K, Port L, Port M, Port N, Port P, Port Q, Port R, Port S, Port T). The GPIO module supports up to 140 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 140 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 3.3-V-tolerant in input configuration
- Advanced High Performance Bus accesses all ports:
  - Ports A-H and J; Ports K-N and P-T
- Fast toggle capable of a change every clock cycle for ports on AHB
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
  - Per-pin interrupts available on Port P and Port Q
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence or a µDMA transfer
- Pin state can be retained during Hibernation mode; pins on port P can be programmed to wake on level in Hibernation mode
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, 6-mA, 8-mA, 10-mA and 12-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for 8-mA, 10-mA and 12-mA pad drive
  - Open drain enables
  - Digital input enables

## 10.1 Signal Description

GPIO signals have alternate hardware functions. The following table lists the GPIO pins and their analog and digital alternate functions. The digital alternate hardware functions are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the PMCx bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric encoding shown in the table below. Analog signals in the table below are also 3.3-V tolerant and are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. The AINx analog signals have internal circuitry to protect them from voltages over $V_{DD}$ (up to the maximum specified in Table 32-1 on page 2095), but analog performance specifications are only guaranteed if the input signal swing at the I/O pad is kept inside the range 0 V < $V_{IN}$ < $V_{DD}$. Note that each pin must be programmed individually; no type of grouping is implied by the columns in the table. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

**Important:** The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 10-1. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 784.

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

**Table 10-2. GPIO Pins and Alternate Functions (212BGA)**

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PA0 | V3 | - | U0Rx | I2C9SCL | T0CCP0 | - | - | - | CAN0Rx | - | - | - | - | - |
| PA1 | W3 | - | U0Tx | I2C9SDA | T0CCP1 | - | - | - | CAN0Tx | - | - | - | - | - |
| PA2 | T6 | - | U4Rx | I2C8SCL | T1CCP0 | - | - | - | - | - | - | - | - | SSI0Clk |
| PA3 | U5 | - | U4Tx | I2C8SDA | T1CCP1 | - | - | - | - | - | - | - | - | SSI0Fss |
| PA4 | V4 | - | U3Rx | I2C7SCL | T2CCP0 | - | - | - | - | - | - | - | - | SSI0XDAT0 |

**Table 10-2. GPIO Pins and Alternate Functions (212BGA) *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PA5 | W4 | - | U3Tx | I2C7SDA | T2CCP1 | - | - | - | - | - | - | - | - | SSI0XDAT1 |
| PA6 | V5 | - | U2Rx | I2C6SCL | T3CCP0 | - | USB0EPEN | - | - | - | - | SSI0XDAT2 | EN0RXCK | EPI0S8 |
| PA7 | R7 | - | U2Tx | I2C6SDA | T3CCP1 | - | USB0PFLT | - | - | - | USB0EPEN | SSI0XDAT3 | - | EPI0S9 |
| PB0 | A16 | USB0ID | U1Rx | I2C5SCL | T4CCP0 | - | - | - | CAN1Rx | - | - | - | - | - |
| PB1 | B16 | USB0VBUS | U1Tx | I2C5SDA | T4CCP1 | - | - | - | CAN1Tx | - | - | - | - | - |
| PB2 | A17 | - | - | I2C0SCL | T5CCP0 | - | EN0MDC | - | - | - | - | - | USB0STP | EPI0S27 |
| PB3 | B17 | - | - | I2C0SDA | T5CCP1 | - | EN0MDIO | - | - | - | - | - | USB0CLK | EPI0S28 |
| PB4 | C6 | AIN10 | U0CTS | I2C5SCL | - | - | - | - | - | - | - | - | - | SSI1Fss |
| PB5 | B6 | AIN11 | U0RTS | I2C5SDA | - | - | - | - | - | - | - | - | - | SSI1Clk |
| PB6 | F2 | - | - | I2C6SCL | T6CCP0 | - | - | - | - | - | - | - | - | - |
| PB7 | F1 | - | - | I2C6SDA | T6CCP1 | - | - | - | - | - | - | - | - | - |
| PC0 | B15 | - | TCK SWCLK | - | - | - | - | - | - | - | - | - | - | - |
| PC1 | C15 | - | TMS SWDIO | - | - | - | - | - | - | - | - | - | - | - |
| PC2 | D14 | - | TDI | - | - | - | - | - | - | - | - | - | - | - |
| PC3 | C14 | - | TDO SWO | - | - | - | - | - | - | - | - | - | - | - |
| PC4 | M2 | C1- | U7Rx | - | T7CCP0 | - | - | - | - | - | - | - | - | EPI0S7 |
| PC5 | M1 | C1+ | U7Tx | - | T7CCP1 | - | - | - | RTCCLK | - | - | - | - | EPI0S6 |
| PC6 | L2 | C0+ | U5Rx | - | - | - | - | - | - | - | - | - | - | EPI0S5 |
| PC7 | K3 | C0- | U5Tx | - | - | - | - | - | - | - | - | - | - | EPI0S4 |
| PD0 | C2 | AIN15 | - | I2C7SCL | T0CCP0 | - | C0o | - | - | - | - | - | - | SSI2XDAT1 |
| PD1 | C1 | AIN14 | - | I2C7SDA | T0CCP1 | - | C1o | - | - | - | - | - | - | SSI2XDAT0 |
| PD2 | D2 | AIN13 | - | I2C8SCL | T1CCP0 | - | C2o | - | - | - | - | - | - | SSI2Fss |
| PD3 | D1 | AIN12 | - | I2C8SDA | T1CCP1 | - | - | - | - | - | - | - | - | SSI2Clk |
| PD4 | A4 | AIN7 | U2Rx | - | T3CCP0 | - | - | - | - | - | - | - | - | SSI1XDAT2 |
| PD5 | B4 | AIN6 | U2Tx | - | T3CCP1 | - | - | - | - | - | - | - | - | SSI1XDAT3 |
| PD6 | B3 | AIN5 | U2RTS | - | T4CCP0 | - | USB0EPEN | - | - | - | - | - | - | SSI2XDAT3 |
| PD7 | B2 | AIN4 | U2CTS | - | T4CCP1 | - | USB0PFLT | - | - | NMI | - | - | - | SSI2XDAT2 |
| PE0 | H3 | AIN3 | U1RTS | - | - | - | - | - | - | - | - | - | - | - |
| PE1 | H2 | AIN2 | U1DSR | - | - | - | - | - | - | - | - | - | - | - |
| PE2 | G1 | AIN1 | U1DCD | - | - | - | - | - | - | - | - | - | - | - |
| PE3 | G2 | AIN0 | U1DTR | - | - | - | OWIRE | - | - | - | - | - | - | - |
| PE4 | A5 | AIN9 | U1RI | - | - | - | - | - | - | - | - | - | - | SSI1XDAT0 |
| PE5 | B5 | AIN8 | - | - | - | - | - | - | - | - | - | - | - | SSI1XDAT1 |

**Table 10-2. GPIO Pins and Alternate Functions (212BGA) *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PE6 | A7 | AIN20 | U0CTS | I2C9SCL | - | - | - | - | - | - | - | - | - | - |
| PE7 | B7 | AIN21 | U0RTS | I2C9SDA | - | - | - | - | - | NMI | - | - | - | - |
| PF0 | U6 | - | - | - | - | - | EN0LED0 | M0PWM0 | - | - | - | - | SSI3XDAT1 | TRD2 |
| PF1 | V6 | - | - | - | - | - | EN0LED2 | M0PWM1 | - | - | - | - | SSI3XDAT0 | TRD1 |
| PF2 | W6 | - | - | - | - | - | EN0MDC | M0PWM2 | - | - | - | - | SSI3Fss | TRD0 |
| PF3 | T7 | - | - | - | - | - | EN0MDIO | M0PWM3 | - | - | - | - | SSI3Clk | TRCLK |
| PF4 | V7 | - | - | - | - | - | EN0LED1 | M0FAULT0 | - | - | - | - | SSI3XDAT2 | TRD3 |
| PF5 | W7 | - | - | - | - | - | - | - | - | - | - | - | SSI3XDAT3 | - |
| PF6 | T8 | - | - | - | - | - | - | - | - | - | - | - | - | LCDMCLK |
| PF7 | U8 | - | - | - | - | - | - | - | - | - | - | - | - | LCDDATA02 |
| PG0 | N15 | - | - | I2C1SCL | - | - | EN0PPS | M0PWM4 | - | - | - | - | - | EPI0S11 |
| PG1 | T14 | - | - | I2C1SDA | - | - | - | M0PWM5 | - | - | - | - | - | EPI0S10 |
| PG2 | V11 | - | - | I2C2SCL | - | - | - | - | - | - | - | - | EN0TXCK | SSI2XDAT3 |
| PG3 | M16 | - | - | I2C2SDA | - | - | - | - | - | - | - | - | EN0TXEN | SSI2XDAT2 |
| PG4 | K17 | - | U0CTS | I2C3SCL | - | - | OWIRE | - | - | - | - | - | EN0TXD0 | SSI2XDAT1 |
| PG5 | K15 | - | U0RTS | I2C3SDA | - | - | OWALT | - | - | - | - | - | EN0TXD1 | SSI2XDAT0 |
| PG6 | V12 | - | - | I2C4SCL | - | - | OWIRE | - | - | - | - | - | EN0RXER | SSI2Fss |
| PG7 | U14 | - | - | I2C4SDA | - | - | OWIRE | - | - | - | - | - | EN0RXDV | SSI2Clk |
| PH0 | P4 | - | U0RTS | - | - | - | - | - | - | - | - | - | - | EPI0S0 |
| PH1 | R2 | - | U0CTS | - | - | - | - | - | - | - | - | - | - | EPI0S1 |
| PH2 | R1 | - | U0DCD | - | - | - | - | - | - | - | - | - | - | EPI0S2 |
| PH3 | T1 | - | U0DSR | - | - | - | - | - | - | - | - | - | - | EPI0S3 |
| PH4 | R3 | - | U0DTR | - | - | - | - | - | - | - | - | - | - | - |
| PH5 | T2 | - | U0RI | - | - | - | EN0PPS | - | - | - | - | - | - | - |
| PH6 | U2 | - | U5Rx | U7Rx | - | - | - | - | - | - | - | - | - | - |
| PH7 | V2 | - | U5Tx | U7Tx | - | - | - | - | - | - | - | - | - | - |
| PJ0 | C8 | - | U3Rx | - | - | - | EN0PPS | - | - | - | - | - | - | - |
| PJ1 | E7 | - | U3Tx | - | - | - | - | - | - | - | - | - | - | - |
| PJ2 | H17 | - | U2RTS | - | - | - | - | - | - | - | - | - | - | LCDDATA14 |
| PJ3 | F16 | - | U2CTS | - | - | - | - | - | - | - | - | - | - | LCDDATA15 |
| PJ4 | F18 | - | U3RTS | - | - | - | - | - | - | - | - | - | - | LCDDATA16 |
| PJ5 | E17 | - | U3CTS | - | - | - | - | - | - | - | - | - | - | LCDDATA17 |
| PJ6 | N1 | - | U4RTS | - | - | - | - | - | - | - | - | - | - | LCDAC |
| PJ7 | K5 | - | U4CTS | - | - | - | - | - | - | - | - | - | - | - |

**Table 10-2. GPIO Pins and Alternate Functions (212BGA) *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PK0 | J1 | AIN16 | U4Rx | - | - | - | - | - | - | - | - | - | - | EPI0S0 |
| PK1 | J2 | AIN17 | U4Tx | - | - | - | - | - | - | - | - | - | - | EPI0S1 |
| PK2 | K1 | AIN18 | U4RTS | - | - | - | - | - | - | - | - | - | - | EPI0S2 |
| PK3 | K2 | AIN19 | U4CTS | - | - | - | - | - | - | - | - | - | - | EPI0S3 |
| PK4 | U19 | - | - | I2C3SCL | - | - | EN0LED0 | M0PWM6 | EN0INTRN | - | - | - | EN0RXD3 | EPI0S32 |
| PK5 | V17 | - | - | I2C3SDA | - | - | EN0LED2 | M0PWM7 | - | - | - | - | EN0RXD2 | EPI0S31 |
| PK6 | V16 | - | - | I2C4SCL | - | - | EN0LED1 | M0FAULT1 | - | - | - | - | EN0TXD2 | EPI0S25 |
| PK7 | W16 | - | U0RI | I2C4SDA | - | - | RTCCLK | M0FAULT2 | - | - | - | - | EN0TXD3 | EPI0S24 |
| PL0 | G16 | - | - | I2C2SDA | - | - | - | M0FAULT3 | - | - | - | - | USB0D0 | EPI0S16 |
| PL1 | H19 | - | - | I2C2SCL | - | - | - | PhA0 | - | - | - | - | USB0D1 | EPI0S17 |
| PL2 | G18 | - | - | - | - | - | C0o | PhB0 | - | - | - | - | USB0D2 | EPI0S18 |
| PL3 | J18 | - | - | - | - | - | C1o | IDX0 | - | - | - | - | USB0D3 | EPI0S19 |
| PL4 | H18 | - | - | - | T0CCP0 | - | - | - | - | - | - | - | USB0D4 | EPI0S26 |
| PL5 | G19 | - | - | - | T0CCP1 | - | - | - | - | - | - | - | USB0D5 | EPI0S33 |
| PL6 | C18 | USB0DP | - | - | T1CCP0 | - | - | - | - | - | - | - | - | - |
| PL7 | B18 | USB0DM | - | - | T1CCP1 | - | - | - | - | - | - | - | - | - |
| PM0 | K18 | - | - | - | T2CCP0 | - | - | - | - | - | - | - | - | EPI0S15 |
| PM1 | K19 | - | - | - | T2CCP1 | - | - | - | - | - | - | - | - | EPI0S14 |
| PM2 | L18 | - | - | - | T3CCP0 | - | - | - | - | - | - | - | - | EPI0S13 |
| PM3 | L19 | - | - | - | T3CCP1 | - | - | - | - | - | - | - | - | EPI0S12 |
| PM4 | M18 | TMPR3 | U0CTS | - | T4CCP0 | - | - | - | - | - | - | - | EN0RREF_CLK | - |
| PM5 | G15 | TMPR2 | U0DCD | - | T4CCP1 | - | - | - | - | - | - | - | - | - |
| PM6 | N19 | TMPR1 | U0DSR | - | T5CCP0 | - | - | - | - | - | - | - | EN0CRS | - |
| PM7 | N18 | TMPR0 | U0RI | - | T5CCP1 | - | - | - | - | - | - | - | EN0COL | - |
| PN0 | C10 | - | U1RTS | - | - | - | - | - | - | - | - | - | - | - |
| PN1 | B11 | - | U1CTS | - | - | - | - | - | - | - | - | - | - | - |
| PN2 | A11 | - | U1DCD | U2RTS | - | - | - | - | - | - | - | - | - | EPI0S29 |
| PN3 | B10 | - | U1DSR | U2CTS | - | - | - | - | - | - | - | - | - | EPI0S30 |
| PN4 | A10 | - | U1DTR | U3RTS | I2C2SDA | - | - | - | - | - | - | - | - | EPI0S34 |
| PN5 | B9 | - | U1RI | U3CTS | I2C2SCL | - | - | - | - | - | - | - | - | EPI0S35 |
| PN6 | T12 | - | - | U4RTS | - | - | - | - | - | - | - | - | EN0TXER | LCDDATA13 |
| PN7 | U12 | - | U1RTS | U4CTS | - | - | - | - | - | - | - | - | - | LCDDATA12 |
| PP0 | D6 | C2+ | U6Rx | - | - | - | T6CCP0 | - | EN0INTRN | - | - | - | - | SSI3XDAT2 |
| PP1 | D7 | C2- | U6Tx | - | - | - | T6CCP1 | - | - | - | - | - | - | SSI3XDAT3 |

**Table 10-2. GPIO Pins and Alternate Functions (212BGA) *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PP2 | B13 | - | U0DTR | - | - | - | - | - | - | - | - | - | USB0NXT | EPI0S29 |
| PP3 | C12 | - | U1CTS | U0DCD | - | - | - | - | RTCCLK | - | - | - | USB0DIR | EPI0S30 |
| PP4 | D8 | - | U3RTS | U0DSR | - | OWIRE | - | - | - | - | - | - | USB0D7 | - |
| PP5 | B12 | - | U3CTS | I2C2SCL | - | OWALT | - | - | - | - | - | - | USB0D6 | - |
| PP6 | B8 | AIN23 | U1DCD | I2C2SDA | - | - | - | - | - | - | - | - | - | - |
| PP7 | A8 | AIN22 | - | - | - | - | OWIRE | - | - | - | - | - | - | - |
| PQ0 | E3 | - | - | - | T6CCP0 | - | - | - | - | - | - | - | SSI3Clk | EPI0S20 |
| PQ1 | E2 | - | - | - | T6CCP1 | - | - | - | - | - | - | - | SSI3Fss | EPI0S21 |
| PQ2 | H4 | - | - | - | T7CCP0 | - | - | - | - | - | - | - | SSI3XDAT0 | EPI0S22 |
| PQ3 | M4 | - | - | - | T7CCP1 | - | - | - | - | - | - | - | SSI3XDAT1 | EPI0S23 |
| PQ4 | A13 | - | U1Rx | - | - | - | - | - | DIVSCLK | - | - | - | - | - |
| PQ5 | W12 | - | U1Tx | - | - | - | - | - | - | - | - | - | EN0RXD0 | - |
| PQ6 | U15 | - | U1DTR | - | - | - | - | - | - | - | - | - | EN0RXD1 | - |
| PQ7 | M3 | - | U1RI | - | - | - | - | - | - | - | - | - | - | - |
| PR0 | N5 | - | U4Tx | I2C1SCL | - | - | - | M0PWM0 | - | - | - | - | - | LCDCP |
| PR1 | N4 | - | U4Rx | I2C1SDA | - | - | - | M0PWM1 | - | - | - | - | - | LCDFP |
| PR2 | N2 | - | - | I2C2SCL | - | - | - | M0PWM2 | - | - | - | - | - | LCDLP |
| PR3 | V8 | - | - | I2C2SDA | - | - | - | M0PWM3 | - | - | - | - | - | LCDDATA03 |
| PR4 | P3 | - | - | I2C3SCL | T0CCP0 | - | - | M0PWM4 | - | - | - | - | - | LCDDATA00 |
| PR5 | P2 | - | U1Rx | I2C3SDA | T0CCP1 | - | - | M0PWM5 | - | - | - | - | - | LCDDATA01 |
| PR6 | W9 | - | U1Tx | I2C4SCL | T1CCP0 | - | - | M0PWM6 | - | - | - | - | - | LCDDATA04 |
| PR7 | R10 | - | - | I2C4SDA | T1CCP1 | - | - | M0PWM7 | - | - | - | - | EN0TXEN | LCDDATA05 |
| PS0 | D12 | - | - | - | T2CCP0 | - | - | M0FAULT0 | - | - | - | - | - | LCDDATA20 |
| PS1 | D13 | - | - | - | T2CCP1 | - | - | M0FAULT1 | - | - | - | - | - | LCDDATA21 |
| PS2 | B14 | - | U1DSR | - | T3CCP0 | - | - | M0FAULT2 | - | - | - | - | - | LCDDATA22 |
| PS3 | A14 | - | - | - | T3CCP1 | - | - | M0FAULT3 | - | - | - | - | - | LCDDATA23 |
| PS4 | V9 | - | - | - | T4CCP0 | - | - | PhA0 | - | - | - | - | EN0TXD0 | LCDDATA06 |
| PS5 | T13 | - | - | - | T4CCP1 | - | - | PhB0 | - | - | - | - | EN0TXD1 | LCDDATA07 |
| PS6 | U10 | - | - | - | T5CCP0 | - | - | IDX0 | - | - | - | - | EN0RXER | LCDDATA08 |
| PS7 | R13 | - | - | - | T5CCP1 | - | - | - | - | - | - | - | EN0RXDV | LCDDATA09 |
| PT0 | W10 | - | - | - | T6CCP0 | - | - | - | CAN0Rx | - | - | - | EN0RXD0 | LCDDATA10 |
| PT1 | V10 | - | - | - | T6CCP1 | - | - | - | CAN0Tx | - | - | - | EN0RXD1 | LCDDATA11 |
| PT2 | E18 | - | - | - | T7CCP0 | - | - | - | CAN1Rx | - | - | - | - | LCDDATA18 |

**Table 10-2. GPIO Pins and Alternate Functions (212BGA) *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PT3 | F17 | - | - | - | T7CCP1 | - | - | - | CAN1Tx | - | - | - | - | LCDDATA19 |

a. The TMPRn signals are digital signals enabled and configured by the Hibernation module. All other signals listed in this column are analog signals.

b. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin. Encodings 9, 10, and 12 are not used on this device.

## 10.2 Pad Capabilities

There are two main types of pads provided on the device:

- Fast GPIO pads: These pads provide variable, programmable drive strength and optimized voltage output levels.
- Slow GPIO pads: These pads provide 2-mA drive strength and are designed to be sensitive to voltage inputs. The following GPIOs port pins are designed with Slow GPIO Pads:
  – PJ1

Please refer to“Recommended GPIO Operating Characteristics” on page 2097 for details on the GPIO operating conditions for these two different pad types.

**Note:** Port pins PL6 and PL7 operate as Fast GPIO pads, but have 4-mA drive capability only. GPIO register controls for drive strength, slew rate and open drain have no effect on these pins. The registers which have no effect are as follows: **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR12R**, **GPIOSLR**, and **GPIOODR**.

**Note:** Port pins PM[7:4] operate as Fast GPIO pads but support only 2-, 4-, 6-, and 8-mA drive capability. 10- and 12-mA drive are not supported. All standard GPIO register controls, except for the **GPIODR12R** register, apply to these port pins.

## 10.3 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 10-1 on page 779 and Figure 10-2 on page 780). The TM4C129XKCZAD microcontroller contains 18 ports and thus of these physical GPIO blocks. Note that not all pins are implemented on every block. Some GPIO pins can function as I/O signals for the on-chip peripheral modules. For information on which GPIO pins are used for alternate hardware functions, refer to Table 31-5 on page 2081.

**Figure 10-1. Digital I/O Pads**

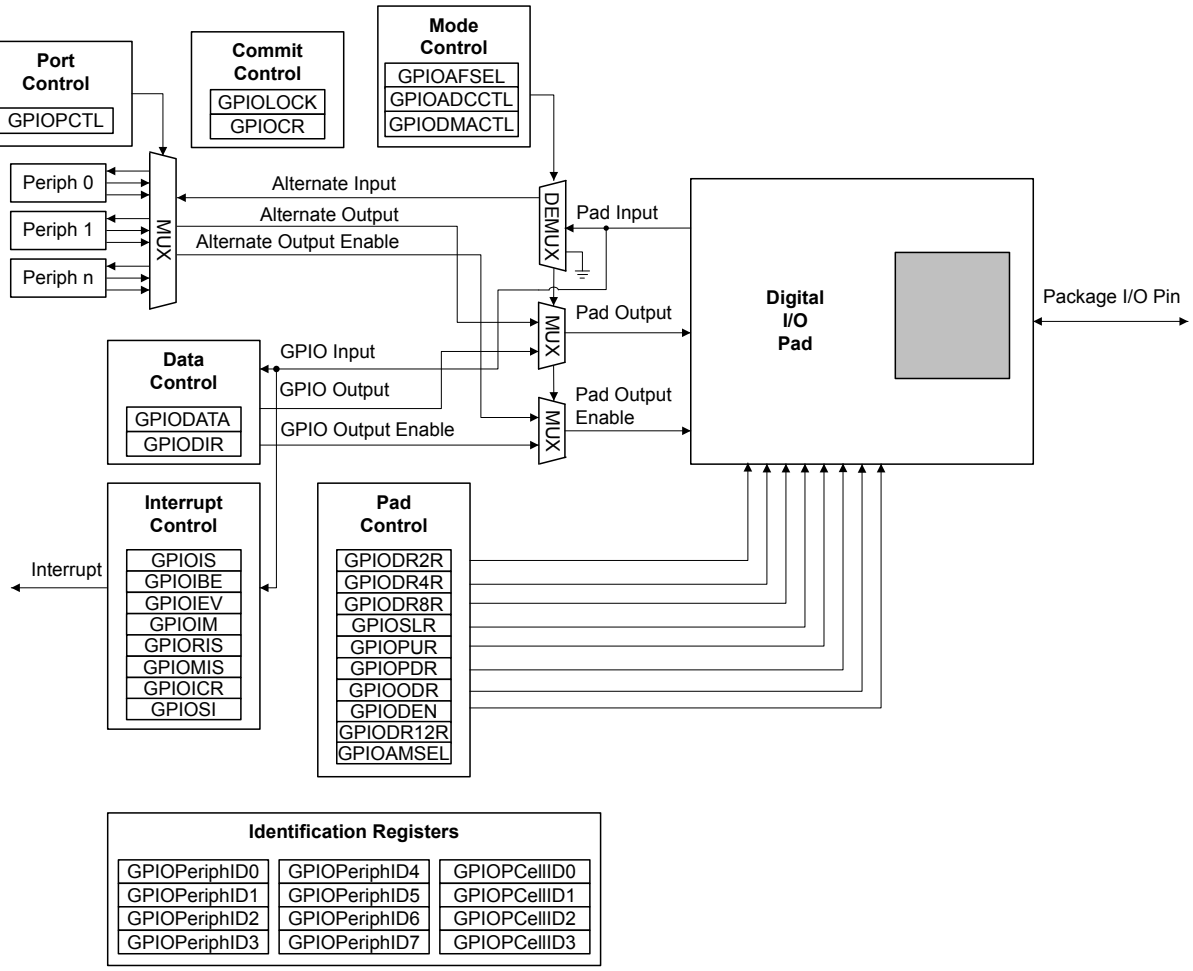
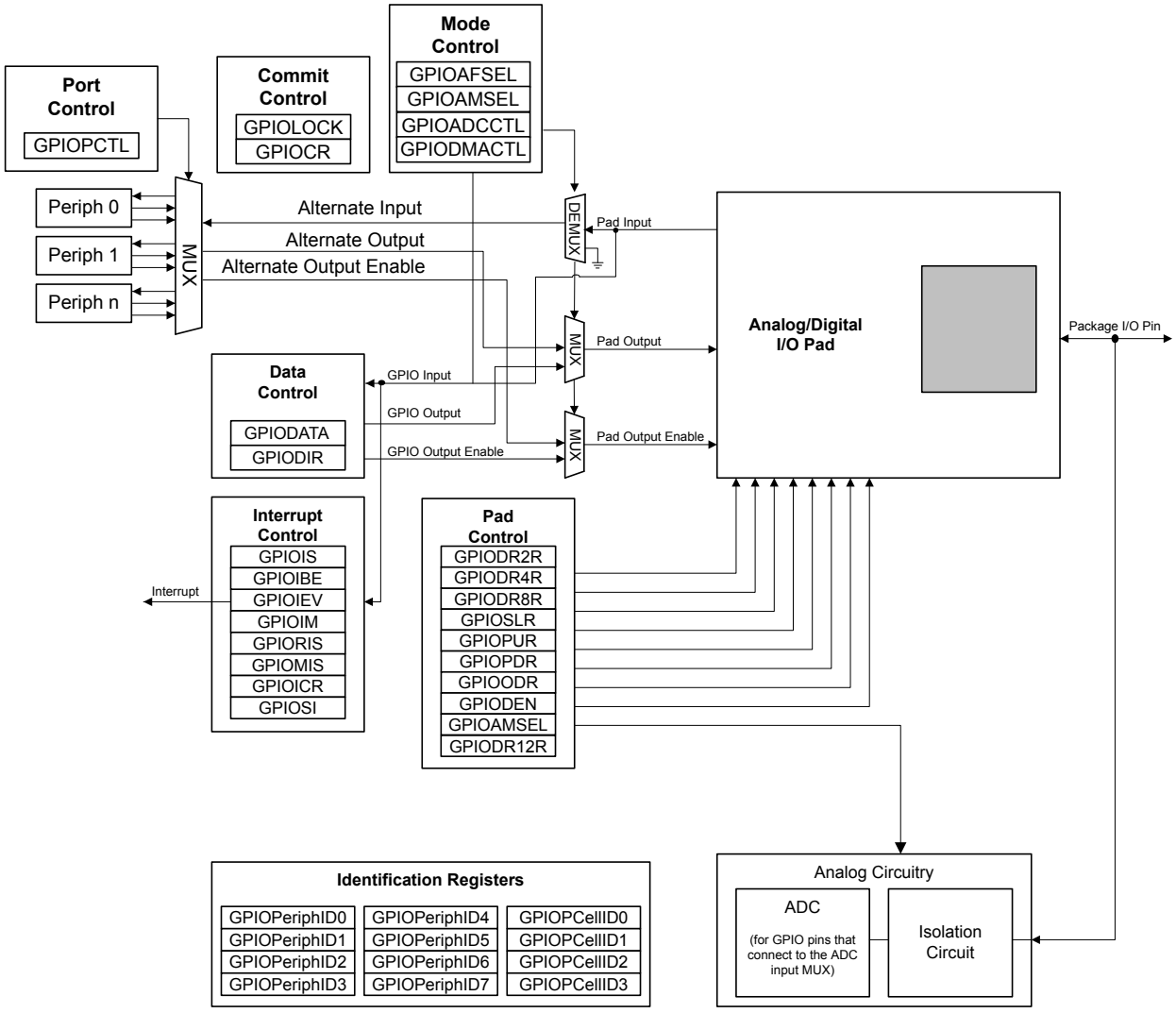


**Figure 10-2. Analog/Digital I/O Pads**




## 10.3.1 Data Control

The data control registers allow software to configure the operational modes of the GPIOs. The data direction register configures the GPIO as an input or an output while the data register either captures incoming data or drives it out to the pads.

**Caution – It is possible to create a software sequence that prevents the debugger from connecting to the TM4C129XKCZAD microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger. In the case that the software routine is not implemented and the device is locked out of the part, this issue can be solved by using the TM4C129XKCZAD Flash Programmer "Unlock" feature. Please refer to LMFLASHPROGRAMMER on the TI web for more information.**

#### 10.3.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 792) is used to configure each individual pin as an input or output. When the data direction bit is cleared, the GPIO is configured as an input, and the corresponding data register bit captures and stores the value on the GPIO port. When the data direction bit is set, the GPIO is configured as an output, and the corresponding data register bit is driven out on the GPIO port.

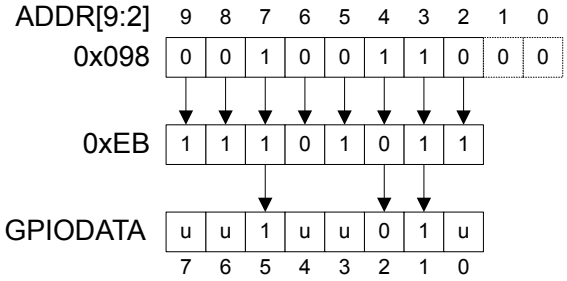
#### 10.3.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 791) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the **GPIODATA** register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set, the value of the **GPIODATA** register is altered. If the address bit is cleared, the data bit is left unchanged.

For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 10-3, where u indicates that data is unchanged by the write. This example demonstrates how **GPIODATA** bits 5, 2, and 1 are written.

**Figure 10-3. GPIODATA Write Example**




During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 10-4. This example shows how to read **GPIODATA** bits 5, 4, and 0.

**Figure 10-4. GPIODATA Read Example**



### 10.3.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. These registers are used to select the source of the interrupt, its polarity, and the edge properties. When one or more GPIO inputs cause an interrupt, a single interrupt output is sent to the interrupt controller for the entire GPIO port. For edge-triggered interrupts, software must clear the interrupt to enable any further interrupts. For a level-sensitive interrupt, the external source must hold the level constant for the interrupt to be recognized by the controller.

Three registers define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 793)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 794)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 796)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 797).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the **GPIO Raw Interrupt Status (GPIORIS)** and **GPIO Masked Interrupt Status (GPIOMIS)** registers (see page 798 and page 800). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the interrupt controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the interrupt controller.

For a GPIO level-detect interrupt, the interrupt signal generating the interrupt must be held until serviced. Once the input signal deasserts from the interrupt generating logical sense, the corresponding `RIS` bit in the **GPIORIS** register clears. For a GPIO edge-detect interrupt, the `RIS` bit in the **GPIORIS** register is cleared by writing a ‘1’ to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register (see page 802). The corresponding **GPIOMIS** bit reflects the masked value of the `RIS` bit.

When programming the interrupt control registers (**GPIOIS**, **GPIOIBE**, or **GPIOIEV**), the interrupts should be masked (**GPIOIM** cleared). Writing any value to an interrupt control register can generate a spurious interrupt if the corresponding bits are enabled.

#### 10.3.2.1 Interrupts Per Pin

Each pin of GPIO Port P and Port Q can trigger an interrupt. Each pin has a dedicated interrupt vector and can be handled by a separate interrupt handler. The `PP0` and `PQ0` interrupts serve as a master interrupt and provide a legacy aggregated interrupt version. For interrupt assignments, see Table 2-9 on page 126.

**Note:** The OR'ed summary interrupt occurs on bit 0 of the **GPIORIS** register. For summary interrupt mode, software should set the **GPIOIM** register to 0xFF and mask the port pin interrupts 1 through 7 in the **Interrupt Clear Enable (DISn)** register (see “NVIC Register Descriptions” on page 164). When servicing this interrupt, write a 1 to the corresponding bit in the **UNPENDn** register to clear the pending interrupt in the NVIC and clear the **GPIORIS** register pin interrupt bits by setting the `IC` field of the **GPIOICR** register to 0xFF.

#### 10.3.2.2 ADC Trigger Source

Any GPIO pin can be configured to be an external trigger for the ADC using the **GPIO ADC Control (GPIOADCCTL)** register. If any GPIO is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), and an interrupt for that port is generated, a trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger,

an ADC conversion is initiated. See page 1242. Note that whether the GPIO is configured to trigger on edge events or level events, a single-clock ADC trigger pulse is created in either event. Thus, when a level event is selected, the ADC sample sequence will run only one time and multiple sample sequences will not be executed if the level remains the same. It is recommended that edge events be used as ADC trigger source.

Note that if the Port B **GPIOADCCTL** register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

#### 10.3.2.3 µDMA Trigger Source

Any GPIO pin can be configured to be an external trigger for the µDMA using the **GPIO DMA Control (GPIODMACTL)** register. If any GPIO is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), a `dma_req` signal is sent to the µDMA. If the µDMA is configured to start a transfer based on the GPIO signal, a transfer is initiated. When transfer is complete, the `dma_done` signal is sent from the µDMA to the GPIO and is reported as a DMA (done) interrupt in the **GPIORIS** register.

#### 10.3.2.4 HIB Wake Source

GPIO pins K[7:4] on Port K can be configured as an external wake source for the hibernation (HIB) module. The pins can be configured in the following way:

1. Write 0x0000.0040 to the **HIBCTL** register at offset 0x010 to enable 32.768-kHz Hibernation oscillator.
2. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x06F.
3. Configure the **GPIOWAKEPEN** and **GPIOWAKELVL** registers at offsets 0x540 and 0x544 in the GPIO module. Enable the I/O wake pad configuration by writing 0x0000.0001 to the **HIBIO** register at offset 0x010.
4. When the `IOWRC` bit in the **HIBIO** register is read as 1, write 0x0000.0000 to the HIBO register to lock the current pad configuration so that any other writes to the **GPIOWAKEPEN** and **GPIOWAKELVL** register will be ignored.
5. The hibernation sequence may be initiated by writing 0x0000.0052 to the **HIBCTL** register.

The **GPIOWAKESTAT** register at offset 0x548 can be read to determine which port caused a wake pin assertion.

### 10.3.3 Mode Control

The GPIO pins can be controlled by either software or hardware. Software control is the default for most signals and corresponds to the GPIO mode, where the **GPIODATA** register is used to read or write the corresponding pins. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the **GPIO Port Control (GPIOPCTL)** register which selects one of several peripheral functions for each GPIO. For information on the configuration options, refer to Table 31-5 on page 2081.

**Note:** If any pin is to be used as an ADC input, the appropriate bit in the **GPIOAMSEL** register must be set to disable the analog isolation circuit.

### 10.3.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 2028 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803), **GPIO Pull Up Select (GPIOPUR)** register (see page 809), **GPIO Pull-Down Select (GPIOPDR)** register (see page 811), and **GPIO Digital Enable (GPIODEN)** register (see page 814) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 816) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 817) have been set.

### 10.3.5 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**,**GPIODR12R**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable for each GPIO. If 3.3V is applied to a GPIO configured as an open-drain output, the output voltage will depend on the strength of your pull-up resistor. The GPIO pad is not electrically configured to output 3.3 V.

**Note:** Port pins PL6 and PL7 operate as Fast GPIO pads, but have 4-mA drive capability only. GPIO register controls for drive strength, slew rate and open drain have no effect on these pins. The registers which have no effect are as follows: **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR12R**, **GPIOSLR**, and **GPIOODR**.

**Note:** Port pins PM[7:4] operate as Fast GPIO pads but support only 2-, 4-, 6-, and 8-mA drive capability. 10- and 12-mA drive are not supported. All standard GPIO register controls, except for the **GPIODR12R** register, apply to these port pins.

#### 10.3.5.1 Extended Drive Enable

The **GPIO Peripheral Configuration (GPIOPC)** register controls the extended drive modes of the GPIO. When the EDE bit in **GPIO Peripheral Properties (GPIOPP)** register is set and the EDMn bit field for a GPIO pin is non-zero in the **GPIOPC** register, the **GPIODRnR** registers do not drive their default value, but instead output an incremental drive strength, which has an additive effect. This allows for more drive strength possibilities. When the EDE bit is set and the EDMn bit field is non-zero, the 2 mA driver is always enabled. Any bits enabled in the **GPIODR4R** register for a pin with a non-zero EDMn value, add an additional 2 mA. Any bits set in the **GPIODR8R** add an extra 4 mA of drive. The **GPIODR12R** register is only valid when the EDMn value is 0x3. For this encoding, setting a bit in the **GPIODR12R** register adds 4 mA of drive to the already existing 8 mA, for a 12 mA drive strength. To attain a 10-mA drive strength, the pin's **GPIODR12R** and **GPIODR8R** register should be enabled; this would result in the addition of two, 4-mA current drivers to the already enabled 2-mA driver. The table below shows the drive capability options. If EDMn is 0x00, then the **GPIODR2R**, **GPIODR4R**, and **GPIODR8R** function as stated in their default register description.

**Note:** A **GPIOPC** register write must precede the configuration of the **GPIODRnR** registers in order for extended drive mode to take effect.

**Table 10-3. GPIO Drive Strength Options**

| EDE (GPIOPP) | EDMn (GPIOPC) | GPIODR12R (+4mA) | GPIODR8R (+4mA) | GPIODR4R (+2mA) | GPIODR2R (2mA) | Drive (mA) |
|---|---|---|---|---|---|---|
| X | 0x0 | N/A | 0 | 0 | 1 | 2 |
| | | | 0 | 1 | 0 | 4 |
| | | | 1 | 0 | 0 | 8 |
| 1 | 0x1 | N/A | 0 | 0 | N/A | 2 |
| | | | 0 | 1 | N/A | 4 |
| | | | 1 | 0 | N/A | 6 |
| | | | 1 | 1 | N/A | 8 |
| 1 | 0x3 | 0 | 0 | 0 | N/A | 2 |
| | | 0 | 0 | 1 | N/A | 4 |
| | | 0 | 1 | 0 | N/A | 6 |
| | | 0 | 1 | 1 | N/A | 8 |
| | | 1 | 1 | 0 | N/A | 10 |
| | | 1 | 1 | 1 | N/A | 12 |
| | | 1 | 0 | N/A | N/A | N/A |
| 1 | 0x2 | N/A | N/A | N/A | N/A | N/A |

### 10.3.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the **GPIOPeriphID0**-**GPIOPeriphID7** registers as well as the **GPIOPCellID0**-**GPIOPCellID3** registers.

## 10.4 Initialization and Configuration

To configure the GPIO pins of a particular port, follow these steps:

1. Enable the clock to the port by setting the appropriate bits in the **RCGCGPIO** register (see page 399). In addition, the **SCGCGPIO** and **DCGCGPIO** registers can be programmed in the same manner to enable clocking in Sleep and Deep-Sleep modes.

2. Set the direction of the GPIO port pins by programming the **GPIODIR** register. A write of a 1 indicates output and a write of a 0 indicates input.

3. Configure the **GPIOAFSEL** register to program each bit as a GPIO or alternate pin. If an alternate pin is chosen for a bit, then the PMCx field must be programmed in the **GPIOPCTL** register for the specific peripheral required. There are also two registers, **GPIOADCCTL** and **GPIODMACTL**, which can be used to program a GPIO pin as a ADC or µDMA trigger, respectively.

4. Set the EDMn field in the **GPIOPC** register as shown in Table 10-3 on page 785.

5. Set or clear the **GPIODR4R** register bits as shown in Table 10-3 on page 785.

6. Set or clear the **GPIODR8R** register bits as shown in Table 10-3 on page 785.

7. Set or clear the **GPIODR12R** register bits as shown in Table 10-3 on page 785.

8. Program each pad in the port to have either pull-up, pull-down, or open drain functionality through the **GPIOPUR**, **GPIOPDR**, **GPIOODR** register. Slew rate may also be programmed, if needed, through the **GPIOSLR** register.

9. To enable GPIO pins as digital I/Os, set the appropriate DEN bit in the **GPIODEN** register. To enable GPIO pins to their analog function (if available), set the GPIOAMSEL bit in the **GPIOAMSEL** register.

10. Program the **GPIOIS**, **GPIOIBE**, **GPIOEV**, and **GPIOIM** registers to configure the type, event, and mask of the interrupts for each port.

    **Note:** To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

    a. Mask the corresponding port by clearing the IME field in the **GPIOIM** register.

    b. Configure the IS field in the **GPIOIS** register and the IBE field in the **GPIOIBE** register.

    c. Clear the **GPIORIS** register.

    d. Unmask the port by setting the IME field in the **GPIOIM** register.

11. Optionally, software can lock the configurations of the NMI and JTAG/SWD pins on the GPIO port pins, by setting the LOCK bits in the **GPIOLOCK** register.

When the internal POR signal is asserted and until otherwise configured, all GPIO pins are configured to be undriven (tristate): **GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0Table 10-4 on page 786 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 10-5 on page 787 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

**Table 10-4. GPIO Pad Configuration Examples**

| Configuration | GPIO Register Bit Value[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | AFSEL | DIR | ODR | DEN | PUR | PDR | DR2R | DR4R | DR8R | DR12R | SLR |
| Digital Input (GPIO) | 0 | 0 | 0 | 1 | ? | ? | X | X | X | X | X |
| Digital Output (GPIO) | 0 | 1 | 0 | 1 | ? | ? | ? | ? | ? | ? | ? |
| Open Drain Output (GPIO) | 0 | 1 | 1 | 1 | X | X | ? | ? | ? | ? | ? |
| Open Drain Input/Output (I2CSDA) | 1 | X | 1 | 1 | X | X | ? | ? | ? | ? | ? |
| Digital Input/Output (I2CSCL) | 1 | X | 0 | 1 | X | X | ? | ? | ? | ? | ? |
| Digital Input (Timer CCP) | 1 | X | 0 | 1 | ? | ? | X | X | X | X | X |
| Digital Input (QEI) | 1 | X | 0 | 1 | ? | ? | X | X | X | X | X |
| Digital Output (PWM) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? | ? |
| Digital Output (Timer PWM) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? | ? |

**Table 10-4. GPIO Pad Configuration Examples *(continued)***

| Configuration | GPIO Register Bit Value[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | AFSEL | DIR | ODR | DEN | PUR | PDR | DR2R | DR4R | DR8R | DR12R | SLR |
| Digital Input/Output (SSI) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? | ? |
| Digital Input/Output (UART) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? | ? |
| Analog Input (Comparator) | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | X |
| Digital Output (Comparator) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? | ? |

a. X=Ignored (don't care bit)

?=Can be either 0 or 1, depending on the configuration

**Table 10-5. GPIO Interrupt Configuration Example**

| Register | Desired Interrupt Event Trigger | Pin 2 Bit Value[a] | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIOIS | 0=edge<br>1=level | X | X | X | X | X | 0 | X | X |
| GPIOIBE | 0=single edge<br>1=both edges | X | X | X | X | X | 0 | X | X |
| GPIOIEV | 0=Low level, or falling edge<br>1=High level, or rising edge | X | X | X | X | X | 1 | X | X |
| GPIOIM | 0=masked<br>1=not masked | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

a. X=Ignored (don't care bit)

## 10.5 Register Map

Table 10-7 on page 789 lists the GPIO registers.

**Important:** The GPIO registers in this chapter are duplicated in each GPIO block; however, depending on the block, all eight bits may not be connected to a GPIO pad. In those cases, writing to unconnected bits has no effect, and reading unconnected bits returns no meaningful data. See "Signal Description" on page 773 for the GPIOs included on this device.

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (AHB): 0x4005.D000
- GPIO Port G (AHB): 0x4005.E000
- GPIO Port H (AHB): 0x4005.F000

- GPIO Port J (AHB): 0x4006.0000
- GPIO Port K (AHB): 0x4006.1000
- GPIO Port L (AHB): 0x4006.2000
- GPIO Port M (AHB): 0x4006.3000
- GPIO Port N (AHB): 0x4006.4000
- GPIO Port P (AHB): 0x4006.5000
- GPIO Port Q (AHB): 0x4006.6000
- GPIO Port R (AHB): 0x4006.7000
- GPIO Port S (AHB): 0x4006.8000
- GPIO Port T (AHB): 0x4006.9000

Note that each GPIO module clock must be enabled before the registers can be programmed (see page 399). There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

**Important:** The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 10-6. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see "Commit Control" on page 784.

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the NMI pin and the four JTAG/SWD pins (see "Signal Tables" on page 2028 for pin numbers). These six pins are the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for the corresponding GPIO Ports is RW.

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the NMI and JTAG/SWD pins (see "Signal Tables" on page 2028 for pin numbers). To ensure that the JTAG and NMI pins are not accidentally programmed as GPIO pins, these pins default to non-committable. Because of this, the default reset value of **GPIOCR** changes for the corresponding ports.

**Table 10-7. GPIO Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | GPIODATA | RW | 0x0000.0000 | GPIO Data | 791 |
| 0x400 | GPIODIR | RW | 0x0000.0000 | GPIO Direction | 792 |
| 0x404 | GPIOIS | RW | 0x0000.0000 | GPIO Interrupt Sense | 793 |
| 0x408 | GPIOIBE | RW | 0x0000.0000 | GPIO Interrupt Both Edges | 794 |
| 0x40C | GPIOIEV | RW | 0x0000.0000 | GPIO Interrupt Event | 796 |
| 0x410 | GPIOIM | RW | 0x0000.0000 | GPIO Interrupt Mask | 797 |
| 0x414 | GPIORIS | RO | 0x0000.0000 | GPIO Raw Interrupt Status | 798 |
| 0x418 | GPIOMIS | RO | 0x0000.0000 | GPIO Masked Interrupt Status | 800 |
| 0x41C | GPIOICR | W1C | 0x0000.0000 | GPIO Interrupt Clear | 802 |
| 0x420 | GPIOAFSEL | RW | - | GPIO Alternate Function Select | 803 |
| 0x500 | GPIODR2R | RW | 0x0000.00FF | GPIO 2-mA Drive Select | 805 |
| 0x504 | GPIODR4R | RW | 0x0000.0000 | GPIO 4-mA Drive Select | 806 |
| 0x508 | GPIODR8R | RW | 0x0000.0000 | GPIO 8-mA Drive Select | 807 |
| 0x50C | GPIOODR | RW | 0x0000.0000 | GPIO Open Drain Select | 808 |
| 0x510 | GPIOPUR | RW | - | GPIO Pull-Up Select | 809 |
| 0x514 | GPIOPDR | RW | 0x0000.0000 | GPIO Pull-Down Select | 811 |
| 0x518 | GPIOSLR | RW | 0x0000.0000 | GPIO Slew Rate Control Select | 813 |
| 0x51C | GPIODEN | RW | - | GPIO Digital Enable | 814 |
| 0x520 | GPIOLOCK | RW | 0x0000.0001 | GPIO Lock | 816 |
| 0x524 | GPIOCR | - | - | GPIO Commit | 817 |
| 0x528 | GPIOAMSEL | RW | 0x0000.0000 | GPIO Analog Mode Select | 819 |
| 0x52C | GPIOPCTL | RW | - | GPIO Port Control | 820 |
| 0x530 | GPIOADCCTL | RW | 0x0000.0000 | GPIO ADC Control | 822 |
| 0x534 | GPIODMACTL | RW | 0x0000.0000 | GPIO DMA Control | 823 |
| 0x538 | GPIOSI | RW | 0x0000.0000 | GPIO Select Interrupt | 824 |
| 0x53C | GPIODR12R | RW | 0x0000.0000 | GPIO 12-mA Drive Select | 825 |
| 0x540 | GPIOWAKEPEN | RW | 0x0000.0000 | GPIO Wake Pin Enable | 826 |
| 0x544 | GPIOWAKELVL | RW | 0x0000.0000 | GPIO Wake Level | 828 |
| 0x548 | GPIOWAKESTAT | RO | 0x0000.0000 | GPIO Wake Status | 830 |
| 0xFC0 | GPIOPP | RO | 0x0000.0001 | GPIO Peripheral Property | 832 |
| 0xFC4 | GPIOPC | RW | 0x0000.0000 | GPIO Peripheral Configuration | 833 |
| 0xFD0 | GPIOPeriphID4 | RO | 0x0000.0000 | GPIO Peripheral Identification 4 | 836 |

**Table 10-7. GPIO Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xFD4 | GPIOPeriphID5 | RO | 0x0000.0000 | GPIO Peripheral Identification 5 | 837 |
| 0xFD8 | GPIOPeriphID6 | RO | 0x0000.0000 | GPIO Peripheral Identification 6 | 838 |
| 0xFDC | GPIOPeriphID7 | RO | 0x0000.0000 | GPIO Peripheral Identification 7 | 839 |
| 0xFE0 | GPIOPeriphID0 | RO | 0x0000.0061 | GPIO Peripheral Identification 0 | 840 |
| 0xFE4 | GPIOPeriphID1 | RO | 0x0000.0000 | GPIO Peripheral Identification 1 | 841 |
| 0xFE8 | GPIOPeriphID2 | RO | 0x0000.0018 | GPIO Peripheral Identification 2 | 842 |
| 0xFEC | GPIOPeriphID3 | RO | 0x0000.0001 | GPIO Peripheral Identification 3 | 843 |
| 0xFF0 | GPIOPCellID0 | RO | 0x0000.000D | GPIO PrimeCell Identification 0 | 844 |
| 0xFF4 | GPIOPCellID1 | RO | 0x0000.00F0 | GPIO PrimeCell Identification 1 | 845 |
| 0xFF8 | GPIOPCellID2 | RO | 0x0000.0005 | GPIO PrimeCell Identification 2 | 846 |
| 0xFFC | GPIOPCellID3 | RO | 0x0000.00B1 | GPIO PrimeCell Identification 3 | 847 |

## 10.6 Register Descriptions

The remainder of this section lists and describes the GPIO registers, in numerical order by address offset.

## Register 1: GPIO Data (GPIODATA), offset 0x000

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 792).

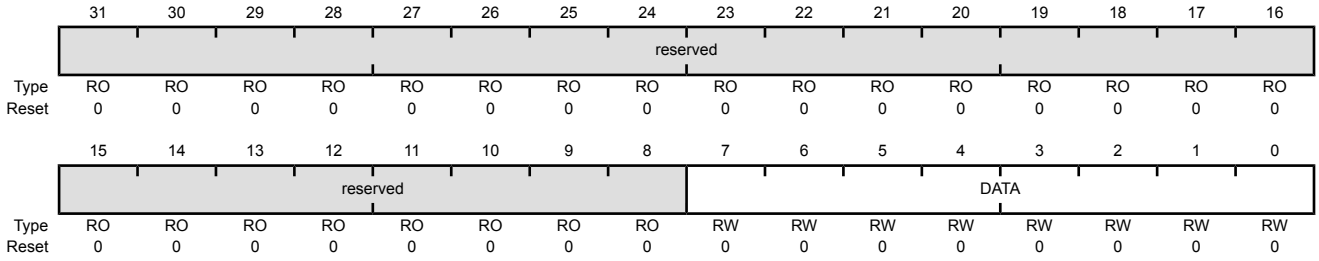
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set. Otherwise, the bit values remain unchanged by the write.

Similarly, the values read from this register are determined for each bit by the mask bit derived from the address used to access the data register, bits [9:2]. Bits that are set in the address mask cause the corresponding bits in **GPIODATA** to be read, and bits that are clear in the address mask cause the corresponding bits in **GPIODATA** to be read as 0, regardless of their value.

A read from **GPIODATA** returns the last bit value written if the respective pins are configured as outputs, or it returns the value on the corresponding input pin when these are configured as inputs. All bits are cleared by a reset.

GPIO Data (GPIODATA)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x000
Type RW, reset 0x0000.0000



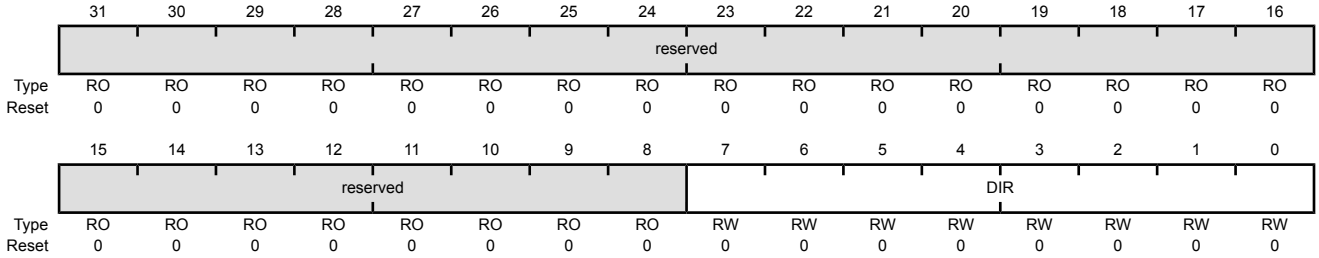

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | RW | 0x00 | GPIO Data<br>This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See “Data Register Operation” on page 781 for examples of reads and writes. |

## Register 2: GPIO Direction (GPIODIR), offset 0x400

The **GPIODIR** register is the data direction register. Setting a bit in the **GPIODIR** register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIO Direction (GPIODIR)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x400
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | DIR | RW | 0x00 | GPIO Data Direction<br>Value Description<br>0 Corresponding pin is an input.<br>1 Corresponding pins is an output. |

## Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404

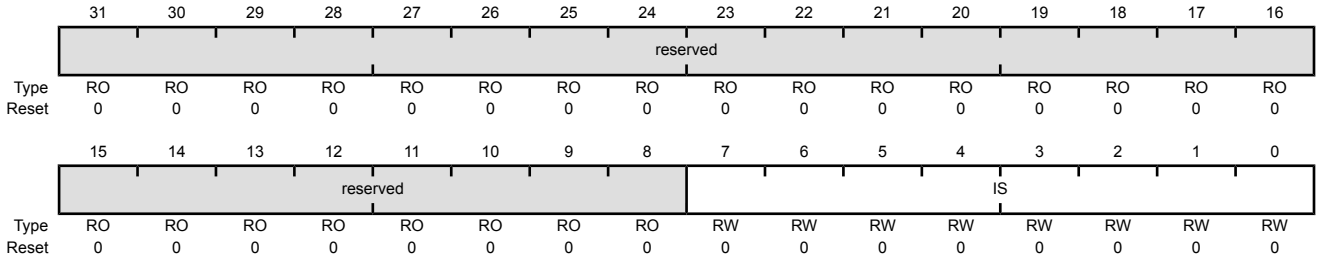
The **GPIOIS** register is the interrupt sense register. Setting a bit in the **GPIOIS** register configures the corresponding pin to detect levels, while clearing a bit configures the corresponding pin to detect edges. All bits are cleared by a reset.

**Note:** To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

1. Mask the corresponding port by clearing the `IME` field in the **GPIOIM** register.
2. Configure the `IS` field in the **GPIOIS** register and the `IBE` field in the **GPIOIBE** register.
3. Clear the **GPIORIS** register.
4. Unmask the port by setting the `IME` field in the **GPIOIM** register.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x404
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | IS | RW | 0x00 | GPIO Interrupt Sense |

| Value | Description |
|---|---|
| 0 | The edge on the corresponding pin is detected (edge-sensitive). |
| 1 | The level on the corresponding pin is detected (level-sensitive). |

## Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408

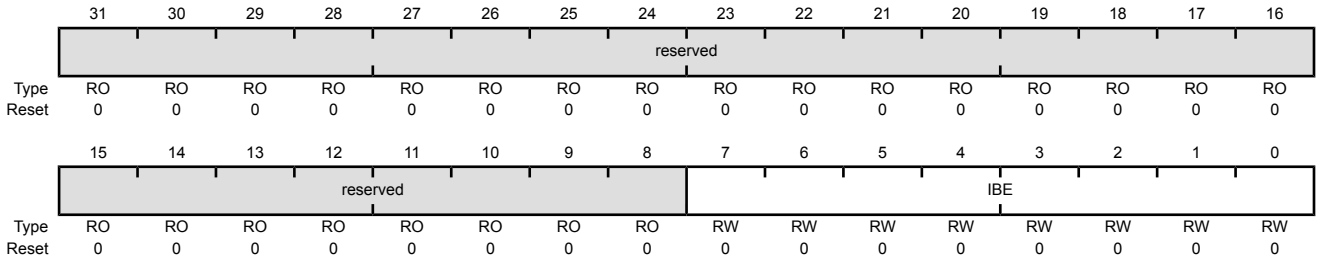
The **GPIOIBE** register allows both edges to cause interrupts. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 793) is set to detect edges, setting a bit in the **GPIOIBE** register configures the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 796). Clearing a bit configures the pin to be controlled by the **GPIOIEV** register. All bits are cleared by a reset.

**Note:** To prevent false interrupts, the following steps should be taken when re-configuring GPIO edge and interrupt sense registers:

1. Mask the corresponding port by clearing the `IME` field in the **GPIOIM** register.
2. Configure the `IS` field in the **GPIOIS** register and the `IBE` field in the **GPIOIBE** register.
3. Clear the **GPIORIS** register.
4. Unmask the port by setting the `IME` field in the **GPIOIM** register.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x408
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | IBE | RW | 0x00 | GPIO Interrupt Both Edges |

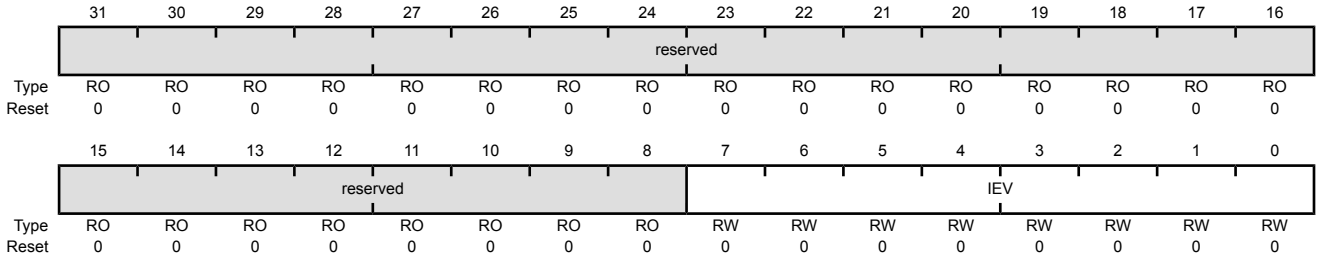
| Value | Description |
|---|---|
| 0 | Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 796). |
| 1 | Both edges on the corresponding pin trigger an interrupt. |

## Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C

The **GPIOIEV** register is the interrupt event register. Setting a bit in the **GPIOIEV** register configures the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 793). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in the **GPIOIS** register. All bits are cleared by a reset.

GPIO Interrupt Event (GPIOIEV)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x40C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | IEV | RW | 0x00 | GPIO Interrupt Event |

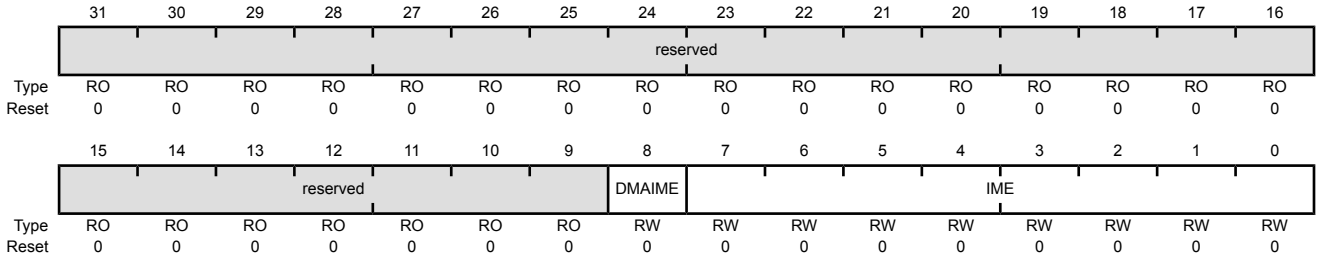
| Value | Description |
|---|---|
| 0 | A falling edge or a Low level on the corresponding pin triggers an interrupt. |
| 1 | A rising edge or a High level on the corresponding pin triggers an interrupt. |

## Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410

The **GPIOIM** register is the interrupt mask register. Setting a bit in the **GPIOIM** register allows interrupts that are generated by the corresponding pin to be sent to the interrupt controller on the combined interrupt signal. Clearing a bit prevents an interrupt on the corresponding pin from being sent to the interrupt controller. All bits are cleared by a reset.

GPIO Interrupt Mask (GPIOIM)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x410
Type RW, reset 0x0000.0000



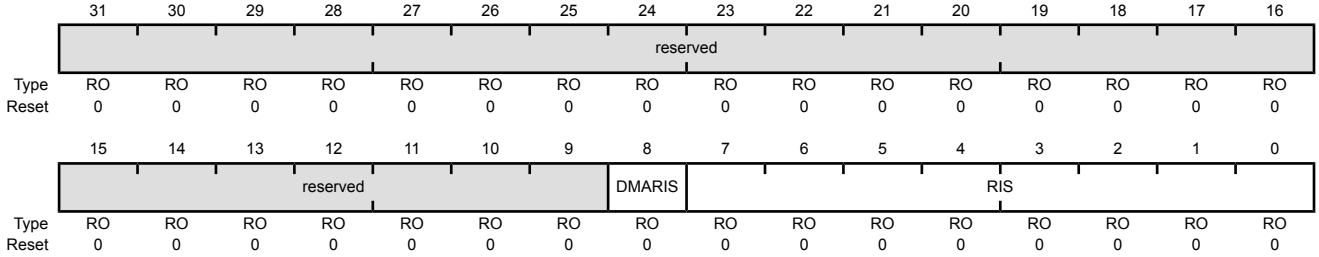

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | DMAIME | RW | 0 | GPIO uDMA Done Interrupt Mask Enable<br>Value Description<br>0 The µDMA done interrupt is masked and does not cause an interrupt.<br>1 The µDMA done interrupt is not masked and can generate an interrupt to the interrupt controller. |
| 7:0 | IME | RW | 0x00 | GPIO Interrupt Mask Enable<br>Value Description<br>0 The interrupt from the corresponding pin is masked.<br>1 The interrupt from the corresponding pin is sent to the interrupt controller. |

## Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414

The **GPIORIS** register is the raw interrupt status register. A bit in this register is set when an interrupt condition occurs on the corresponding GPIO pin or if a µDMA done interrupt occurs. If the corresponding bit in the **GPIO Interrupt Mask (GPIOIM)** register (see page 797) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. For a GPIO level-detect interrupt, the interrupt signal generating the interrupt must be held until serviced. Once the input signal deasserts from the interrupt generating logical sense, the corresponding `RIS` bit in the **GPIORIS** register clears. For a GPIO edge-detect interrupt, the `RIS` bit in the **GPIORIS** register is cleared by writing a '1' to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register. The corresponding **GPIOMIS** bit reflects the masked value of the `RIS` bit.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x414
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | 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. |
| 8 | DMARIS | RO | 0 | GPIO µDMA Done Interrupt Raw Status<br><br>Value Description<br>0 A µDMA done interrupt has not occurred.<br>1 A µDMA done interrupt has occurred and an interrupt has been triggered and is pending.<br><br>This bit is cleared by writing a 1 to the `DMAIC` bit in the **GPIOICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | RIS | RO | 0x00 | GPIO Interrupt Raw Status |

| Value | Description |
|---|---|
| 0 | An interrupt condition has not occurred on the corresponding pin. |
| 1 | An interrupt condition has occurred on the corresponding pin. |

For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted.

## Register 8: GPIO Masked Interrupt Status (GPIOMIS), offset 0x418

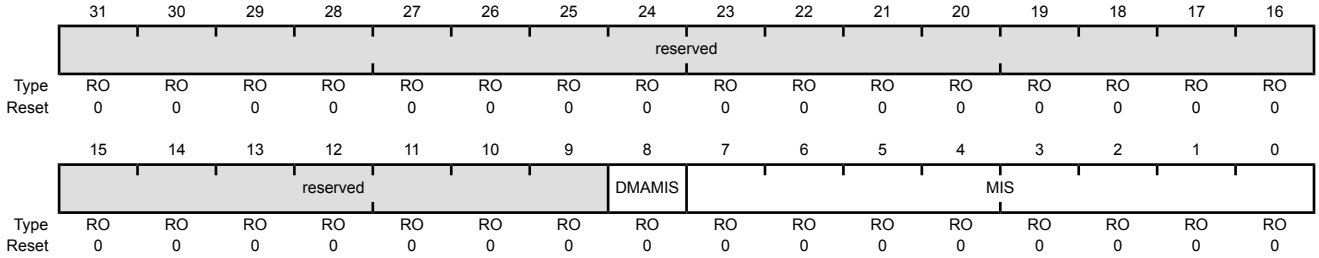
The **GPIOMIS** register is the masked interrupt status register. If a bit is set in this register, the corresponding interrupt has triggered an interrupt to the interrupt controller. If a bit is clear, either no interrupt has been generated, or the interrupt is masked.

Note that if the Port B **GPIOADCCTL** register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

**GPIOMIS** is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x418
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | 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. |
| 8 | DMAMIS | RO | 0 | GPIO µDMA Done Masked Interrupt Status<br><br>Value Description<br>0 The µDMA done interrupt is masked or has not occurred.<br>1 An unmasked µDMA done interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the `DMAIC` bit in the **GPIOICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | MIS | RO | 0x00 | GPIO Masked Interrupt Status |

| Value | Description |
|---|---|
| 0 | An interrupt condition on the corresponding pin is masked or has not occurred. |
| 1 | An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller. |

For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

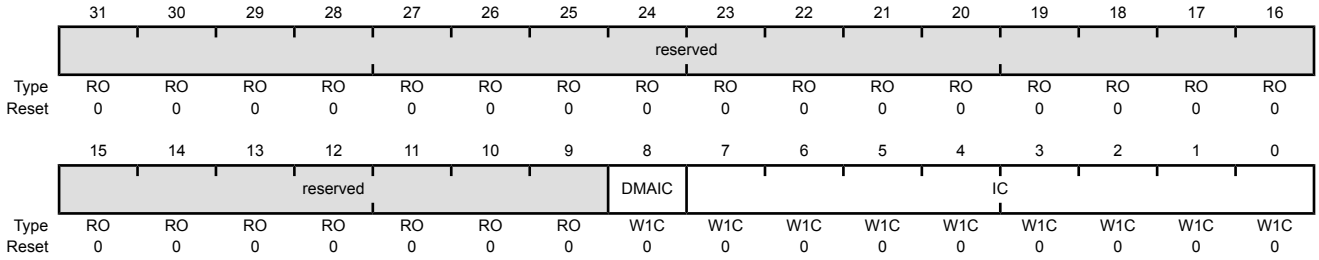
For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted.

## Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C

The **GPIOICR** register is the interrupt clear register. Writing a 1 to the DMAIC bit in this register clears the corresponding interrupt bit in the **GPIORIS** and **GPIOMIS** registers. For edge-detect interrupts, writing a 1 to the IC bit in the **GPIOICR** register clears the corresponding bit in the **GPIORIS** and **GPIOMIS** registers. If the interrupt is a level-detect, the IC bit in this register has no effect. In addition, writing a 0 to any of the bits in the **GPIOICR** register has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x41C
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | 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. |
| 8 | DMAIC | W1C | 0 | GPIO µDMA Interrupt Clear<br>Value Description<br>0 The µDMA done interrupt is unaffected.<br>1 The µDMA done interrupt is cleared. |
| 7:0 | IC | W1C | 0x00 | GPIO Interrupt Clear<br>Value Description<br>0 The corresponding interrupt is unaffected.<br>1 The corresponding interrupt is cleared. |

## Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420

**Note:** Tamper pins enabled in the **Hibernate Tamper IO Control and Status (HIBTPIO)** register override the AFSEL configuration.

The **GPIOAFSEL** register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The **GPIO Port Control (GPIOPCTL)** register is used to select one of the possible functions. Table 31-5 on page 2081 details which functions are muxed on each GPIO pin. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

**Important:** The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 10-8. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 784.

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

**Caution – It is possible to create a software sequence that prevents the debugger from connecting to the TM4C129XKCZAD microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger. In the case that the software routine is not implemented and the device is locked out of the part, this issue can be solved by using the TM4C129XKCZAD Flash Programmer "Unlock" feature. Please refer to LMFLASHPROGRAMMER on the TI web for more information.**

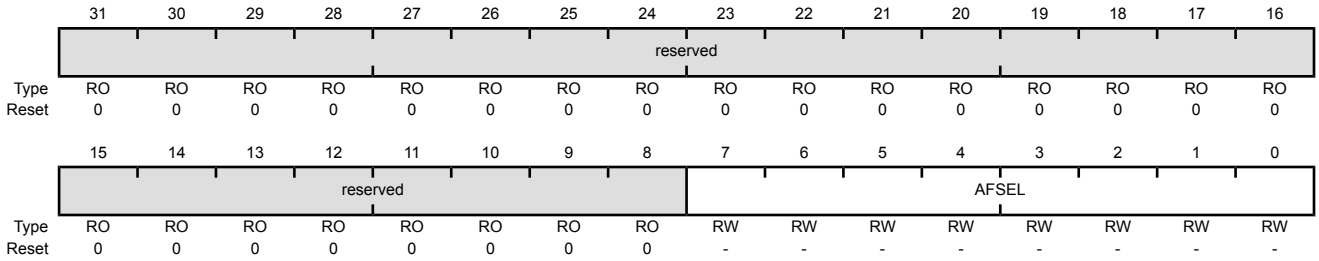
The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 2028 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803), **GPIO**

**Pull Up Select (GPIOPUR)** register (see page 809), **GPIO Pull-Down Select (GPIOPDR)** register (see page 811), and **GPIO Digital Enable (GPIODEN)** register (see page 814) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 816) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 817) have been set.

When using the $I^2C$ module, in addition to setting the **GPIOAFSEL** register bits for the $I^2C$ clock and data pins, the data pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register (see examples in "Initialization and Configuration" on page 785).

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x420
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | AFSEL | RW | - | GPIO Alternate Function Select |

| Value | Description |
|---|---|
| 0 | The associated pin functions as a GPIO and is controlled by the GPIO registers. |
| 1 | The associated pin functions as a peripheral signal and is controlled by the alternate hardware function. |

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 773.

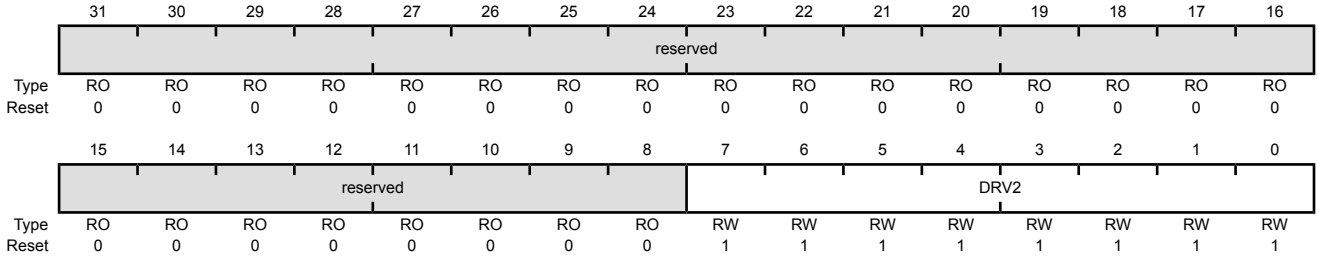
## Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500

The **GPIODR2R** register is the 2-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV2 bit for a GPIO signal, the corresponding DRV4 bit in the **GPIODR4R** register and DRV8 bit in the **GPIODR8R** register are automatically cleared by hardware. By default, all GPIO pins have 2-mA drive.

**Note:** This register has no effect on port pins PL6 and PL7.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x500
Type RW, reset 0x0000.00FF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | DRV2 | RW | 0xFF | Output Pad 2-mA Drive Enable |

| Value | Description |
|---|---|
| 0 | The drive for the corresponding GPIO pin is controlled by the **GPIODR4R** or **GPIODR8R** register. |
| 1 | The corresponding GPIO pin has 2-mA drive. |

Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the next clock cycle.

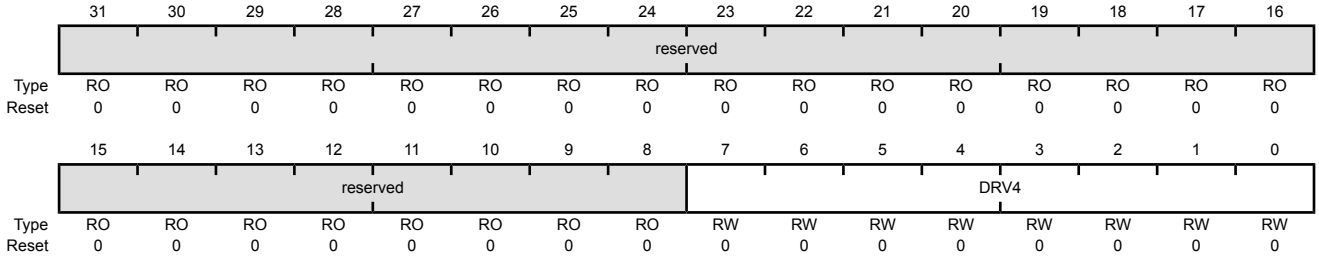
## Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504

The **GPIODR4R** register is the 4-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV4 bit for a GPIO signal, the corresponding DRV2 bit in the **GPIODR2R** register and DRV8 bit in the **GPIODR8R** register are automatically cleared by hardware.

**Note:** This register has no effect on port pins PL6 and PL7.

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x504
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | DRV4 | RW | 0x00 | Output Pad 4-mA Drive Enable |

| Value | Description |
|---|---|
| 0 | The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR8R** register. |
| 1 | The corresponding GPIO pin has 4-mA drive. |

Setting a bit in either the **GPIODR2** register or the **GPIODR8** register clears the corresponding 4-mA enable bit. The change is effective on the next clock cycle.

## Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

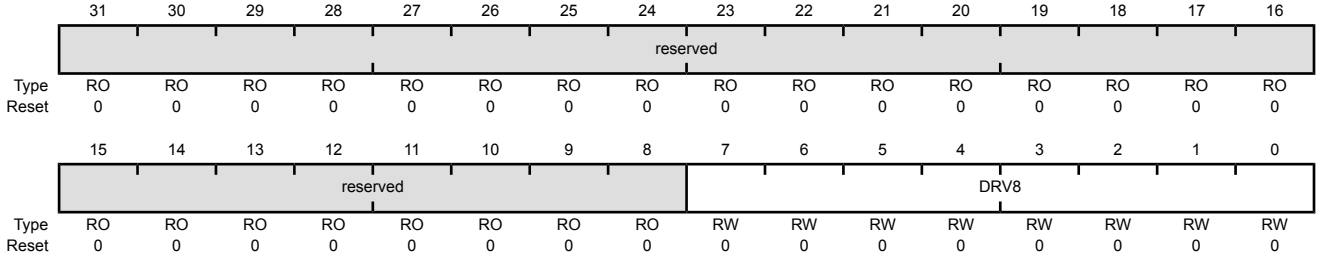
The **GPIODR8R** register is the 8-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the DRV8 bit for a GPIO signal, the corresponding DRV2 bit in the **GPIODR2R** register and DRV4 bit in the **GPIODR4R** register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

**Note:** There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the $V_{OH}/V_{OL}$ levels. See “Recommended Operating Conditions” on page 2097 for further information.

**Note:** This register has no effect on port pins PL6 and PL7.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x508
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | DRV8 | RW | 0x00 | Output Pad 8-mA Drive Enable<br><br>Value Description<br>0 The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR4R** register.<br>1 The corresponding GPIO pin has 8-mA drive.<br><br>Setting a bit in either the **GPIODR2** register or the **GPIODR4** register clears the corresponding 8-mA enable bit. The change is effective on the next clock cycle. |

## Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C

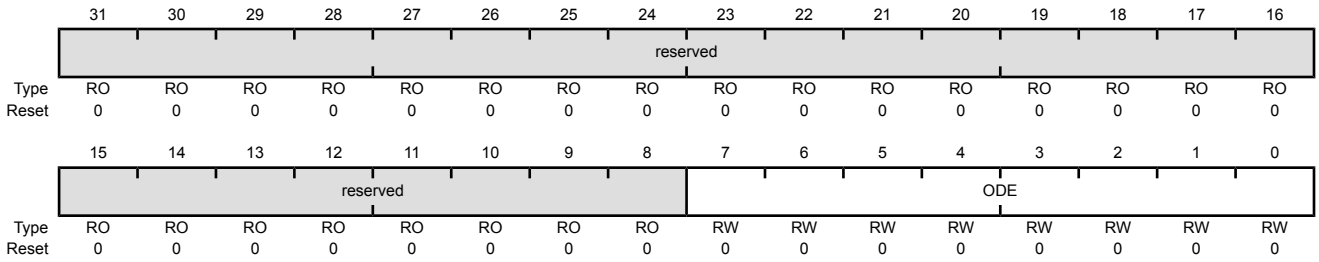
The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open-drain configuration of the corresponding GPIO pad. When open-drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Enable (GPIODEN)** register (see page 814). Corresponding bits in the drive strength and slew rate control registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired fall times. The GPIO acts as an input if the corresponding bit in the **GPIODIR** register is cleared. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

When using the $I^2C$ module, in addition to configuring the data pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the $I^2C$ clock and data pins should be set (see examples in “Initialization and Configuration” on page 785).

**Note:** This register has no effect on port pins PL6 and PL7.

GPIO Open Drain Select (GPIOODR)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x50C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | ODE | RW | 0x00 | Output Pad Open Drain Enable<br>Value Description<br>0 The corresponding pin is not configured as open drain.<br>1 The corresponding pin is configured as open drain. |

## Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510

The **GPIOPUR** register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 811). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are cleared prevent writes to the equivalent bit in this register.

**Important:** The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 10-9. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

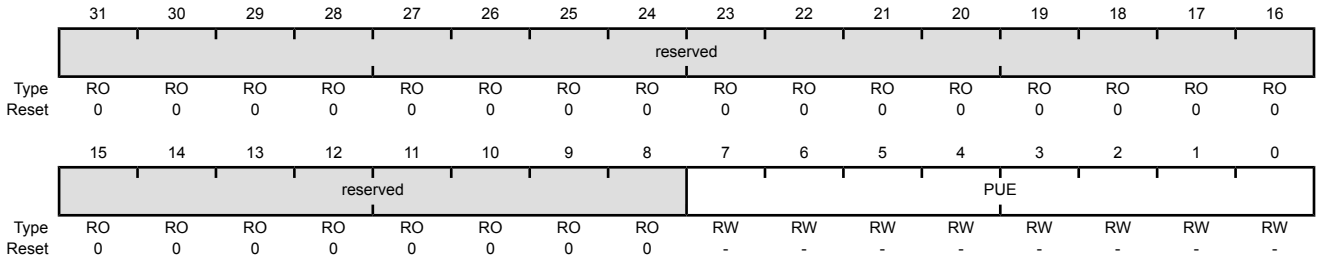
The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 784.

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 2028 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803), **GPIO Pull Up Select (GPIOPUR)** register (see page 809), **GPIO Pull-Down Select (GPIOPDR)** register (see page 811), and **GPIO Digital Enable (GPIODEN)** register (see page 814) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 816) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 817) have been set.

## GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x510
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | PUE | RW | - | Pad Weak Pull-Up Enable |

| Value | Description |
|---|---|
| 0 | The corresponding pin's weak pull-up resistor is disabled. |
| 1 | The corresponding pin's weak pull-up resistor is enabled. |

Setting a bit in the **GPIOPDR** register clears the corresponding bit in the **GPIOPUR** register. The change is effective on the next clock cycle.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 773.

## Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514

The **GPIOPDR** register is the pull-down control register. When a bit is set, a weak pull-down resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 809).

**Important:** The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 10-10. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

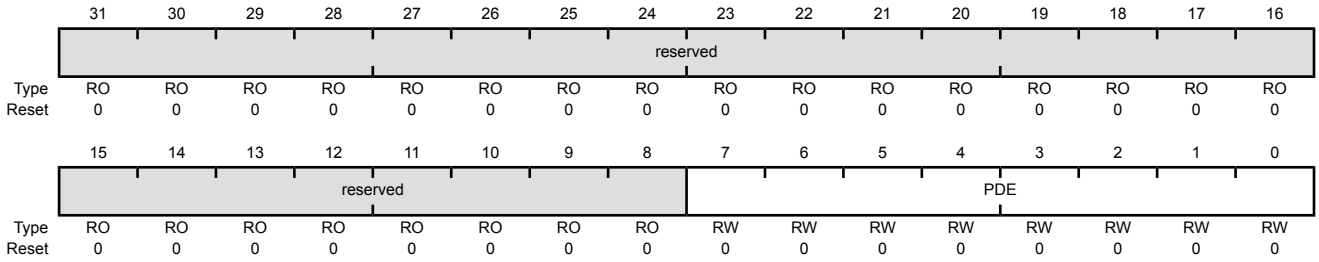
The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 784.

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 2028 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803), **GPIO Pull Up Select (GPIOPUR)** register (see page 809), **GPIO Pull-Down Select (GPIOPDR)** register (see page 811), and **GPIO Digital Enable (GPIODEN)** register (see page 814) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 816) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 817) have been set.

## GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x514
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | PDE | RW | 0x00 | Pad Weak Pull-Down Enable |

| Value | Description |
|---|---|
| 0 | The corresponding pin's weak pull-down resistor is disabled. |
| 1 | The corresponding pin's weak pull-down resistor is enabled. |

Setting a bit in the **GPIOPUR** register clears the corresponding bit in the **GPIOPDR** register. The change is effective on the next clock cycle.

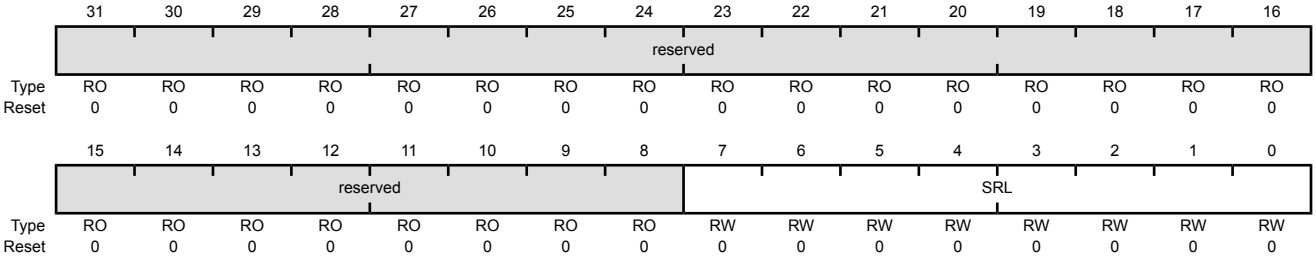
## Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA, 10-mA or 12-mA drive strength option. The selection of drive strength is done through the **GPIO Drive Select (GPIODRnR** registers and the **GPIO Peripheral Configuration (GPIOPC)** register.

**Note:** This register has no effect on port pins PL6 and PL7.

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x518
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | SRL | RW | 0x00 | Slew Rate Limit Enable (8-mA, 10-mA and 12-mA drive only)<br>Value Description<br>0 Slew rate control is disabled for the corresponding pin.<br>1 Slew rate control is enabled for the corresponding pin. |

## Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C

**Note:** Pins configured as digital inputs are Schmitt-triggered.

The **GPIODEN** register is the digital enable register. By default, all GPIO signals except those listed below are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

**Important:** The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 10-11. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

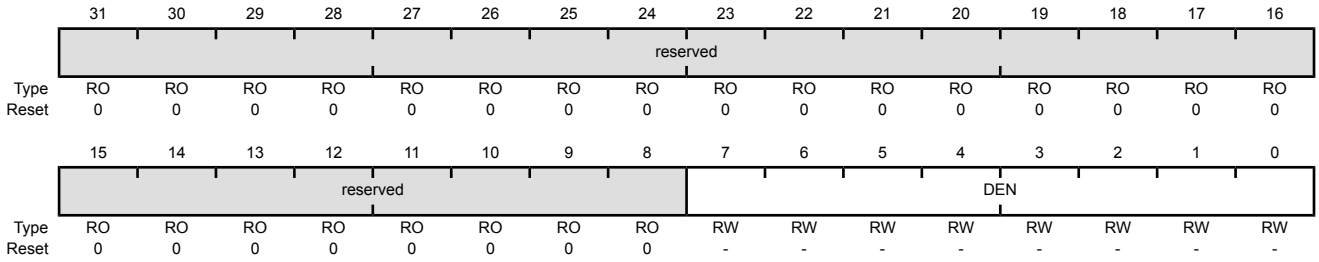
The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 784.

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the GPIO pins that can be used as the four JTAG/SWD pins and the NMI pin (see “Signal Tables” on page 2028 for pin numbers). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803), **GPIO Pull Up Select (GPIOPUR)** register (see page 809), **GPIO Pull-Down Select (GPIOPDR)** register (see page 811), and **GPIO Digital Enable (GPIODEN)** register (see page 814) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 816) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 817) have been set.

## GPIO Digital Enable (GPIODEN)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x51C
Type RW, reset -



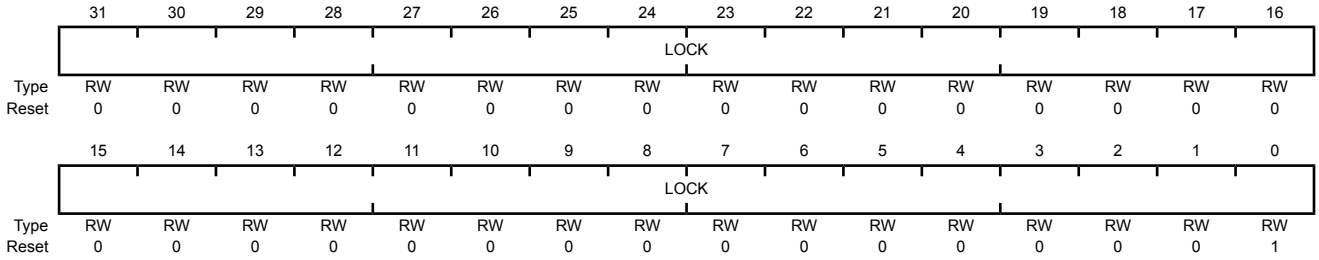

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DEN | RW | - | Digital Enable<br><br>Value Description<br>0 The digital functions for the corresponding pin are disabled.<br>1 The digital functions for the corresponding pin are enabled.<br>The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 773. |

## Register 19: GPIO Lock (GPIOLOCK), offset 0x520

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 817). Writing 0x4C4F.434B to the **GPIOLOCK** register unlocks the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x0000.0001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x0000.0000.

GPIO Lock (GPIOLOCK)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x520
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LOCK | RW | 0x0000.0001 | GPIO Lock<br>A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access.A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates.<br>A read of this register returns the following values:<br>Value Description<br>0x1 The **GPIOCR** register is locked and may not be modified.<br>0x0 The **GPIOCR** register is unlocked and may be modified. |

## Register 20: GPIO Commit (GPIOCR), offset 0x524

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, and **GPIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is cleared, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers is committed to the register and reflects the new value.

The contents of the **GPIOCR** register can only be modified if the status in the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the status in the **GPIOLOCK** register is locked.

**Important:** This register is designed to prevent accidental programming of the registers that control connectivity to the NMI and JTAG/SWD debug hardware. By initializing the bits of the **GPIOCR** register to 0 for the NMI and JTAG/SWD pins (see “Signal Tables” on page 2028 for pin numbers), the NMI and JTAG/SWD debug port can only be converted to GPIOs through a deliberate set of writes to the **GPIOLOCK**, **GPIOCR**, and the corresponding registers.

Because this protection is currently only implemented on the NMI and JTAG/SWD pins (see “Signal Tables” on page 2028 for pin numbers), all of the other bits in the **GPIOCR** registers cannot be written with 0x0. These bits are hardwired to 0x1, ensuring that it is always possible to commit new values to the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** register bits of these other pins.

GPIO Commit (GPIOCR)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x524
Type -, reset -

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | CR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | - | - | - | - | - | - | - | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | CR | - | - | GPIO Commit<br>Value Description<br>0 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits cannot be written.<br>1 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits can be written.<br>**Note:** The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the `NMI` pin and the four JTAG/SWD pins (see “Signal Tables” on page 2028 for pin numbers). These six pins are the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for the corresponding GPIO Ports is RW.<br>The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the `NMI` and JTAG/SWD pins (see “Signal Tables” on page 2028 for pin numbers). To ensure that the JTAG and `NMI` pins are not accidentally programmed as GPIO pins, these pins default to non-committable. Because of this, the default reset value of **GPIOCR** changes for the corresponding ports. |

## Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

**Important:** This register is only valid for ports and pins that can be used as ADC AINx inputs.

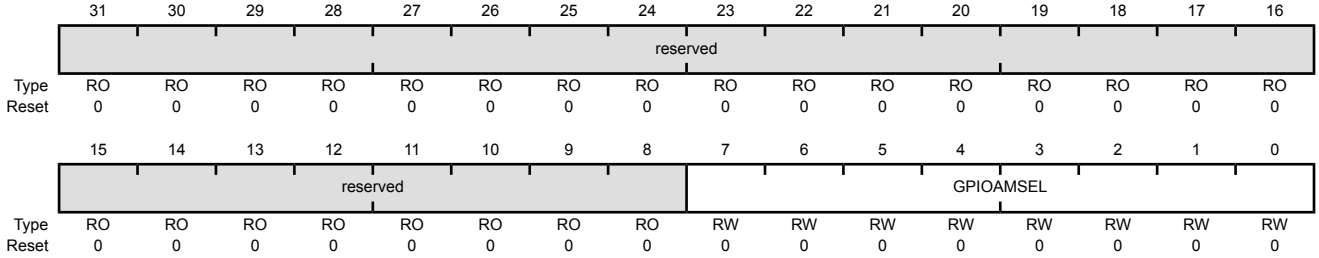
If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be set to disable the analog isolation circuit.

The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 3.3-V source and affect analog operation, analog circuitry requires isolation from the pins when they are not used in their analog function.

Each bit of this register controls the isolation circuitry for the corresponding GPIO signal. For information on which GPIO pins can be used for ADC functions, refer to Table 31-5 on page 2081.

GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x528
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | GPIOAMSEL | RW | 0x00 | GPIO Analog Mode Select |

| Value | Description |
|---|---|
| 0 | The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers. |
| 1 | The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions. |

**Note:** This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad.

The reset state of this register is 0 for all signals.

## Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C

The **GPIOPCTL** register is used in conjunction with the **GPIOAFSEL** register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the **GPIOAFSEL** register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the **GPIOAFSEL** register, the corresponding GPIO signal is controlled by an associated peripheral. The **GPIOPCTL** register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the defined encodings for the bit fields in this register, refer to Table 31-5 on page 2081. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

**Note:** If a particular input signal to a peripheral is assigned to two different GPIO port pins, the signal is assigned to the port with the lowest letter and the assignment to the higher letter port is ignored. If a particular output signal from a peripheral is assigned to two different GPIO port pins, the signal will output to both pins. Assigning an output signal from a peripheral to two different GPIO pins is not recommended.

**Important:** The table below shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 10-12. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

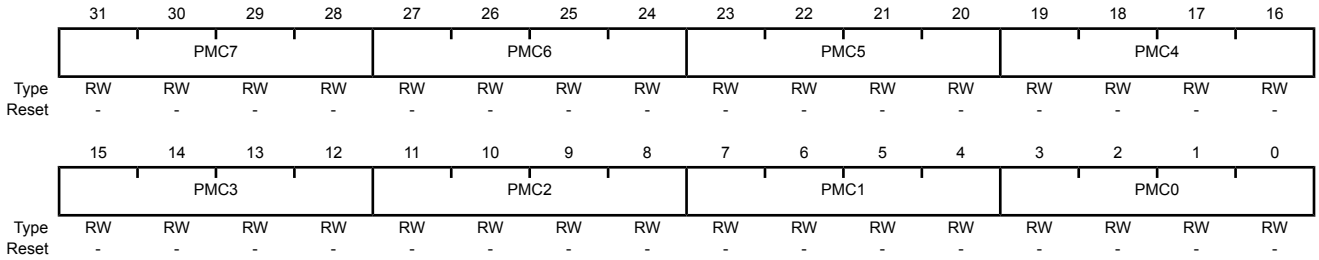
a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. The commit control process must be followed for these pins, even if they are programmed as alternate functions other than JTAG/SWD or NMI; see “Commit Control” on page 784.

**Note:** If the device fails initialization during reset, the hardware toggles the TDO output as an indication of failure. Thus, during board layout, designers should not designate the TDO pin as a GPIO in sensitive applications where the possibility of toggling could affect the design.

## GPIO Port Control (GPIOPCTL)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x52C
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | PMC7 | RW | - | Port Mux Control 7<br>This field controls the configuration for GPIO pin 7. |
| 27:24 | PMC6 | RW | - | Port Mux Control 6<br>This field controls the configuration for GPIO pin 6. |
| 23:20 | PMC5 | RW | - | Port Mux Control 5<br>This field controls the configuration for GPIO pin 5. |
| 19:16 | PMC4 | RW | - | Port Mux Control 4<br>This field controls the configuration for GPIO pin 4. |
| 15:12 | PMC3 | RW | - | Port Mux Control 3<br>This field controls the configuration for GPIO pin 3. |
| 11:8 | PMC2 | RW | - | Port Mux Control 2<br>This field controls the configuration for GPIO pin 2. |
| 7:4 | PMC1 | RW | - | Port Mux Control 1<br>This field controls the configuration for GPIO pin 1. |
| 3:0 | PMC0 | RW | - | Port Mux Control 0<br>This field controls the configuration for GPIO pin 0. |

## Register 23: GPIO ADC Control (GPIOADCCTL), offset 0x530

This register is used to configure a GPIO pin as a source for the ADC trigger.

Note that if the Port B **GPIOADCCTL** register is cleared, PB4 can still be used as an external trigger for the ADC. This is a legacy mode which allows code written for previous devices to operate on this microcontroller.

GPIO ADC Control (GPIOADCCTL)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x530
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | ADCEN | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | ADCEN | RW | 0x00 | ADC Trigger Enable |

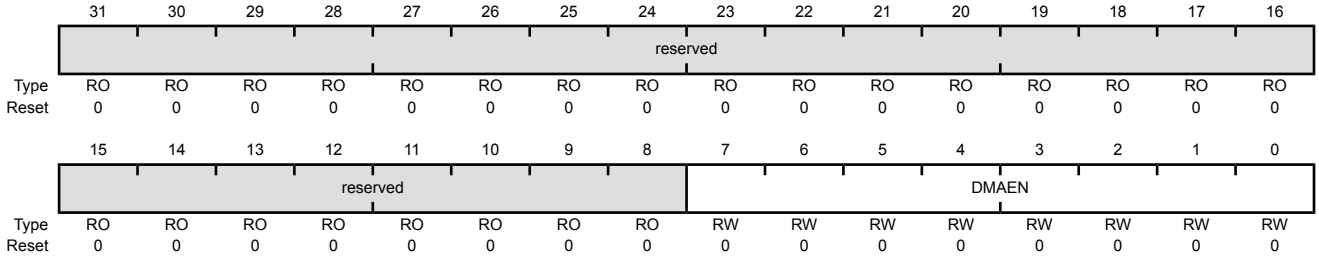
| Value | Description |
|---|---|
| 0 | The corresponding pin is not used to trigger the ADC. |
| 1 | The corresponding pin is used to trigger the ADC. |

## Register 24: GPIO DMA Control (GPIODMACTL), offset 0x534

This register is used to configure a GPIO pin as a source for the µDMA trigger.

GPIO DMA Control (GPIODMACTL)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x534
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | DMAEN | RW | 0x00 | µDMA Trigger Enable |

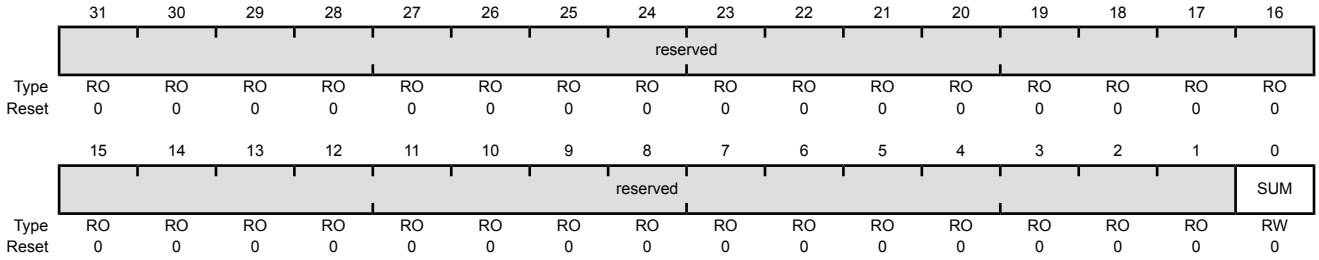
| Value | Description |
|---|---|
| 0 | The corresponding pin is not used to trigger the µDMA. |
| 1 | The corresponding pin is used to trigger the µDMA. |

## Register 25: GPIO Select Interrupt (GPIOSI), offset 0x538

This register is used to enable individual interrupts for each pin.

**Note:** This register is only available on Port P and Port Q.

GPIO Select Interrupt (GPIOSI)
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x538
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | SUM | RW | 0 | Summary Interrupt |

| Value | Description |
|---|---|
| 0 | All port pin interrupts are OR'ed together to produce a summary interrupt.<br><br>**Note:** The OR'ed summary interrupt occurs on bit 0 of the **GPIORIS** register. For summary interrupt mode, software should set the **GPIOIM** register to 0xFF and mask the port pin interrupts 1 through 7 in the **Interrupt Clear Enable (DISn)** register (see "NVIC Register Descriptions" on page 164). When servicing this interrupt, write a 1 to the corresponding bit in the **UNPENDn** register to clear the pending interrupt in the NVIC and clear the **GPIORIS** register pin interrupt bits by setting the `IC` field of the **GPIOICR** register to 0xFF. |
| 1 | Each pin has its own interrupt vector. |

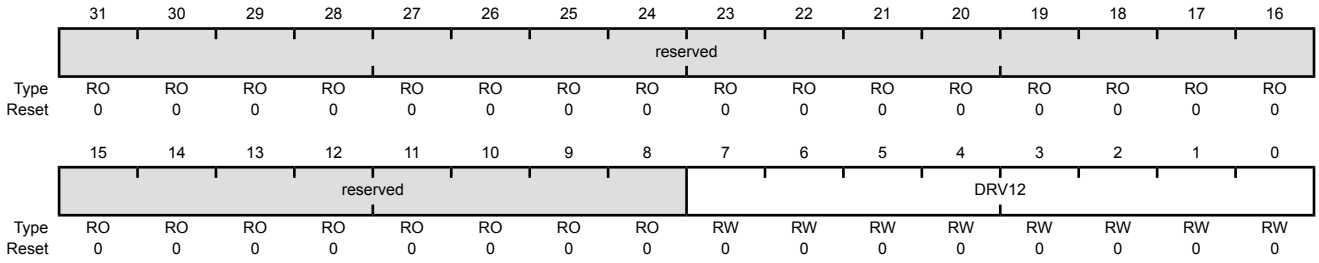
## Register 26: GPIO 12-mA Drive Select (GPIODR12R), offset 0x53C

The **GPIODR12R** register is the 12-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads.

**Note:** This register has no effect on port pins PL6 and PL7 or PM[7:4].

GPIO 12-mA Drive Select (GPIODR12R)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x53C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | DRV12 | RW | 0x00 | Output Pad 12-mA Drive Enable |

| Value | Description |
|---|---|
| 0 | The drive for the corresponding GPIO pin is controlled by the **GPIODR2R**, **GPIODR4R**, and/or the **GPIODR8R** register. |
| 1 | The corresponding GPIO pin has 12-mA drive. This encoding is only valid if the **GPIOPP** EDE bit is set and the appropriate **GPIOPC** EDM bit field is programmed to 0x3. |

**Note:** Please refer to Table 10-3 on page 785 for information on how to configure the drive strength.

Changes in the **GPIODR2R**, the **GPIODR4R** register and/or the **GPIODR8R** registers to configure 12 mA are effective on the next clock cycle.

## Register 27: GPIO Wake Pin Enable (GPIOWAKEPEN), offset 0x540

This register is used to configure K[7:4] as a wake enable source for the hibernation module. The wake level must be programmed in the **GPIOWAKELVL** register at offset 0x544. In order for this register configuration to become implemented, the WUUNLK bit needs to be set in the **HIBIO** register at offset 0x02C in the hibernation module.

**Note:** This register is only available on Port K.

GPIO Wake Pin Enable (GPIOWAKEPEN)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x540
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | WAKEP7 | WAKEP6 | WAKEP5 | WAKEP4 | reserved | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.000 | 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. |
| 7 | WAKEP7 | RW | 0 | K[7] Wake Enable<br>Value Description<br>0 Wake-on level is not enabled.<br>1 Wake-on level is enabled. |
| 6 | WAKEP6 | RW | 0 | K[6] Wake Enable<br>Value Description<br>0 Wake-on level is not enabled.<br>1 Wake-on level is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | WAKEP5 | RW | 0 | K[5] Wake Enable<br><br>Value Description<br>0 Wake-on level is not enabled.<br>1 Wake-on level is enabled. |
| 4 | WAKEP4 | RW | 0 | K[4] Wake Enable<br><br>Value Description<br>0 Wake-on level is not enabled.<br>1 Wake-on level is enabled. |
| 3:0 | reserved | RO | 0 | 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. |

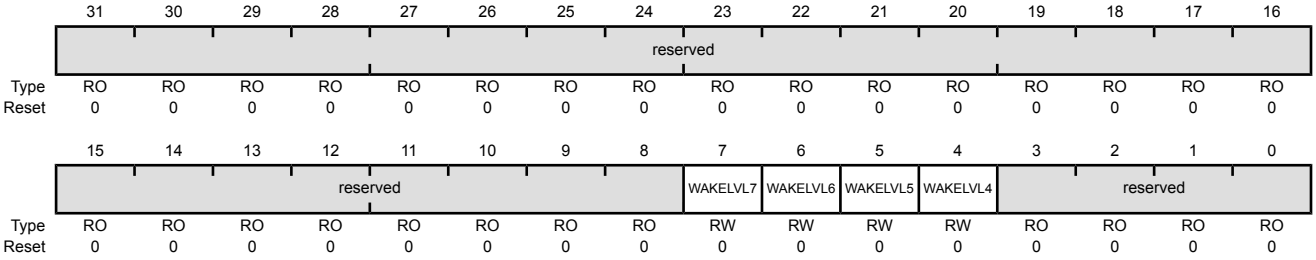
## Register 28: GPIO Wake Level (GPIOWAKELVL), offset 0x544

This register is used to configure the wake level for K[7:4] in the hibernation module. The wake source must be enabled in the **GPIOWAKEPEN** register at offset 0x540. In order for this register configuration to become implemented, the WUUNLK bit needs to be set in the **HIBIO** register at offset 0x02C in the hibernation module.

**Note:** This register is only available on Port K.

GPIO Wake Level (GPIOWAKELVL)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x544
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.000 | 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. |
| 7 | WAKELVL7 | RW | 0 | K[7] Wake Level<br>Value Description<br>0 Wake level low<br>1 Wake level high |
| 6 | WAKELVL6 | RW | 0 | K[6] Wake Level<br>Value Description<br>0 Wake level low<br>1 Wake level high |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | WAKELVL5 | RW | 0 | K[5] Wake Level<br>Value Description<br>0 Wake level low<br>1 Wake level high |
| 4 | WAKELVL4 | RW | 0 | K[4] Wake Level<br>Value Description<br>0 Wake level low<br>1 Wake level high |
| 3:0 | reserved | RO | 0 | 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. |

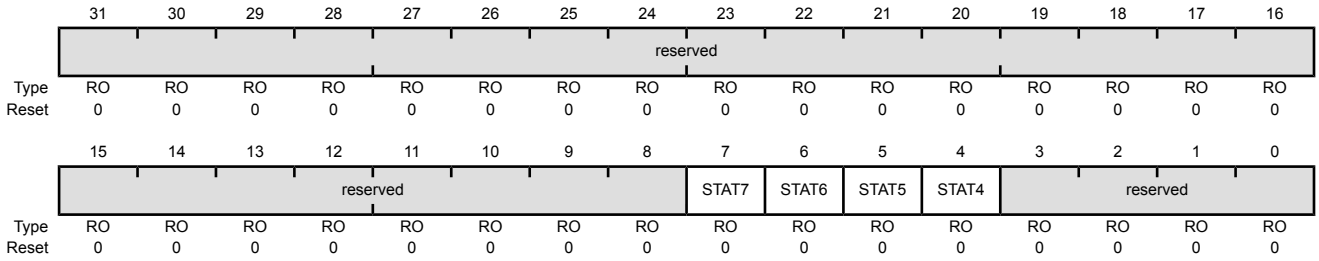
## Register 29: GPIO Wake Status (GPIOWAKESTAT), offset 0x548

This register indicates the GPIO wake event status. If a register bit has been set for K[7:4] , a wake event signal has been sent to the Hibernate module.

**Note:** This register is only available on Port K.

GPIO Wake Status (GPIOWAKESTAT)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0x548
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.000 | 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. |
| 7 | STAT7 | RO | 0 | K[7] Wake Status<br>This is for future use.<br><br>Value Description<br>0 Pin is not wake up source<br>1 Pin wake event asserted to hibernate module |
| 6 | STAT6 | RO | 0 | K[6] Wake Status<br>This is for future use.<br><br>Value Description<br>0 Pin is not wake up source<br>1 Pin wake event asserted to hibernate module |

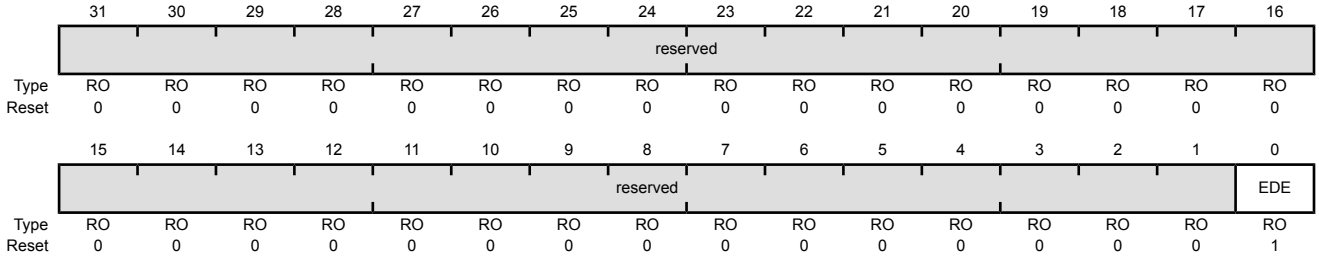
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | STAT5 | RO | 0 | K[5] Wake Status<br>This is for future use.<br><br>Value Description<br>0 Pin is not wake up source<br>1 Pin wake event asserted to hibernate module |
| 4 | STAT4 | RO | 0 | K[4] Wake Status<br><br>Value Description<br>0 Pin is not wake up source<br>1 Pin wake event asserted to hibernate module |
| 3:0 | reserved | RO | 0 | 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. |

## Register 30: GPIO Peripheral Property (GPIOPP), offset 0xFC0

The **GPIOPP** register provides information regarding the GPIO properties.

GPIO Peripheral Property (GPIOPP)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFC0
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0 | 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. |
| 0 | EDE | RO | 0x1 | Extended Drive Enable<br>This bit specifies whether the extended drive capabilities are provided. Extended drive is configured by the EDM bits in the **GPIOPC** register.<br>Value Description<br>0 No Extended Drive Capability provided.<br>1 Extended Drive Capability provided. |

## Register 31: GPIO Peripheral Configuration (GPIOPC), offset 0xFC4

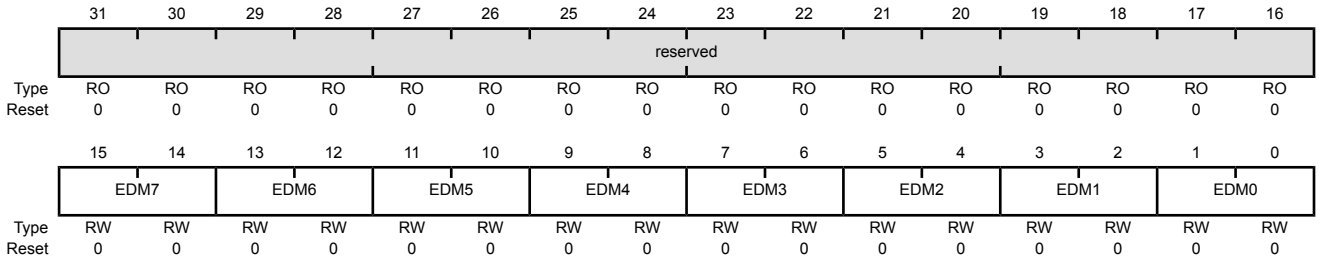
This **GPIOPC** register controls the extended drive modes of the GPIO and must be configured before the **GPIODRnR** registers in order for extended drive mode to take effect. When the EDE bit in **GPIOPP** register is set and the EDMn bit field is non-zero, the **GPIODRnR** registers do not drive their default value, but instead output an incremental drive strength, which has an additive effect. This allows for more drive strength possibilities. When the EDE bit is set and the EDMn bit field is non-zero, the 2 mA driver is always enabled. Any bits enabled in the **GPIODR4R** register will add an additional 2 mA; any bits set in the **GPIODR8R** add an extra 4 mA of drive. The **GPIODR12R** register is only valid when the EDMn value is 0x3. For this encoding, setting a bit in the **GPIODR12R** register adds 4 mA of drive to the already existing 8 mA, for a 12 mA drive strength. Table 10-3 on page 785 shows the drive capability options. If EDMn is 0x00, then the **GPIODR2R**, **GPIODR4R**, and **GPIODR8R** function as stated in their default register description.

**Table 10-13. GPIO Drive Strength Options**

| EDE (GPIOPP) | EDMn (GPIOPC) | GPIODR12R (+4mA) | GPIODR8R (+4mA) | GPIODR4R (+2mA) | GPIODR2R (2mA) | Drive (mA) |
|---|---|---|---|---|---|---|
| X | 0x0 | N/A | 0 | 0 | 1 | 2 |
| | | | 0 | 1 | 0 | 4 |
| | | | 1 | 0 | 0 | 8 |
| 1 | 0x1 | N/A | 0 | 0 | N/A | 2 |
| | | | 0 | 1 | N/A | 4 |
| | | | 1 | 0 | N/A | 6 |
| | | | 1 | 1 | N/A | 8 |
| 1 | 0x3 | 0 | 0 | 0 | N/A | 2 |
| | | 0 | 0 | 1 | N/A | 4 |
| | | 0 | 1 | 0 | N/A | 6 |
| | | 0 | 1 | 1 | N/A | 8 |
| | | 1 | 1 | 0 | N/A | 10 |
| | | 1 | 1 | 1 | N/A | 12 |
| | | 1 | 0 | N/A | N/A | N/A |
| 1 | 0x2 | N/A | N/A | N/A | N/A | N/A |

## GPIO Peripheral Configuration (GPIOPC)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFC4
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000.000 | 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. |
| 15:14 | EDM7 | RW | 0 | Extended Drive Mode Bit 7<br>Same encoding as EDM0, but applies to bit 7 of GPIO port. |
| 13:12 | EDM6 | RW | 0 | Extended Drive Mode Bit 6<br>Same encoding as EDM0, but applies to bit 6 of GPIO port. |
| 11:10 | EDM5 | RW | 0 | Extended Drive Mode Bit 5<br>Same encoding as EDM0, but applies to bit 5 of GPIO port. |
| 9:8 | EDM4 | RW | 0 | Extended Drive Mode Bit 4<br>Same encoding as EDM0, but applies to bit 4 of GPIO port. |
| 7:6 | EDM3 | RW | 0 | Extended Drive Mode Bit 3<br>Same encoding as EDM0, but applies to bit 3 of GPIO port. |
| 5:4 | EDM2 | RW | 0 | Extended Drive Mode Bit 2<br>Same encoding as EDM0, but applies to bit 2 of GPIO port. |
| 3:2 | EDM1 | RW | 0 | Extended Drive Mode Bit 1<br>Same encoding as EDM0, but applies to bit 1 of GPIO port. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | EDM0 | RW | 0 | Extended Drive Mode Bit 0<br>This field controls extended drive modes of bit 0 of the GPIO port.<br>Note that depending on the encoding used the GPIO drive strength control registers may change their decoding. Moreover, the write one, clear other register behavior may be disabled. |

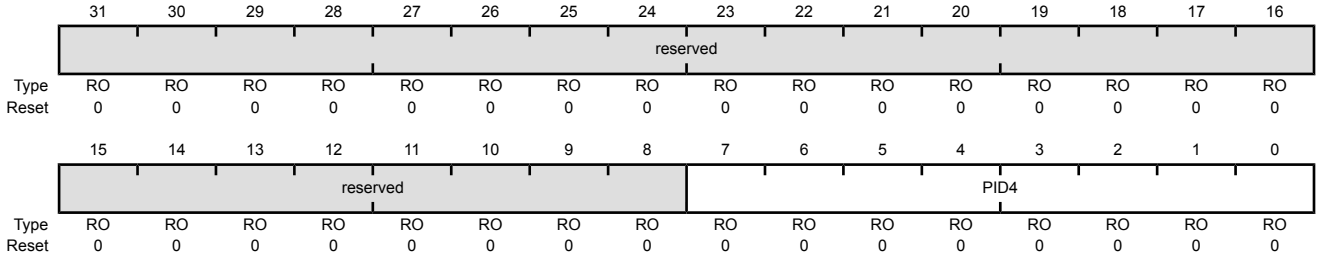
| Value | Description |
|---|---|
| 0x0 | Drive values of 2, 4 and 8 mA are maintained. **GPIO n Drive Select (GPIODRnR)** registers function as normal. |
| 0x1 | An additional 6 mA option is provided.<br>Write one, clear other behavior of **GPIODDRnR** registers is disabled.<br>A 2 mA driver is always enabled; setting the corresponding **GPIODR4R** register bit adds 2 mA and setting the corresponding **GPIODR8R** register bit adds an additional 4 mA. |
| 0x2 | reserved |
| 0x3 | Additional drive strength options of 6, 10, and 12 mA are provided.<br>The write one, clear other behavior of **GPIODDRnR** registers is disabled.<br>A 2 mA driver is always enabled; setting the corresponding **GPIODR4R** register bit adds 2 mA and setting the corresponding **GPIODR8R** of **GPIODR12R** register bit adds an additional 4 mA. |

## Register 32: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 4 (GPIOPeriphID4)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFD0
Type RO, reset 0x0000.0000



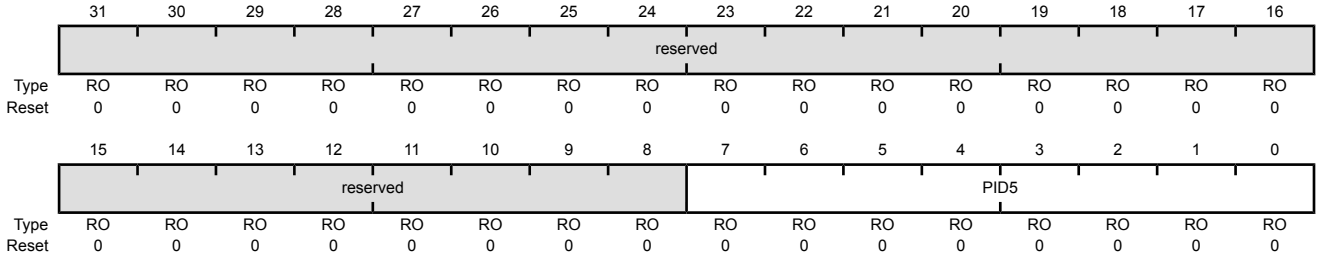

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | GPIO Peripheral ID Register [7:0] |

## Register 33: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 5 (GPIOPeriphID5)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFD4
Type RO, reset 0x0000.0000



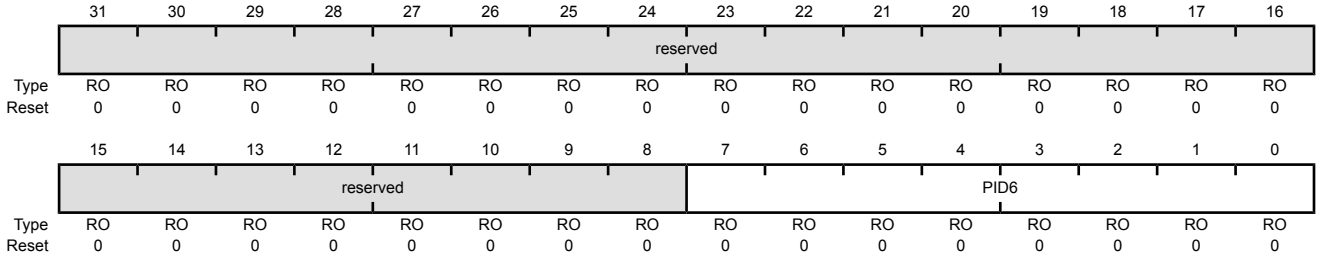

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | GPIO Peripheral ID Register [15:8] |

## Register 34: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 6 (GPIOPeriphID6)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFD8
Type RO, reset 0x0000.0000



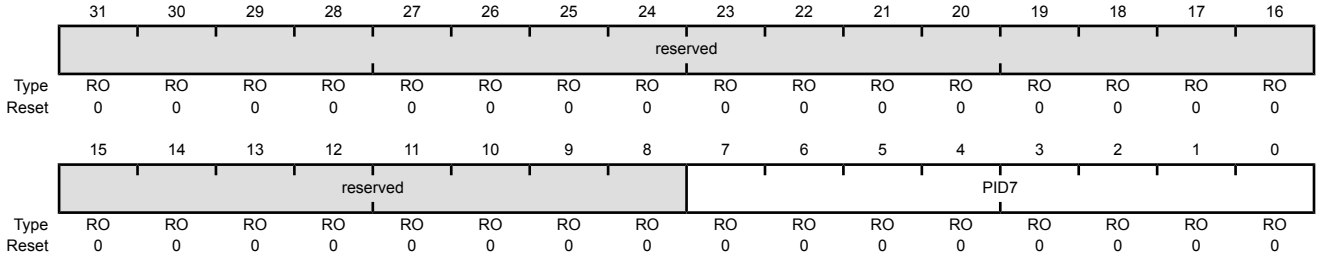

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | GPIO Peripheral ID Register [23:16] |

## Register 35: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 7 (GPIOPeriphID7)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFDC
Type RO, reset 0x0000.0000



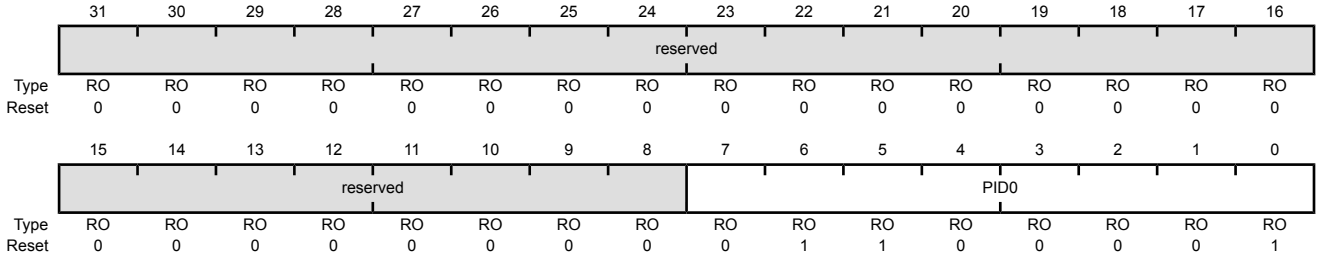

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | GPIO Peripheral ID Register [31:24] |

## Register 36: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 0 (GPIOPeriphID0)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFE0
Type RO, reset 0x0000.0061



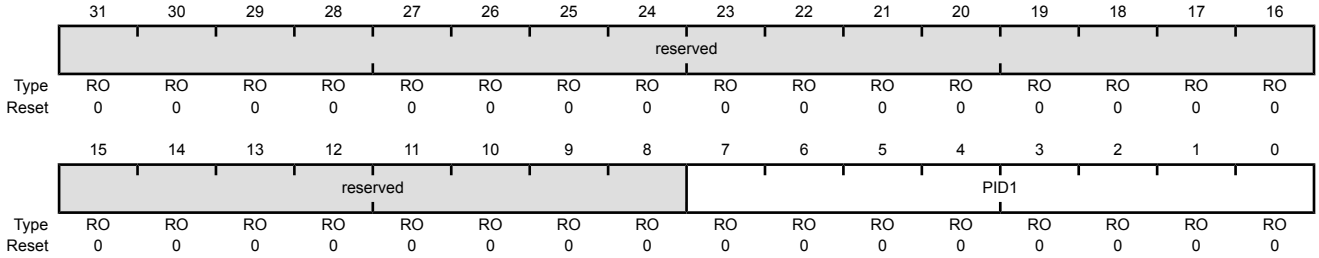

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x61 | GPIO Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 37: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 1 (GPIOPeriphID1)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFE4
Type RO, reset 0x0000.0000



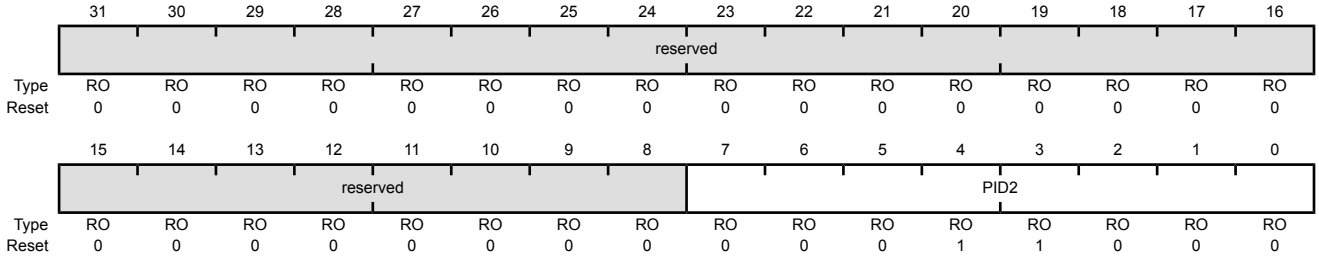

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x00 | GPIO Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 38: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 2 (GPIOPeriphID2)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFE8
Type RO, reset 0x0000.0018



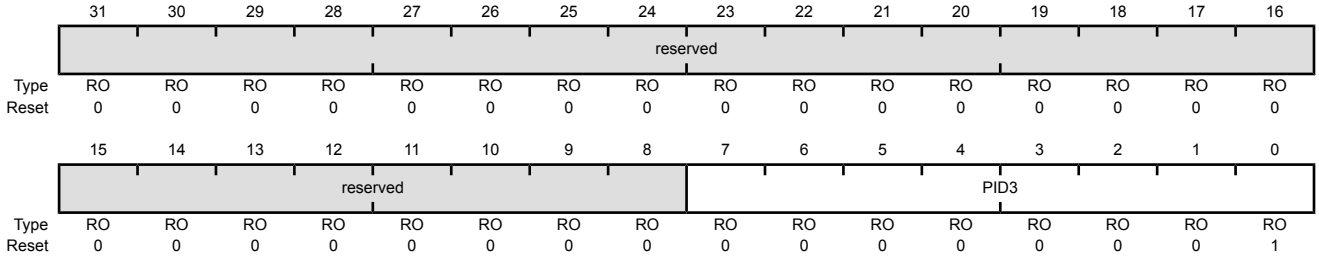

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | GPIO Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 39: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 3 (GPIOPeriphID3)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFEC
Type RO, reset 0x0000.0001



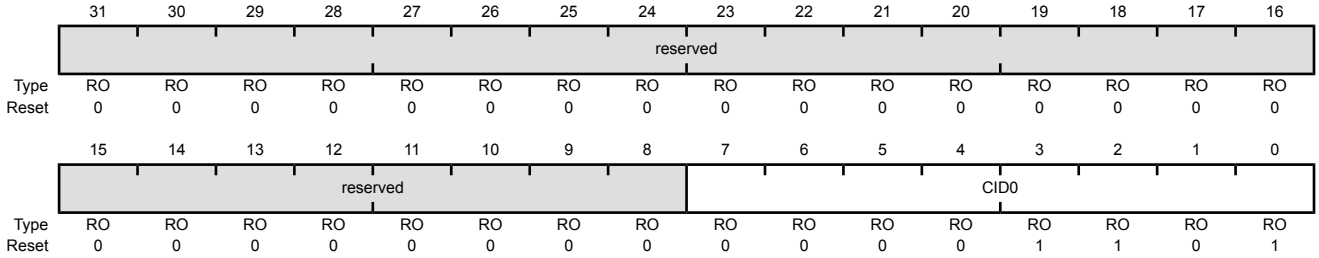

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | GPIO Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 40: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 0 (GPIOPCellID0)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFF0
Type RO, reset 0x0000.000D



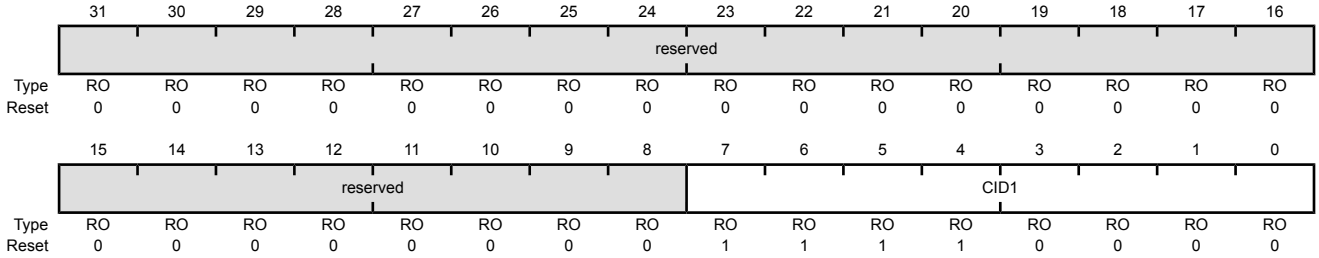

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | GPIO PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 41: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 1 (GPIOPCellID1)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFF4
Type RO, reset 0x0000.00F0



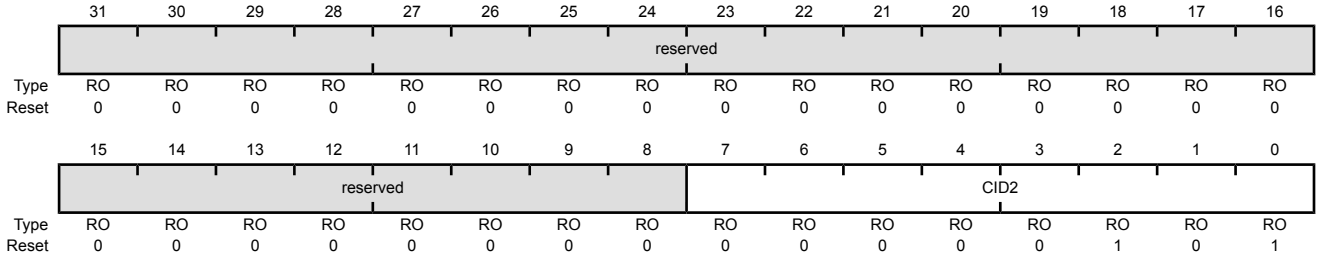

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | GPIO PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 42: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 2 (GPIOPCellID2)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFF8
Type RO, reset 0x0000.0005



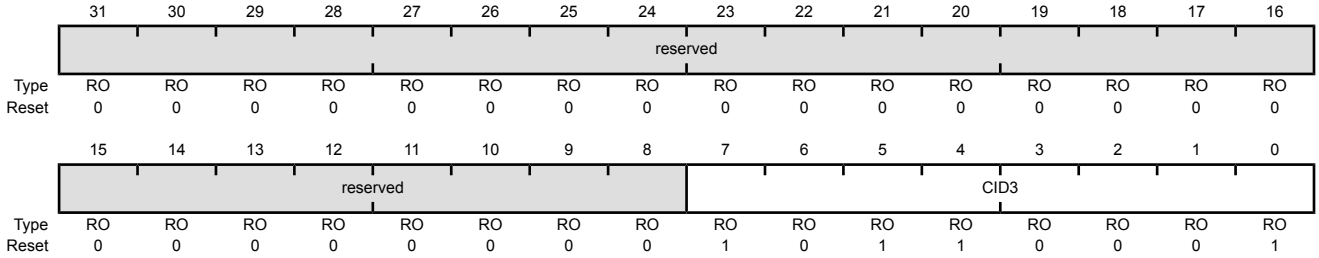

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | GPIO PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 43: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 3 (GPIOPCellID3)

GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (AHB) base: 0x4005.F000
GPIO Port J (AHB) base: 0x4006.0000
GPIO Port K (AHB) base: 0x4006.1000
GPIO Port L (AHB) base: 0x4006.2000
GPIO Port M (AHB) base: 0x4006.3000
GPIO Port N (AHB) base: 0x4006.4000
GPIO Port P (AHB) base: 0x4006.5000
GPIO Port Q (AHB) base: 0x4006.6000
GPIO Port R (AHB) base: 0x4006.7000
GPIO Port S (AHB) base: 0x4006.8000
GPIO Port T (AHB) base: 0x4006.9000
Offset 0xFFC
Type RO, reset 0x0000.00B1




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | CID3 | RO | 0xB1 | GPIO PrimeCell ID Register [31:24]<br>Provides software a standard cross-peripheral identification system. |

# 11 External Peripheral Interface (EPI)

The External Peripheral Interface is a high-speed parallel bus for external peripherals or memory. It has several modes of operation to interface gluelessly to many types of external devices. The External Peripheral Interface is similar to a standard microprocessor address/data bus, except that it must typically be connected to just one type of external device. Enhanced capabilities include µDMA support, clocking control and support for external FIFO buffers.

The EPI has the following features:

- 8/16/32-bit dedicated parallel bus for external peripherals and memory
- Memory interface supports contiguous memory access independent of data bus width, thus enabling code execution directly from SDRAM, SRAM and Flash memory
- Blocking and non-blocking reads
- Separates processor from timing details through use of an internal write FIFO
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for read and write
  - Read channel request asserted by programmable levels on the internal Non-Blocking Read FIFO (NBRFIFO)
  - Write channel request asserted by empty on the internal Write FIFO (WFIFO)

The EPI supports three primary functional modes: Synchronous Dynamic Random Access Memory (SDRAM) mode, Traditional Host-Bus mode, and General-Purpose mode. The EPI module also provides custom GPIOs; however, unlike regular GPIOs, the EPI module uses a FIFO in the same way as a communication mechanism and is speed-controlled using clocking.

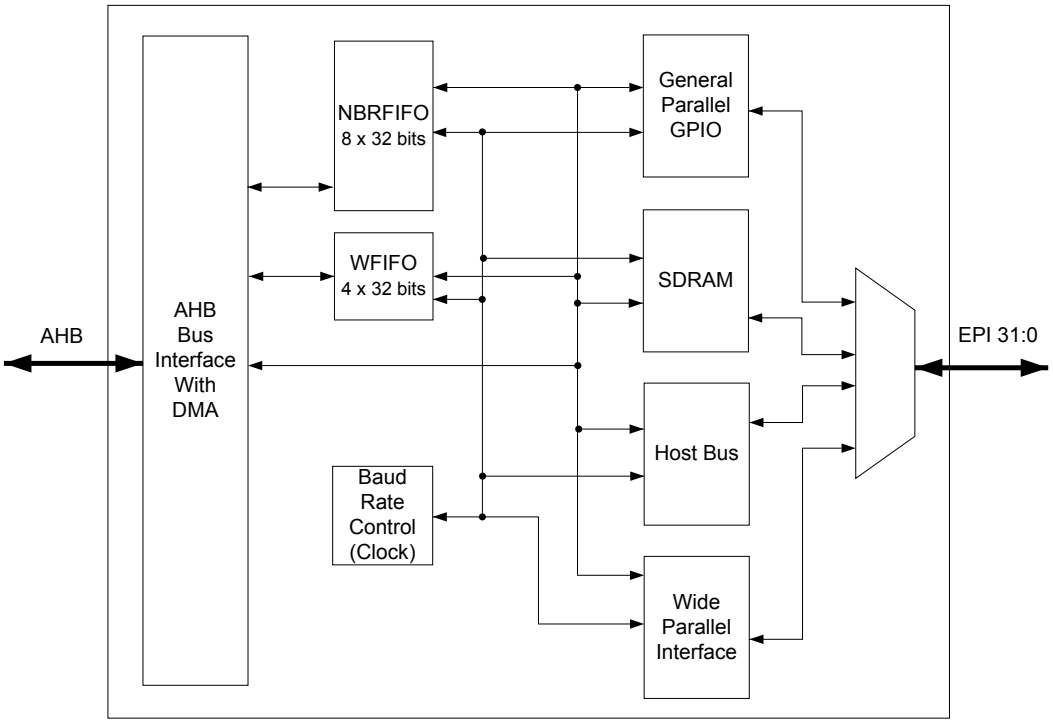
- Synchronous Dynamic Random Access Memory (SDRAM) mode
  - Supports x16 (single data rate) SDRAM at up to 60 MHz
  - Supports low-cost SDRAMs up to 64 MB (512 megabits)
  - Includes automatic refresh and access to all banks/rows
  - Includes a Sleep/Standby mode to keep contents active with minimal power draw
  - Multiplexed address/data interface for reduced pin count
- Host-Bus mode
  - Traditional x8 and x16 MCU bus interface capabilities
  - Similar device compatibility options as PIC, ATmega, 8051, and others
  - Access to SRAM, NOR Flash memory, and other devices, with up to 1 MB of addressing in non-multiplexed mode and 256 MB in multiplexed mode (512 MB in Host-Bus 16 mode with no byte selects)

  - Support for up to 512 Mb PSRAM in quad chip select mode, with dedicated configuration register read and write enable.
  - Support of both muxed and de-muxed address and data
  - Access to a range of devices supporting the non-address FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) EMPTY and FULL signals
  - Speed controlled, with read and write data wait-state counters
  - Support for read/write burst mode to Host Bus
  - Multiple chip select modes including single, dual, and quad chip selects, with and without ALE
  - External iRDY signal provided for stall capability of reads and writes
  - Manual chip-enable (or use extra address pins)
- General-Purpose mode
  - Wide parallel interfaces for fast communications with CPLDs and FPGAs
  - Data widths up to 32 bits
  - Data rates up to 150 MB/second
  - Optional "address" sizes from 4 bits to 20 bits
  - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
  - 1 to 32 bits, FIFOed with speed control
  - Useful for custom peripherals or for digital data acquisition and actuator controls

## 11.1 EPI Block Diagram

Figure 11-1 on page 850 provides a block diagram of a TM4C129XKCZAD EPI module.

**Figure 11-1. EPI Block Diagram**




## 11.2 Signal Description

The following table lists the external signals of the EPI controller and describes the function of each. The EPI controller signals are alternate functions for GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the EPI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the EPI controller function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the EPI signals to the specified GPIO port pins. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 11-1. External Peripheral Interface Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| EPI0S0 | P4<br>J1 | PH0 (15)<br>PK0 (15) | I/O | TTL | EPI module 0 signal 0. |
| EPI0S1 | R2<br>J2 | PH1 (15)<br>PK1 (15) | I/O | TTL | EPI module 0 signal 1. |
| EPI0S2 | R1<br>K1 | PH2 (15)<br>PK2 (15) | I/O | TTL | EPI module 0 signal 2. |
| EPI0S3 | T1<br>K2 | PH3 (15)<br>PK3 (15) | I/O | TTL | EPI module 0 signal 3. |
| EPI0S4 | K3 | PC7 (15) | I/O | TTL | EPI module 0 signal 4. |
| EPI0S5 | L2 | PC6 (15) | I/O | TTL | EPI module 0 signal 5. |
| EPI0S6 | M1 | PC5 (15) | I/O | TTL | EPI module 0 signal 6. |
| EPI0S7 | M2 | PC4 (15) | I/O | TTL | EPI module 0 signal 7. |

**Table 11-1. External Peripheral Interface Signals (212BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| EPI0S8 | V5 | PA6 (15) | I/O | TTL | EPI module 0 signal 8. |
| EPI0S9 | R7 | PA7 (15) | I/O | TTL | EPI module 0 signal 9. |
| EPI0S10 | T14 | PG1 (15) | I/O | TTL | EPI module 0 signal 10. |
| EPI0S11 | N15 | PG0 (15) | I/O | TTL | EPI module 0 signal 11. |
| EPI0S12 | L19 | PM3 (15) | I/O | TTL | EPI module 0 signal 12. |
| EPI0S13 | L18 | PM2 (15) | I/O | TTL | EPI module 0 signal 13. |
| EPI0S14 | K19 | PM1 (15) | I/O | TTL | EPI module 0 signal 14. |
| EPI0S15 | K18 | PM0 (15) | I/O | TTL | EPI module 0 signal 15. |
| EPI0S16 | G16 | PL0 (15) | I/O | TTL | EPI module 0 signal 16. |
| EPI0S17 | H19 | PL1 (15) | I/O | TTL | EPI module 0 signal 17. |
| EPI0S18 | G18 | PL2 (15) | I/O | TTL | EPI module 0 signal 18. |
| EPI0S19 | J18 | PL3 (15) | I/O | TTL | EPI module 0 signal 19. |
| EPI0S20 | E3 | PQ0 (15) | I/O | TTL | EPI module 0 signal 20. |
| EPI0S21 | E2 | PQ1 (15) | I/O | TTL | EPI module 0 signal 21. |
| EPI0S22 | H4 | PQ2 (15) | I/O | TTL | EPI module 0 signal 22. |
| EPI0S23 | M4 | PQ3 (15) | I/O | TTL | EPI module 0 signal 23. |
| EPI0S24 | W16 | PK7 (15) | I/O | TTL | EPI module 0 signal 24. |
| EPI0S25 | V16 | PK6 (15) | I/O | TTL | EPI module 0 signal 25. |
| EPI0S26 | H18 | PL4 (15) | I/O | TTL | EPI module 0 signal 26. |
| EPI0S27 | A17 | PB2 (15) | I/O | TTL | EPI module 0 signal 27. |
| EPI0S28 | B17 | PB3 (15) | I/O | TTL | EPI module 0 signal 28. |
| EPI0S29 | A11<br>B13 | PN2 (15)<br>PP2 (15) | I/O | TTL | EPI module 0 signal 29. |
| EPI0S30 | B10<br>C12 | PN3 (15)<br>PP3 (15) | I/O | TTL | EPI module 0 signal 30. |
| EPI0S31 | V17 | PK5 (15) | I/O | TTL | EPI module 0 signal 31. |
| EPI0S32 | U19 | PK4 (15) | I/O | TTL | EPI module 0 signal 32. |
| EPI0S33 | G19 | PL5 (15) | I/O | TTL | EPI module 0 signal 33. |
| EPI0S34 | A10 | PN4 (15) | I/O | TTL | EPI module 0 signal 34. |
| EPI0S35 | B9 | PN5 (15) | I/O | TTL | EPI module 0 signal 35. |

## 11.3 Functional Description

The EPI controller provides a glueless, programmable interface to a variety of common external peripherals such as SDRAM x 16, Host Bus x8 and x16 devices, RAM, NOR Flash memory, CPLDs and FPGAs. In addition, the EPI controller provides custom GPIO that can use a FIFO with speed control by using either the internal write FIFO (WFIFO) or the non-blocking read FIFO (NBRFIFO). The WFIFO can hold 4 words of data that are written to the external interface at the rate controlled by the **EPI Main Baud Rate (EPIBAUD)** registers. The NBRFIFO can hold 8 words of data and samples at the rate controlled by the **EPIBAUD** register. The EPI controller provides predictable operation and thus has an advantage over regular GPIO which has more variable timing due to on-chip bus arbitration and delays across bus bridges. Blocking reads stall the CPU until the transaction completes. Non-blocking reads are performed in the background and allow the processor

to continue operation. In addition, write data can also be stored in the WFIFO to allow multiple writes with no stalls.

**Note:** Both the WTAV bit field in the **EPIWFIFOCNT** register and the WBUSY bit in the **EPISTAT** register must be polled to determine if there is a current write transaction from the WFIFO. If both of these bits are clear, then a new bus access may begin.

Main read and write operations can be performed in subsets of the range 0x6000.0000 to 0xDFFF.FFFF. A read from an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value load, the read is done as a burst (when available) to maximize performance. A write to an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value store, the write is done as a burst (when available) to maximize performance.

### 11.3.1 Master Access to EPI

The following lists the Bus Masters which have access to the EPI:

- CPU
- µDMA
- LCD

### 11.3.2 Non-Blocking Reads

The EPI Controller supports a special kind of read called a non-blocking read, also referred to as a posted read. Where a normal read stalls the processor or µDMA until the data is returned, a non-blocking read is performed in the background.

A non-blocking read is configured by writing the start address into a **EPIRADDRn** register, the size per transaction into a **EPIRSIZEn** register, and then the count of operations into a **EPIRPSTDn** register. After each read is completed, the result is written into the NBRFIFO and the **EPIRADDRn** register is incremented by the size (1, 2, or 4). The three most significant bits of **EPIRADDRn** register are only relevant in the Host Bus multi-chip select mode when they are used to enable the different chip selects.

If the NBRFIFO is filled, then the reads pause until space is made available. The NBRFIFO can be configured to interrupt the processor or trigger the µDMA based on fullness using the **EPIFIFOLVL** register. By using the trigger/interrupt method, the µDMA (or processor) can keep space available in the NBRFIFO and allow the reads to continue unimpeded.

When performing non-blocking reads, the SDRAM controller issues two additional read transactions after the burst request is terminated. The data for these additional transfers is discarded. This situation is transparent to the user other than the additional EPI bus activity and can safely be ignored.

Two non-blocking read register sets are available to allow sequencing and ping-pong use. When one completes, the other then activates. So, for example, if 20 words are to be read from 0x100 and 10 words from 0x200, the **EPIRPSTD0** register can be set up with the read from 0x100 (with a count of 20), and the **EPIRPSTD1** register can be set up with the read from 0x200 (with a count of 10). When **EPIRPSTD0** finishes (count goes to 0), the **EPIRPSTD1** register then starts its operation. The NBRFIFO has then passed 30 values. When used with the µDMA, it may transfer 30 values (simple sequence), or the primary/alternate model may be used to handle the first 20 in one way and the second 10 in another. It is also possible to reload the **EPIRPSTD0** register when it is finished (and the **EPIRPSTD1** register is active); thereby, keeping the interface constantly busy.

To cancel a non-blocking read, the **EPIRPSTDn** register is cleared. Care must be taken, however if the register set was active to drain away any values read into the NBRFIFO and ensure that any read in progress is allowed to complete.

To ensure that the cancel is complete, the following algorithm is used (using the **EPIRPSTD0** register for example):

**EPIRPSTD0** = 0;

while ((**EPISTAT** & 0x11) == 0x10)

; // we are active and busy

// if here, then other one is active or interface no longer busy

cnt = (**EPIRADDR0** – original_address) / **EPIRSIZE0**; // count of values read

cnt -= values_read_so_far;

// cnt is now number left in FIFO

while (cnt--)

value = **EPIREADFIFO**; // drain

The above algorithm can be optimized in code; however, the important point is to wait for the cancel to complete because the external interface could have been in the process of reading a value when the cancel came in, and it must be allowed to complete.

### 11.3.3 DMA Operation

The µDMA can be used to achieve maximum transfer rates on the EPI through the NBRFIFO and the WFIFO. The µDMA has one channel for write and one for read. For writes, the **EPI DMA Transmit Count (EPIDMATXCNT)** register is programmed with the total number of transfers by the µDMA. An equivalent value is programmed into the **DMA Channel Control Word (DMACHCTL)** register of the uDMA at offset 0x008. A µDMA request is asserted by the EPI WRFIFO when the TXCNT value of the **EPIDMATXCNT** register is greater than zero and the WTAV bit field of the **EPIWFIFOCNT** register is less than the programmed threshold trigger, WRFIFO, of the **EPIFIFOLVL** register. The write channel continues to write data until the TXCNT value in the **EPIDMATXCNT** register is zero.

**Note:** When the WRFIFO bit in the **EPIFIFOLVL** register is set to 0x4 and the application bursts four words to an empty FIFO, the WRFIFO trigger may or may not deassert depending on if all four words were written to the WRFIFO or if the first word was passed immediately to the function requiring it. Thus, the application may not see the WRRIS bit in the **EPIRIS** register clear on a burst of four words.

The non-blocking read channel copies values from the NBRFIFO when the NBRFIFO is at the level specified by the **EPIFIFOLVL** register. For non-blocking reads, the start address, the size per transaction, and the count of elements must be programmed in the µDMA. Note that both non-blocking read register sets can be used, and they fill the NBRFIFO such that one runs to completion, then the next one starts (they do not interleave). Using the NBRFIFO provides the best possible transfer rate.

For blocking reads, the µDMA software channel (or another unused channel) is used for memory-to-memory transfers (or memory to peripheral, where some other peripheral is used). In this situation, the µDMA stalls until the read is complete and is not able to service another channel until the read is done. As a result, the arbitration size should normally be programmed to one access at a time. The µDMA controller can also transfer from and to the NBRFIFO and the WFIFO using the µDMA software channel in memory mode, however, the µDMA is stalled once the NBRFIFO is

empty or the WFIFO is full. Note that when the µDMA controller is stalled, the core continues operation. See “Micro Direct Memory Access (µDMA)” on page 708 for more information on configuring the µDMA.

The size of the FIFOs must be taken into consideration when configuring the µDMA to transfer data to and from the EPI. The arbitration size should be 4 or less when writing to EPI address space and 8 or less when reading from EPI address space.

## 11.4 Initialization and Configuration

To enable and initialize the EPI controller, the following steps are necessary:

1. Enable the EPI module using the **RCGCEPI** register. See page 403.
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register. See page 399. To find out which GPIO port to enable, refer to “Signal Description” on page 850.
3. Set the GPIO AFSEL bits for the appropriate pins. See page 803. To determine which GPIOs to configure, see Table 31-4 on page 2064.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected. See page 805 and page 813.
5. Configure the PMCn fields in the **GPIOPCTL** register to assign the EPI signals to the appropriate pins. See page 820 and Table 31-5 on page 2081.
6. Select the mode for the EPI block to SDRAM, HB8, HB16, or general parallel use, using the MODE field in the **EPI Configuration (EPICFG)** register. Set the mode-specific details (if needed) using the appropriate mode configuration **EPI Host Bus Configuration (EPIHBnCFGn)** registers for the desired chip-select configuration. Set the **EPI Main Baud Rate (EPIBAUD)** and **EPI Main Baud Rate 2 (EPIBAUD2)** register if the baud rate must be slower than the system clock rate.
7. Configure the address mapping using the **EPI Address Map (EPIADDRMAP)** register. The selected start address and range is dependent on the type of external device and maximum address (as appropriate). For example, for a 512-megabit SDRAM, program the ERADR field to 0x1 for address 0x6000.0000 or 0x2 for address 0x8000.0000; and program the ERSZ field to 0x3 for 256 MB. If using General-Purpose mode and no address at all, program the EPADR field to 0x1 for address 0xA000.0000 or 0x2 for address 0xC000.0000; and program the EPSZ field to 0x0 for 256 bytes.
8. To read or write directly, use the mapped address area (configured with the **EPIADDRMAP** register). Up to 4 or 5 writes can be performed at once without blocking. Each read is blocked until the value is retrieved.
9. To perform a non-blocking read, see “Non-Blocking Reads” on page 852.

**Note:** The application should not attempt to access externally until eight system clock cycles after the EPI has been fully configured.

**Note:** Once a MODE field has been programmed in the **EPICFG** register, the application should reset all configuration registers before re-programming to a new MODE value.

The following sub-sections describe the initialization and configuration for each of the modes of operation. Care must be taken to initialize everything properly to ensure correct operation. Control of the GPIO states is also important, as changes may cause the external device to interpret pin

states as actions or commands (see “Register Descriptions” on page 790). Normally, a pull-up or pull-down is needed on the board to at least control the chip-select or chip-enable as the TM4C129XKCZAD GPIOs come out of reset in tri-state.

### 11.4.1 EPI Interface Options

There are a variety of memories and peripherals that can interface to the EPI module. Table 11-2 on page 855 shows the various configurations with their maximum performance.

**Table 11-2. EPI Interface Options**

| Interface | Maximum Frequency |
|---|---|
| Single SDRAM | 60 MHz |
| Single SRAM | 60 MHz |
| Single PSRAM without iRDY signal use | 55 MHz |
| Single PSRAM with iRDY signal use | 52 MHz |
| FPGAs, CPLDs, etc using General Purpose Mode | 60 MHz |
| Memory configurations with 2 chip selects | 40 MHz |
| Memory configurations with 4 chip selects | 20 MHz |

### 11.4.2 SDRAM Mode

When activating the SDRAM mode, it is important to consider a few points:

1. Generally, it takes over 100 µs from when the mode is activated to when the first operation is allowed. The SDRAM controller begins the SDRAM initialization sequence as soon as the mode is selected and enabled via the **EPICFG** register. It is important that the GPIOs are properly configured before the SDRAM mode is enabled, as the EPI controller is relying on the GPIO block's ability to drive the pins immediately. As part of the initialization sequence, the LOAD MODE REGISTER command is automatically sent to the SDRAM with a value of 0x27, which sets a CAS latency of 2 and a full page burst length.

2. The INITSEQ bit in the **EPI Status (EPISTAT)** register can be checked to determine when the initialization sequence is complete.

3. When using a frequency range and/or refresh value other than the default value, it is important to configure the FREQ and RFSH fields in the **EPI SDRAM Configuration (EPISDRAMCFG)** register shortly after activating the mode. After the 100-µs startup time, the EPI block must be configured properly to keep the SDRAM contents stable.

4. The SLEEP bit in the **EPISDRAMCFG** register may be configured to put the SDRAM into a low-power self-refreshing state. It is important to note that the SDRAM mode must not be disabled once enabled, or else the SDRAM is no longer clocked and the contents are lost.

5. Before entering SLEEP mode, make sure all non-blocking reads and normal reads and writes have completed. If the system is running at 30 to 50 MHz, wait 2 EPI clocks after clearing the SLEEP bit before executing non-blocking reads, or normal reads and writes. If the system is configured to greater than 50 MHz, wait 5 EPI clocks before read and write transactions. For all other configurations, wait 1 EPI clock.

The SIZE field of the **EPISDRAMCFG** register must be configured correctly based on the amount of SDRAM in the system.

The FREQ field must be configured according to the value that represents the range being used. Based on the range selected, the number of external clocks used between certain operations (for example, PRECHARGE or ACTIVATE) is determined. If a higher frequency is given than is used, then the only downside is that the peripheral is slower (uses more cycles for these delays). If a lower frequency is given, incorrect operation occurs.

See "External Peripheral Interface (EPI)" on page 2130 for timing details for the SDRAM mode.

#### 11.4.2.1 External Signal Connections

Table 11-3 on page 856 defines how EPI module signals should be connected to SDRAMs. The table applies when using a x16 SDRAM up to 512 megabits. Note that the EPI signals must use 8-mA drive when interfacing to SDRAM, see page 807. Any unused EPI controller signals can be used as GPIOs or another alternate function.

**Table 11-3. EPI SDRAM x16 Signal Connections**

| EPI Signal | SDRAM Signal[a] | |
|---|---|---|
| EPI0S0 | A0 | D0 |
| EPI0S1 | A1 | D1 |
| EPI0S2 | A2 | D2 |
| EPI0S3 | A3 | D3 |
| EPI0S4 | A4 | D4 |
| EPI0S5 | A5 | D5 |
| EPI0S6 | A6 | D6 |
| EPI0S7 | A7 | D7 |
| EPI0S8 | A8 | D8 |
| EPI0S9 | A9 | D9 |
| EPI0S10 | A10 | D10 |
| EPI0S11 | A11 | D11 |
| EPI0S12 | A12[b] | D12 |
| EPI0S13 | BA0 | D13 |
| EPI0S14 | BA1 | D14 |
| EPI0S15 | D15 | |
| EPI0S16 | DQML | |
| EPI0S17 | DQMH | |
| EPI0S18 | CASn | |
| EPI0S19 | RASn | |
| EPI0S20-EPI0S27 | not used | |
| EPI0S28 | WEn | |
| EPI0S29 | CSn | |
| EPI0S30 | CKE | |
| EPI0S31 | CLK | |

a. If two signals are listed, connect the EPI signal to both pins.

b. Only for 256/512 megabit SDRAMs.

#### 11.4.2.2 Refresh Configuration

The refresh count is based on the external clock speed and the number of rows per bank as well as the refresh period. The RFSH field represents how many external clock cycles remain before an AUTO-REFRESH is required. The normal formula is:

RFSH = ($t_{Refresh_us}$ / number_rows) / ext_clock_period

A refresh period is normally 64 ms, or 64000 µs. The number of rows is normally 4096 or 8192. The ext_clock_period is a value expressed in µsec and is derived by dividing 1000 by the clock speed expressed in MHz. So, 50 MHz is 1000/50=20 ns, or 0.02 µs. A typical SDRAM is 4096 rows per bank if the system clock is running at 50 MHz with an **EPIBAUD** register value of 0:

RFSH = (64000/4096) / 0.02 = 15.625 µs / 0.02 µs = 781.25

The default value in the RFSH field is 750 decimal or 0x2EE to allow for a margin of safety and providing 15 µs per refresh. It is important to note that this number should always be smaller or equal to what is required by the above equation. For example, if running the external clock at 25 MHz (40 ns per clock period), 390 is the highest number that may be used. Note that the external clock may be 25 MHz when the system clock is 25 MHz or when the system clock is 50 MHz and configuring the COUNT0 field in the **EPIBAUD** register to 1 (divide by 2).

If a number larger than allowed is used, the SDRAM is not refreshed often enough, and data is lost.

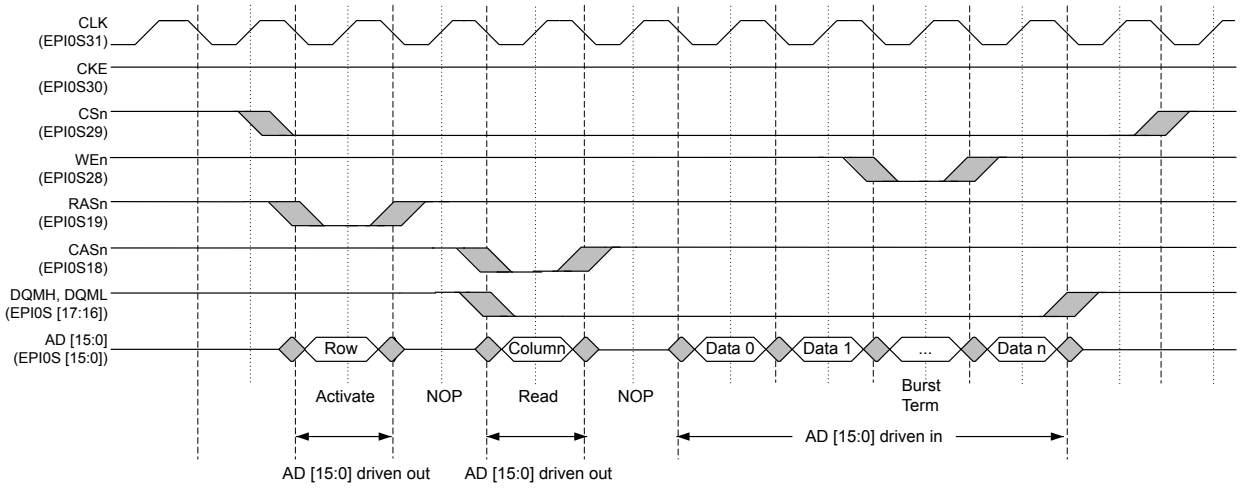
#### 11.4.2.3 Bus Interface Speed

The EPI Controller SDRAM interface can operate up to 60 MHz. The COUNT0 field in the **EPIBAUD** register configures the speed of the EPI clock. For system clock (SysClk) speeds up to 60 MHz, the COUNT0 field can be 0x0000, and the SDRAM interface can run at the same speed as SysClk. However, if SysClk is running at higher speeds, the bus interface can run only as fast as half speed, and the COUNT0 field must be configured to at least 0x0001.

#### 11.4.2.4 Non-Blocking Read Cycle

Figure 11-2 on page 858 shows a non-blocking read cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed CAS latency of 2, the Read command with the column address on the EPI0S[15:0] signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the EPI0S[15:0] signals on every rising clock edge. The Burst Terminate command is issued during the cycle when the next-to-last halfword is read in. The DQMH and DQML signals are deasserted after the last halfword of data is received; the CSn signal deasserts on the following clock cycle, signaling the end of the read cycle. At least one clock period of inactivity separates any two SDRAM cycles.

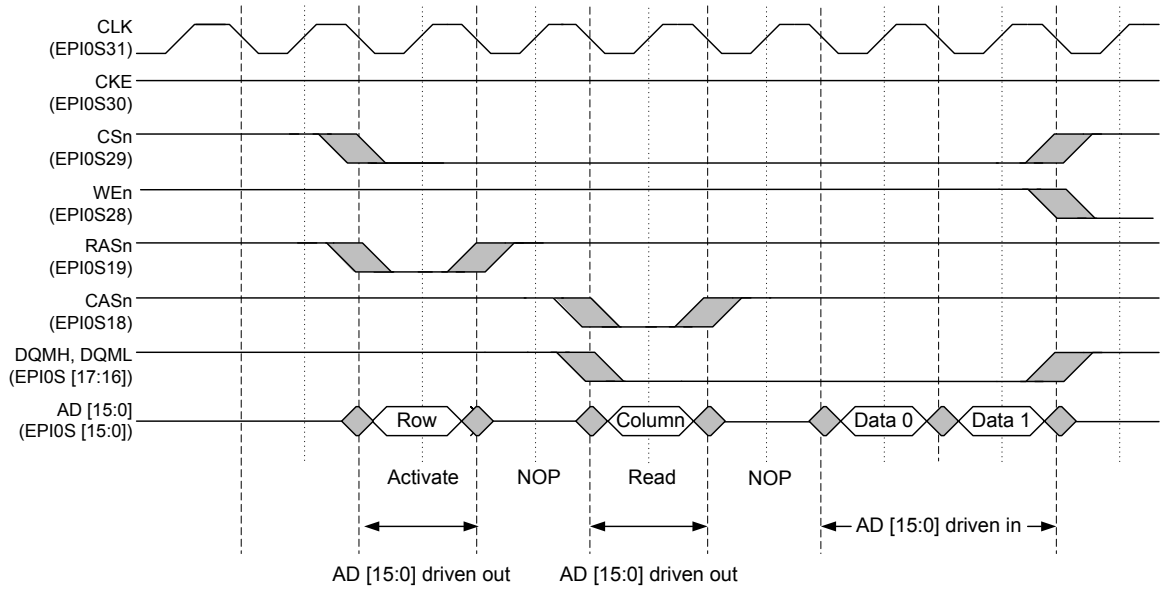
**Figure 11-2. SDRAM Non-Blocking Read Cycle**




### 11.4.2.5 Normal Read Cycle

Figure 11-3 on page 858 shows a normal read cycle of n halfwords; n can be 1 or 2. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed CAS latency of 2, the Read command with the column address on the EPI0S[15:0] signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the EPI0S[15:0] signals on every rising clock edge. The DQMH, DQML, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the cycle. At least one clock period of inactivity separates any two SDRAM cycles.

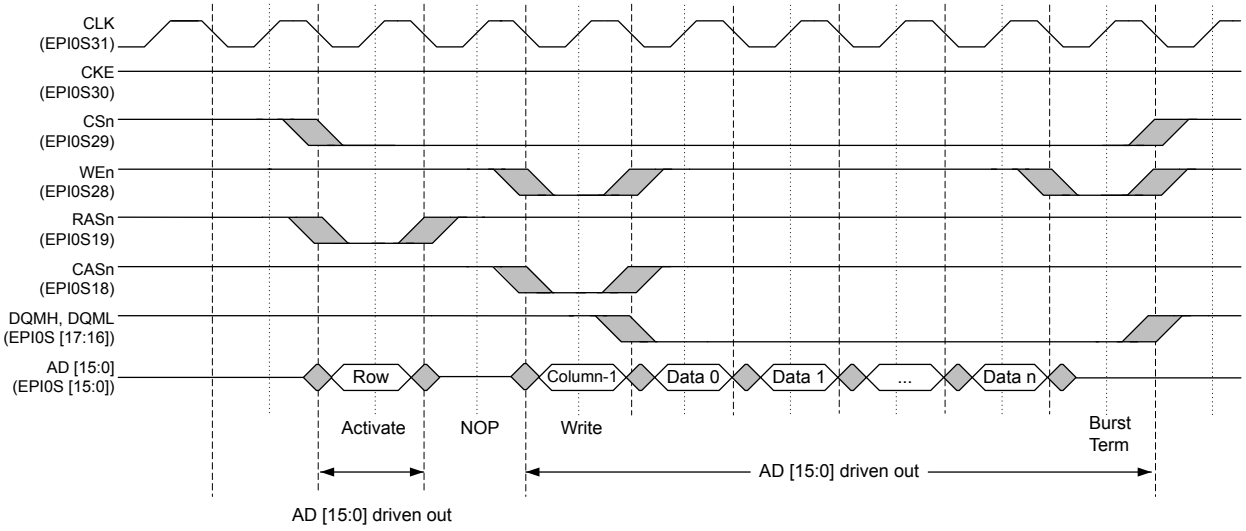
**Figure 11-3. SDRAM Normal Read Cycle**



#### 11.4.2.6 Write Cycle

Figure 11-4 on page 859 shows a write cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the EPI0S[15:0] signals. With the programmed CAS latency of 2, the Write command with the column address on the EPI0S[15:0] signals follows after 2 clock cycles. When writing to SDRAMs, the Write command is presented with the first halfword of data. Because the address lines and the data lines are multiplexed, the column address is modified to be (programmed address -1). During the Write command, the DQMH and DQML signals are high, so no data is written to the SDRAM. On the next clock, the DQMH and DQML signals are asserted, and the data associated with the programmed address is written. The Burst Terminate command occurs during the clock cycle following the write of the last halfword of data. The WEn, DQMH, DQML, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the access. At least one clock period of inactivity separates any two SDRAM cycles.

**Figure 11-4. SDRAM Write Cycle**




### 11.4.3 Host Bus Mode

Host Bus supports the traditional 8-bit and 16-bit interfaces popularized by the 8051 devices and SRAM devices, as well as PSRAM and NOR Flash memory. This interface is asynchronous and uses strobe pins to control activity. Addressable memory can be doubled using Host Bus-16 mode as it performs half-word accesses. The EPI0S0 is the LSB of the address and is equivalent to the internal Cortex-M4 A1 address. EPI0S0 should be connected to A0 of 16-bit memories.

#### 11.4.3.1 Control Pins

The main three strobes are Address Latch Enable (ALE), Write (WRn), and Read (RDn, sometimes called OEn). Note that the timings are designed for older logic and so are hold-time versus setup-time specific. The polarity of the read and write strobes can be active High or active Low by clearing or setting the RDHIGH and WRHIGH bits in the **EPI Host-Bus n Configuration (EPIHBnCFGn)** register.

The ALE can be changed to an active-low chip select signal, CSn, through the **EPIHBnCFGn** register. The ALE is best used for Host-Bus muxed mode in which EPI address and data pins are shared. All Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold it until the data phase. The polarity of the ALE

can be active High or Low by clearing or setting the ALEHIGH bit in the **EPI Host-Bus n Configuration (EPIHBnCFGn)** register. CSn is best used for Host-Bus unmuxed mode in which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access are occurring. Both the ALE and the CSn modes can be enhanced to access four external devices using settings in the **EPIHBnCFGn** register. PSRAM accesses must use both ALE and CSn . Wait states can be added to the data phase of the access using the WRWS and RDWS bits in the **EPIHBnCFGn** register. Additionally, within these wait state options, the WRWSM and RDWSM bit of the **EPIHBnTIMEn** register can be set to reduce the given wait states by 1 EPI clock cycle for finer granularity.

For FIFO mode, the ALE is not used, and two input holds are optionally supported to gate input and output to what the XFIFO can handle. FIFO mode is only applicable in EPI asynchronous mode.

Host-Bus 8 and Host-Bus 16 modes are very configurable. The user has the ability to connect 1,2, or 4 external devices to the EPI signals, as well as control whether byte select signals are provided in HB16 mode. These capabilities depend on the configuration of the MODE field in the **EPIHBnCFG** register, the CSCFG field and the CSCFGEXT bit in the **EPIHBnCFGn** register, and the BSEL bit in the **EPIHB16CFG** register. The CSCFGEXT bit extends the chip select configuration possibilities by providing the most significant bit of the CSCFG field. Refer to Table 11-4 on page 860 for the possible ALE and chip select options that can be programmed by the combination of the CSCFGEXT and CSCFG bits. Note that CSCFGEXT is the most significant bit.

**Table 11-4. CSCFGEXT + CSCFG Encodings**

| Value | Description |
|---|---|
| 0x0 | ALE Configuration<br>EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (MODE field in the **EPIHB8CFG** register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle. |
| 0x1 | CSn Configuration<br>EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (MODE field in the **EPIHB8CFG** register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low. |
| 0x2 | Dual CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral. |
| 0x3 | ALE with Dual CSn Configuration<br>EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. |
| 0x4 | ALE with Single CSn Configuration<br>EPI0S30 is used as address latch (ALE) and EPI0S27 is used as CSn. |
| 0x5 | Quad CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n. |
| 0x6 | ALE with Quad CSn Configuration<br>EPI0S30 is ALE, EPI0S26 is CS0n and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n. |
| 0x7 | Reserved |

If one of the Dual-Chip-Select modes is selected (CSCFGEXT is 0x0 and CSCFG is 0x2 or 0x3 in the **EPIHBnCFGn** register), both chip selects can share the peripheral, code, or the memory space, or one chip select can use the peripheral space and the other can use the memory or code space. In the **EPIADDRMAP** register, if the EPADR field is not 0x0, the ECADR field is 0x0, and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with CS0n being asserted when the MSB of the address range is 0 and CS1n being asserted when the MSB of the address range is 1. If the ERADR field is not 0x0, the ECADR field is 0x0, and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation. If both the EPADR and the ERADR are not 0x0, and the ECADR field is 0x0 and the EPI is configured for dual-chip selects, then CS0n is asserted for either address range defined by EPADR and CS1n is asserted for either address range defined by ERADR. The two chip selects can also be shared between the code space and memory or peripheral space. If the ECADR field is 0x1, ERADR field is 0x0, and the EPADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by EPADR. If the ECADR field is 0x1, EPADR field is 0x0, and the ERADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by ERADR.

In quad chip select mode (CSCFGEXT is 0x1 and CSCFG is 0x1 or 0x2 in the **EPIHBnCFG2** register), both the peripheral and the memory space must be enabled. In the **EPIADDRMAP** register, the EPADR field is 0x3, the ERADR field is 0x3, and the ECADR field is 0x0. With this configuration, CS0n asserts for the address range beginning at 0x6000.0000, CS1n asserts for 0x8000.0000, CS2n for 0xA000.0000, and CS3n for 0xC000.0000. Table 11-5 on page 861 gives a detailed explanation of chip select address range mappings based on combinations of enabled peripheral and memory space.

**Note:** Only one memory area can be mapped to a single chip select. Enabling multiple memory areas for one chip select may produce unexpected results.

**Table 11-5. Dual- and Quad- Chip Select Address Mappings**

| Chip Select Mode | ERADR | EPADR | ECADR | CS0[a] | CS1 | CS2 | CS3 |
|---|---|---|---|---|---|---|---|
| Dual-chip select | 0x0 | 0x1 or 0x2 | 0x0 | EPADR defined address range (0xA000.000 or 0xC000.0000) | EPADR defined address range (0xA000.000 or 0xC000.0000) | N/A | N/A |
| Dual-chip select | 0x1 or 0x2 | 0x0 | 0x0 | ERADR defined address range (0x6000.000 or 0x8000.000) | ERADR defined address range (0x6000.000 or 0x8000.000) | N/A | N/A |
| Dual-chip select | 0x1 or 0x2 | 0x1 or 0x2 | 0x0 | EPADR defined address range (0xA000.000 or 0xC000.0000) | ERADR defined address range (0x6000.000 or 0x8000.000) | N/A | N/A |
| Dual-chip select | 0x0 | 0x1 or 0x2 | 0x1 | ECADR defined address range (0x1000.000) | EPADR defined address range (0xA000.0000 or 0xC000.0000) | N/A | N/A |
| Dual-chip select | 0x1 or 0x2 | 0x0 | 0x1 | ECADR defined address range (0x1000.000) | ERADR defined address range (0x6000.000 or 0x8000.000) | N/A | N/A |
| Quad-chip select | 0x3 | 0x3 | 0x0 | 0x6000.0000 | 0x8000.0000 | 0xA000.0000 | 0xC000.0000 |

a. When CS0 & CS1 share address space, CS0 asserts when the MSB of the address is 0 and CS1, when the MSB of the address is '1.'

The MODE field of the **EPIHBnCFGn** registers configure the interface for the chip selects, which support ADMUX or ADNOMUX. See Table 11-6 on page 862 for details on which configuration register controls each chip select. If the CSBAUD bit is clear, all chip selects are configured by the MODE bit field of the **EPIHBnCFG** register.

If the CSBAUD bit in the **EPIHBnCFG2** register is set in Dual-chip select mode, the 2 chip selects can use different clock frequencies, wait states and strobe polarity. If the CSBAUD bit is clear, both chip selects use the clock frequency, wait states, and strobe polarity defined for CS0n. Additionally, if the CSBAUD bit is set, the two chip selects can use different interface modes. If any interface modes are programmed to ADMUX, then dual chip select mode must include the ALE capability. In quad chip select mode, if the CSBAUD bit in the **EPIHBnCFG2** register is set, the 4 chip selects can use different clock frequencies, wait states and strobe polarity. If the CSBAUD bit is clear, all chip selects use the clock frequency, wait states, and strobe polarity defined for CS0n. If the CSBAUD bit is set, the four chip selects can use different interface modes.

**Table 11-6. Chip Select Configuration Register Assignment**

| Configuration Register[a] | Corresponding Chip Select |
|---|---|
| EPIHBnCFG | CS0n |
| EPIHBnCFG2 | CS1n |
| EPIHBnCFG3 | CS2n |
| EPIHBnCFG4 | CS3n |

a. If the CSBAUD bit in the **EPIHBnCFG2** register is clear and multiple chip selects are enabled, then all chip selects are configured by the MODE bit field in the **EPIHBnCFG** register.

Note that multiple chip select modes do not allow the intermixing of Host-Bus 8 and Host-Bus16 modes.

When BSEL=1 in the **EPIHB16CFG** register, byte select signals are provided, so byte-sized data can be read and written at any address, however these signals reduce the available address width by 2 pins. The byte select signals are active Low. BSEL0n corresponds to the LSB of the halfword, and BSEL1n corresponds to the MSB of the halfword.

When BSEL=0, byte reads and writes at odd addresses only act on the even byte, and byte writes at even addresses write invalid values into the odd byte. As a result, accesses should be made as half-words (16-bits) or words (32-bits). In C/C++, programmers should use only short int and long int for accesses. Also, because data accesses in HB16 mode with no byte selects are on 2-byte boundaries, the available address space is doubled. For example, 28 bits of address accesses 512 MB in this mode. Table 11-7 on page 862 shows the capabilities of the HB8 and HB16 modes as well as the available address bits with the possible combinations of these bits.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system.

**Table 11-7. Capabilities of Host Bus 8 and Host Bus 16 Modes**

| Host Bus Type | MODE | CSCFGEXT | CSCFG | Max # of External Devices | BSEL | Byte Access | Available Address | Addressable Memory |
|---|---|---|---|---|---|---|---|---|
| HB8 | 0x0 | 0 | 0x0, 0x1 | 1 | N/A | Always | 28 bits | 256 MB |
| HB8 | 0x0 | 0 | 0x2 | 2 | N/A | Always | 27 bits | 128 MB |
| HB8 | 0x0 | 0 | 0x3 | 2 | N/A | Always | 26 bits | 64 MB |
| HB8 | 0x0 | 1 | 0x0 | 1 | N/A | Always | 27 bits | 128 MB |
| HB8 | 0x0 | 1 | 0x1 | 4 | N/A | Always | 27 bits | 128 MB |

**Table 11-7. Capabilities of Host Bus 8 and Host Bus 16 Modes *(continued)***

| Host Bus Type | MODE | CSCFGEXT | CSCFG | Max # of External Devices | BSEL | Byte Access | Available Address | Addressable Memory |
|---|---|---|---|---|---|---|---|---|
| HB8 | 0x0 | 1 | 0x2 | 4 | N/A | Always | 26 bits | 64 MB |
| HB8 | 0x1 | 0 | 0x0, 0x1 | 1 | N/A | Always | 20 bits | 1 MB |
| HB8 | 0x1 | 0 | 0x2 | 2 | N/A | Always | 19 bits | 512 kB |
| HB8 | 0x1 | 0 | 0x3 | 2 | N/A | Always | 18 bits | 256 kB |
| HB8 | 0x1 | 1 | 0x0 | 1 | N/A | Always | 19 bits | 512 kB |
| HB8 | 0x1 | 1 | 0x1 | 4 | N/A | Always | 19 bits | 512 MB |
| HB8 | 0x1 | 1 | 0x2 | 4 | N/A | Always | 18 bits | 256 kB |
| HB8 | 0x2 | 0 | 0x1 | 1 | N/A | Always | 20 bits | 1 MB |
| HB8 | 0x3 | 0 | 0x1 | 1 | N/A | Always | none | - |
| HB8 | 0x3 | 0 | 0x3 | 2 | N/A | Always | none | - |
| HB8 | 0x3 | 1 | 0x0 | 1 | N/A | Always | none | - |
| HB8 | 0x3 | 1 | 0x1 | 4 | N/A | Always | none | - |
| HB8 | 0x3 | 1 | 0x2 | 4 | N/A | Always | none | - |
| HB16 | 0x0 | 0 | 0x0, 0x1 | 1 | 0 | No | 28 bits[a] | 512 MB |
| HB16 | 0x0 | 0 | 0x0, 0x1 | 1 | 1 | Yes | 26 bits[b] | 128 MB |
| HB16 | 0x0 | 0 | 0x2 | 2 | 0 | No | 27 bits[a] | 256 MB |
| HB16 | 0x0 | 0 | 0x2 | 2 | 1 | Yes | 25 bits[b] | 64 MB |
| HB16 | 0x0 | 0 | 0x3 | 2 | 0 | No | 26 bits[a] | 128 MB |
| HB16 | 0x0 | 0 | 0x3 | 2 | 1 | Yes | 24 bits[b] | 32 MB |
| HB16 | 0x0 | 1 | 0x0 | 1 | 0 | No | 27 bits[a] | 256 MB |
| HB16 | 0x0 | 1 | 0x0 | 1 | 1 | Yes | 25 bits[b] | 128 MB |
| HB16 | 0x0 | 1 | 0x1 | 4 | 0 | No | 27 bits[a] | 256 MB |
| HB16 | 0x0 | 1 | 0x1 | 4 | 1 | Yes | 25 bits[b] | 64 MB |
| HB16 | 0x0 | 1 | 0x2 | 4 | 0 | No | 26 bits[a] | 128 MB |
| HB16 | 0x0 | 1 | 0x2 | 4 | 1 | Yes | 24 bits[b] | 32 MB |
| HB16 | 0x1 | 0 | 0x0, 0x1 | 1 | 0 | No | 12 bits[a] | 8 kB |
| HB16 | 0x1 | 0 | 0x0, 0x1 | 1 | 1 | Yes | 10 bits[b] | 2 kB |
| HB16 | 0x1 | 0 | 0x2 | 2 | 0 | No | 11 bits[a] | 4 kB |
| HB16 | 0x1 | 0 | 0x2 | 2 | 1 | Yes | 9 bits[b] | 1 kB |
| HB16 | 0x1 | 0 | 0x3 | 2 | 0 | No | 10 bits[a] | 2 kB |
| HB16 | 0x1 | 0 | 0x3 | 2 | 1 | Yes | 8 bits[b] | 512 B |
| HB16 | 0x1 | 1 | 0x0 | 1 | 0 | No | 11 bits[a] | 4 kB |
| HB16 | 0x1 | 1 | 0x0 | 1 | 1 | Yes | 9 bits[b] | 1 kB |
| HB16 | 0x1 | 1 | 0x1 | 4 | 0 | No | 11 bits[a] | 4 kB |
| HB16 | 0x1 | 1 | 0x1 | 4 | 1 | Yes | 9 bits[b] | 1 kB |
| HB16 | 0x1 | 1 | 0x2 | 4 | 0 | No | 10 bits[a] | 2 kB |
| HB16 | 0x1 | 1 | 0x2 | 4 | 1 | Yes | 8 bits[b] | 512 B |
| HB16 | 0x3 | 0 | 0x1 | 1 | 0 | No | none | - |
| HB16 | 0x3 | 0 | 0x1 | 1 | 1 | Yes | none | - |

**Table 11-7. Capabilities of Host Bus 8 and Host Bus 16 Modes *(continued)***

| Host Bus Type | MODE | CSCFGEXT | CSCFG | Max # of External Devices | BSEL | Byte Access | Available Address | Addressable Memory |
|---|---|---|---|---|---|---|---|---|
| HB16 | 0x3 | 0 | 0x3 | 2 | 0 | No | none | - |
| HB16 | 0x3 | 0 | 0x3 | 2 | 1 | Yes | none | - |
| HB16 | 0x3 | 1 | 0x0 | 1 | 0 | No | none | - |
| HB16 | 0x3 | 1 | 0x0 | 1 | 1 | Yes | none | - |
| HB16 | 0x3 | 1 | 0x1 | 4 | 0 | No | none | - |
| HB16 | 0x3 | 1 | 0x1 | 4 | 1 | Yes | none | - |
| HB16 | 0x3 | 1 | 0x2 | 4 | 0 | No | none | - |
| HB16 | 0x3 | 1 | 0x2 | 4 | 1 | Yes | none | - |

a. If byte selects are not used, data accesses are on 2-byte boundaries. As a result, the available address space is doubled.

b. Two EPI signals are used for byte selects, reducing the available address space by two bits.

Table 11-8 on page 864 shows how the EPI[31:0] signals function while in Host-Bus 8 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the **EPIHB8CFGn** register and on the chip select configuration selected by the CSCFG and CSCFGEXT field in the **EPIHB8CFG2** register.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system. Any unused EPI controller signals can be used as GPIOs or another alternate function.

**Table 11-8. EPI Host-Bus 8 Signal Connections**

| EPI Signal | CSCFG | HB8 Signal (MODE =ADMUX) | HB8 Signal (MODE =ADNOMUX (Cont. Read)) | HB8 Signal (MODE =XFIFO) |
|---|---|---|---|---|
| EPI0S0 | X[a] | AD0 | D0 | D0 |
| EPI0S1 | X | AD1 | D1 | D1 |
| EPI0S2 | X | AD2 | D2 | D2 |
| EPI0S3 | X | AD3 | D3 | D3 |
| EPI0S4 | X | AD4 | D4 | D4 |
| EPI0S5 | X | AD5 | D5 | D5 |
| EPI0S6 | X | AD6 | D6 | D6 |
| EPI0S7 | X | AD7 | D7 | D7 |
| EPI0S8 | X | A8 | A0 | - |
| EPI0S9 | X | A9 | A1 | - |
| EPI0S10 | X | A10 | A2 | - |
| EPI0S11 | X | A11 | A3 | - |
| EPI0S12 | X | A12 | A4 | - |
| EPI0S13 | X | A13 | A5 | - |
| EPI0S14 | X | A14 | A6 | - |
| EPI0S15 | X | A15 | A7 | - |
| EPI0S16 | X | A16 | A8 | - |
| EPI0S17 | X | A17 | A9 | - |
| EPI0S18 | X | A18 | A10 | - |

**Table 11-8. EPI Host-Bus 8 Signal Connections *(continued)***

| EPI Signal | CSCFG | HB8 Signal (MODE =ADMUX) | HB8 Signal (MODE =ADNOMUX (Cont. Read)) | HB8 Signal (MODE =XFIFO) |
|---|---|---|---|---|
| EPI0S19 | X | A19 | A11 | - |
| EPI0S20 | X | A20 | A12 | - |
| EPI0S21 | X | A21 | A13 | - |
| EPI0S22 | X | A22 | A14 | - |
| EPI0S23 | X | A23 | A15 | - |
| EPI0S24 | X | A24 | A16 | - |
| EPI0S25 | 0x0 | A25[b] | A17 | - |
| | 0x1 | | | |
| | 0x2 | | | CS1n |
| | 0x3 | | | - |
| | 0x4 | | | - |
| | 0x5 | | | - |
| | 0x6 | | | - |
| EPI0S26 | 0x0 | A26 | A18 | FEMPTY |
| | 0x1 | | | |
| | 0x2 | | | |
| | 0x3 | CS0n | CS0n | |
| | 0x4 | A26 | A18 | |
| | 0x5 | | | |
| | 0x6 | CS0n | CS0n | |
| EPI0S27 | 0x0 | A27 | A19 | FFULL |
| | 0x1 | | | |
| | 0x2 | CS1n | CS1n | |
| | 0x3 | | | |
| | 0x4 | CS0n | CS0n | |
| | 0x5 | CS1n | CS1n | |
| | 0x6 | | | |
| EPI0S28 | X | RDn/OEn | RDn/OEn | RDn |
| EPI0S29 | X | WRn | WRn | WRn |
| EPI0S30 | 0x0 | ALE | ALE | - |
| | 0x1 | CSn | CSn | CSn |
| | 0x2 | CS0n | CS0n | CS0n |
| | 0x3 | ALE | ALE | - |
| | 0x4 | | | - |
| | 0x5 | CS0n | CS0n | - |
| | 0x6 | ALE | ALE | - |
| EPI0S31 | X | Clock[c] | Clock[c] | Clock[c] |
| EPI0S32 | X | iRDY | iRDY | iRDY |

**Table 11-8. EPI Host-Bus 8 Signal Connections *(continued)***

| EPI Signal | CSCFG | HB8 Signal (MODE =ADMUX) | HB8 Signal (MODE =ADNOMUX (Cont. Read)) | HB8 Signal (MODE =XFIFO) |
|---|---|---|---|---|
| EPI0S33 | 0x0 | X | X | X |
| | 0x1 | X | X | X |
| | 0x2 | X | X | X |
| | 0x3 | X | X | X |
| | 0x4 | X | X | X |
| | 0x5 | CS3n | CS3n | X |
| | 0x6 | | | X |
| EPI0S34 | 0x0 | X | X | X |
| | 0x1 | X | X | X |
| | 0x2 | X | X | X |
| | 0x3 | X | X | X |
| | 0x4 | X | X | X |
| | 0x5 | CS2n | CS2n | X |
| | 0x6 | | | X |
| EPI0S35 | 0x0 | X | X | X |
| | 0x1 | X | X | X |
| | 0x2 | X | X | X |
| | 0x3 | X | X | X |
| | 0x4 | X | X | X |
| | 0x5 | CRE | CRE | X |
| | 0x6 | | | X |

a. "X" indicates the state of this field is a don't care.

b. When an entry straddles several row, the signal configuration is the same for all rows.

c. The clock signal is not required for this mode.

Table 11-9 on page 866 shows how the EPI[31:0] signals function while in Host-Bus 16 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the **EPIHB16CFGn** register, on the chip select configuration selected by the CSCFG and CSCFGEXT field in the same register, and on whether byte selects are used as configured by the BSEL bit in the **EPIHB16CFG** register.

Although the EPI0S31 signal can be configured for the EPI clock signal in Host-Bus mode, it is not required and should be configured as a GPIO to reduce EMI in the system. Any unused EPI controller signals can be used as GPIOs or another alternate function.

**Table 11-9. EPI Host-Bus 16 Signal Connections**

| EPI Signal | CSCFG | BSEL | HB16 Signal (MODE =ADMUX) | HB16 Signal (MODE =ADNOMUX (Cont. Read)) | HB16 Signal (MODE =XFIFO) |
|---|---|---|---|---|---|
| EPI0S0 | X[a] | X | AD0[b] | D0 | D0 |
| EPI0S1 | X | X | AD1 | D1 | D1 |
| EPI0S2 | X | X | AD2 | D2 | D2 |
| EPI0S3 | X | X | AD3 | D3 | D3 |

**Table 11-9. EPI Host-Bus 16 Signal Connections *(continued)***

| EPI Signal | CSCFG | BSEL | HB16 Signal (MODE =ADMUX) | HB16 Signal (MODE =ADNOMUX (Cont. Read)) | HB16 Signal (MODE =XFIFO) |
|---|---|---|---|---|---|
| EPI0S4 | X | X | AD4 | D4 | D4 |
| EPI0S5 | X | X | AD5 | D5 | D5 |
| EPI0S6 | X | X | AD6 | D6 | D6 |
| EPI0S7 | X | X | AD7 | D7 | D7 |
| EPI0S8 | X | X | AD8 | D8 | D8 |
| EPI0S9 | X | X | AD9 | D9 | D9 |
| EPI0S10 | X | X | AD10 | D10 | D10 |
| EPI0S11 | X | X | AD11 | D11 | D11 |
| EPI0S12 | X | X | AD12 | D12 | D12 |
| EPI0S13 | X | X | AD13 | D13 | D13 |
| EPI0S14 | X | X | AD14 | D14 | D14 |
| EPI0S15 | X | X | AD15 | D15 | D15 |
| EPI0S16 | X | X | A16 | A0[b] | - |
| EPI0S17 | X | X | A17 | A1 | - |
| EPI0S18 | X | X | A18 | A2 | - |
| EPI0S19 | X | X | A19 | A3 | - |
| EPI0S20 | X | X | A20 | A4 | - |
| EPI0S21 | X | X | A21 | A5 | - |
| EPI0S22 | X | X | A22 | A6 | - |
| EPI0S23 | X[c] | 0 | A23 | A7 | - |
| | | 1 | | | |
| EPI0S24 | 0x0 | 0 | A24 | A8 | - |
| | | 1 | | | |
| | 0x1 | 0 | | | |
| | | 1 | | | |
| | 0x2 | 0 | | | |
| | | 1 | | | |
| | 0x3 | 0 | | | |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x4 | 0 | A24 | A8 | - |
| | | 1 | | | |
| | 0x5 | 0 | | | - |
| | | 1 | | | |
| | 0x6 | 0 | | | - |
| | | 1 | BSEL0n | BSEL0n | |

**Table 11-9. EPI Host-Bus 16 Signal Connections *(continued)***

| EPI Signal | CSCFG | BSEL | HB16 Signal (MODE =ADMUX) | HB16 Signal (MODE =ADNOMUX (Cont. Read)) | HB16 Signal (MODE =XFIFO) |
|---|---|---|---|---|---|
| EPI0S25 | 0x0 | X | A25 | A9 | - |
| | 0x1 | | | | |
| | 0x2 | 0 | A25 | A9 | CS1n |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x3 | 0 | A25 | A9 | -- |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x4 | 0 | A25 | A9 | - |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x5 | 0 | A25 | A9 | - |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x6 | 0 | A25 | A9 | - |
| | | 1 | BSEL1n | BSEL1n | |
| EPI0S26 | 0x0 | 0 | A26 | A10 | FEMPTY |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x1 | 0 | A26 | A10 | |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x2 | 0 | A26 | A10 | |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x3 | X | CS0n | CS0n | |
| | 0x4 | 0 | A26 | A10 | - |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x5 | 0 | A26 | A10 | - |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x6 | 0 | CS0n | CS0n | - |
| | | 1 | | | |
| EPI0S27 | 0x0 | 0 | A27 | A11 | FFULL |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x1 | 0 | A27 | A11 | |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x2 | X | CS1n | CS1n | |
| | 0x3 | X | CS1n | CS1n | |
| | 0x4 | X | CS0n | CS0n | - |
| | 0x5 | X | CS1n | CS1n | - |
| | 0x6 | X | CS1n | CS1n | - |
| EPI0S28 | X | X | RDn/OEn | RDn/OEn | RDn |
| EPI0S29 | X | X | WRn | WRn | WRn |

**Table 11-9. EPI Host-Bus 16 Signal Connections *(continued)***

| EPI Signal | CSCFG | BSEL | HB16 Signal (MODE =ADMUX) | HB16 Signal (MODE =ADNOMUX (Cont. Read)) | HB16 Signal (MODE =XFIFO) |
|---|---|---|---|---|---|
| EPI0S30 | 0x0 | X | ALE | ALE | - |
| | 0x1 | X | CSn | CSn | CSn |
| | 0x2 | X | CS0n | CS0n | CS0n |
| | 0x3 | X | ALE | ALE | - |
| | 0x4 | X | ALE | ALE | - |
| | 0x5 | X | CS0n | CS0n | - |
| | 0x6 | X | ALE | ALE | - |
| EPI0S31 | X | X | Clock[d] | Clock[d] | Clock[d] |
| EPI0S32 | X | X | iRDY | iRDY | iRDY |
| EPI0S33 | X | X | CS3n | CS3n | X |
| EPI0S34 | X | X | CS2n | CS2n | X |
| EPI0S35 | X | X | CRE | CRE | X |

a. "X" indicates the state of this field is a don't care.

b. In this mode, half-word accesses are used. A0 is the LSB of the address and is equivalent to the internal Cortex-M3 A1 address. This pin should be connected to A0 of 16-bit memories.

c. When an entry straddles several row, the signal configuration is the same for all rows.

d. The clock signal is not required for this mode.

The RDYEN in the **EPIHBnCFG** enables the monitoring of the external iRDY pin to stall accesses. On the rising edge of EPI clock, if iRDY is low, access is stalled. The IRDYDLY can program the number of EPI clock cycles in advance to the stall (1,2 or 3) as shown in Figure 11-5 on page 870. This is a conceptual timing diagram of how the iRDY signal works with different IRDYDLY configurations. When enabled, the iRDY stalls the EPI's internal states, while IRDYDLY controls the delay pipeline when this stall takes affect. The iRDY signal can be connected to multiple devices with a pull up resistor as shown in Figure 11-6 on page 870. Note that when multiple PSRAMs are connected to iRDY, the **EPIHPnCFG** registers must be programmed to the same iRDY signal polarity through the IRDYINV bit. When connected to a PSRAM, iRDY is used to control the address to data latency.

**Figure 11-5. iRDY Access Stalls, IRDYDLY==01, 10, 11**

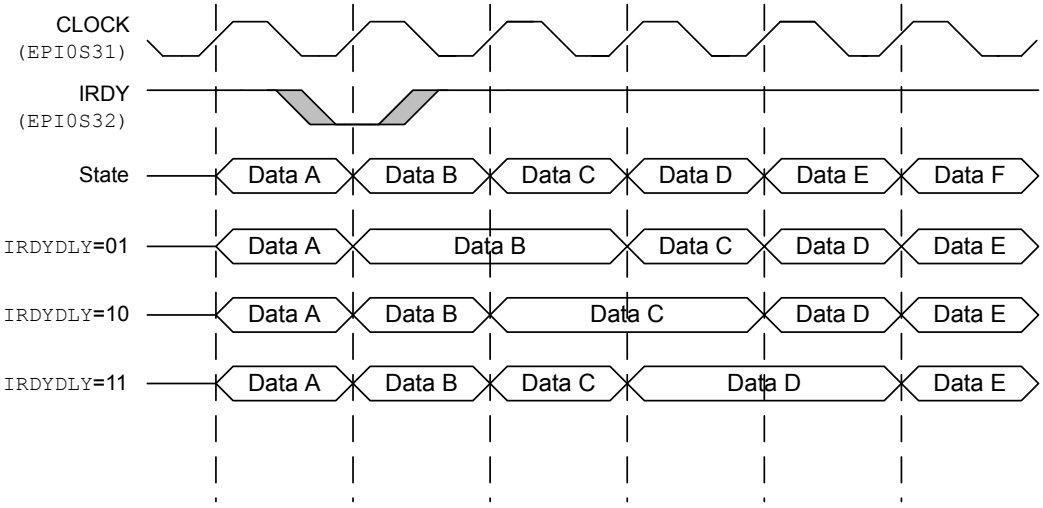
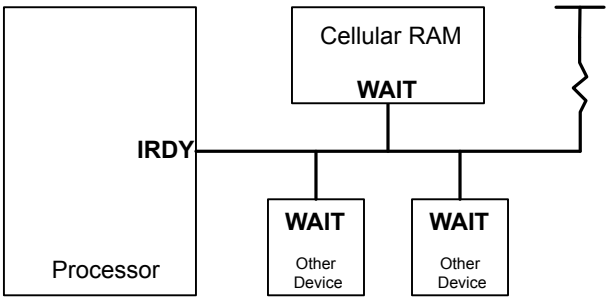



**Figure 11-6. iRDY Signal Connection**




### 11.4.3.2 PSRAM Support

The EPI Host Bus supports both a synchronous and asynchronous interface to PSRAM memory when configured in 16-bit bus multiplexed mode. The **EPIHBPSRAM** register holds the values for the PSRAM's bus configuration registers (CR). The contents of the **EPIHBPSRAM** register can be sent to different memories depending on which WRCRE or RDCRE bit is set in the various **EPIHB16CFGn** registers. For example, if the WRCRE bit is enabled in **EPIHB16CFG**, then the CRE signal asserts and the contents are sent to the memory enabled by CS0. Enabling the WRCRE or RDCRE bit in **EPIHB16CFG2** register activates CS1n during a PSRAM configuration register write or read. The WRCRE and RDCRE bit in **EBIHB16CFG3** corresponds to CS2n and **EPIHB16CFG4**, to CS3n. The WRCRE bit clears when the transfer is done. There must not be any system access or non-blocking read activity during the CRE read or write-enable transfer. During a write to the PSRAM's CR, the configuration data is written out on data pins [20:0] of the EPI bus. For a PSRAM configuration read access, the RDCRE bit in the **EPIHB16CFG** register is set to signal that the next access is a read of the PSRAM configuration register (CR). The address for the CR is written to bits CR[19:18] of the **EPIHBPSRAM** register. The read data is returned at CR bits [15:0] of the **EPIHBPSRAM** register.

**Note:** ■ CRE read and write operations may only occur in asynchronous mode. During synchronous mode the CRE bit should be disabled. Setting the CRE bit during synchronous PSRAM accesses can lead to unpredictable behavior.

- When the chip select is programmed to access the PSRAM, the MODE bit of the **EPIHBnCFGn** register must be programmed to enable address and data muxed (ADMUX). Page mode accesses are not supported by the EPI.
- BURST is optimized for word-length bursting for SDRAM and PSRAM accesses.

The subsequent list identifies the steps for initializing the PSRAM interface:

1. Follow the EPI initialization steps in “Initialization and Configuration” on page 854.
2. Enable Host Bus 16 Mode by setting the MODE bits in the **EPICFG** register to 0x13. Choose between an integer or formula clock divide for the baud rate by configuring the INTDIV bit in the **EPICFG** register.
3. Configure the **EPIBAUD** register to the desired baud rate.
4. Since the EPI module only supports asynchronous programming of the configuration registers, clock gate the EPI clock by programming both the CLKGATE and CLKGATEI bits in the **EPIHB16CFG** register to 0.
5. Prepare for writing the PSRAM's Bus Configuration Register by setting the ALEHIGH = 1 and MODE=0x0 in the **EPIHB16CFG** register.
6. Program the **EPIHBPSRAM** register to be loaded into the CR register of the PSRAM by configuring bits [21:0].
   - CR[20:19] =0x0, reserved
   - CR[19:18] = 0x2 to enable configuring of the CR register
   - CR[15]= 0x1 to enable asynchronous access
   - CR [14] = 0 if the iRDY signal is used for memory transfers; if the design will not use the iRDY signal CR[14] should be cleared.
   - CR[13:11] must be programmed to have a matching read and write wait state configuration as is programmed in the **EPIHB16CFG** and **EPIHB16TIME** register.
   - CR[10] configures the polarity of the WAIT signal and should match the configuration of the IRDYINV bit in the **EPIHB16CFG** register.
   - CR[8]=0x1 to configure the appropriate wait configuration of the data
   - CR[2:0]=0x7 since the EPI interface in PSRAM mode is a continuous burst access.
7. Set the WRCRE bit in the **EPIHB16CFGn** register to initiate a write from the **EPIHBPSRAM** register to PSRAM's CR register.

**Note:** If the PSRAM's CR register must be reprogrammed after initialization, the application should allow the previous transfer to complete before beginning configuration to ensure proper PSRAM functionality.

**Table 11-10. PSRAM Fixed Latency Wait State Configuration**

| Latency Counter | Latency in Clocks | RDWS[1:0]/WRWS[1:0] | RWSM/WRWSM |
|---|---|---|---|
| BCR Code 2 | 3 | 0x0 | 0 |
| BCR Code 3 | 4 | 0x1 | 1 |
| BCR Code 4 | 5 | 0x1 | 0 |
| BCR Code 5 | 6 | 0x2 | 1 |
| BCR Code 6 | 7 | 0x2 | 0 |
| BCR Code 8 | 9 | 0x3 | 0 |

In variable initial latency mode, the memory's WAIT (iRDY) pin guides the EPI module when to read and write. The WAIT (iRDY) pin stalls the access for the duration of the latency and adds cycles if there is a refresh collision. To get the best performance, set CR[13:11] = 0x2, the WRWS field of the **EPIHB16CFG** register to 0x0, and the WRWSM and RDWSM bit of the **EPI16TIMEn** register to 0. For the WAIT pin to be recognized correctly set the IRDYDLY bit in the **EPI16TIMEn** register to 1 and the CR[8] =1 in the **EPIHBPSRAM** register.

**Note:** Wait state latency works differently in PSRAM Burst mode than in other modes. In PSRAM Burst mode the RDWS and WRWS bit fields define the latency for only the first access of the write or read cycle. Every access after that is a single access.

Figure 11-7 on page 872 and Figure 11-8 on page 873 depict a PSRAM burst read and write.

**Figure 11-7. PSRAM Burst Read**

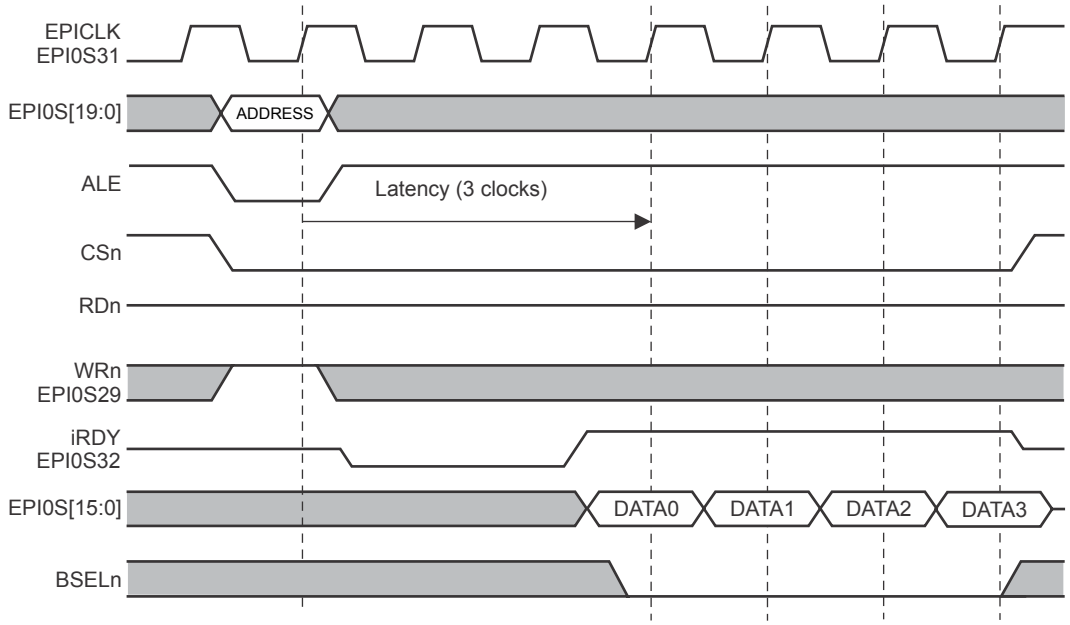
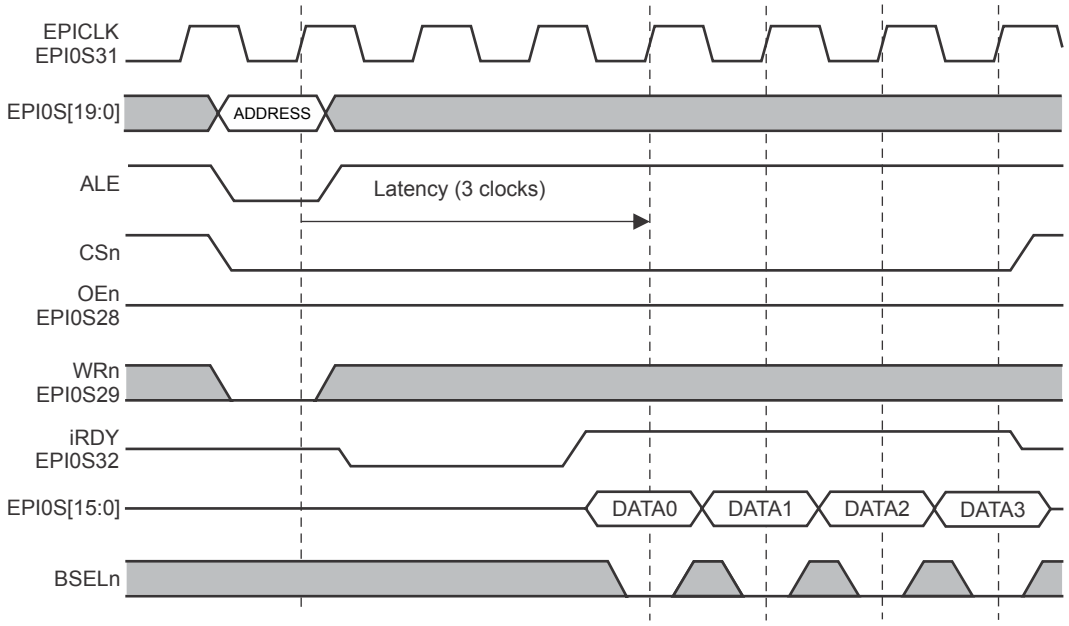


**Figure 11-8. PSRAM Burst Write**




Note that if a read or write transfer attempts to begin during a refresh event, the transfer is held off by the assertion of the iRDY pin by the memory to the EPI module. Figure 11-9 on page 874 and Figure 11-10 on page 874 depict the delay in data transfer during a refresh collision.

**Figure 11-9. Read Delay During Refresh Event**

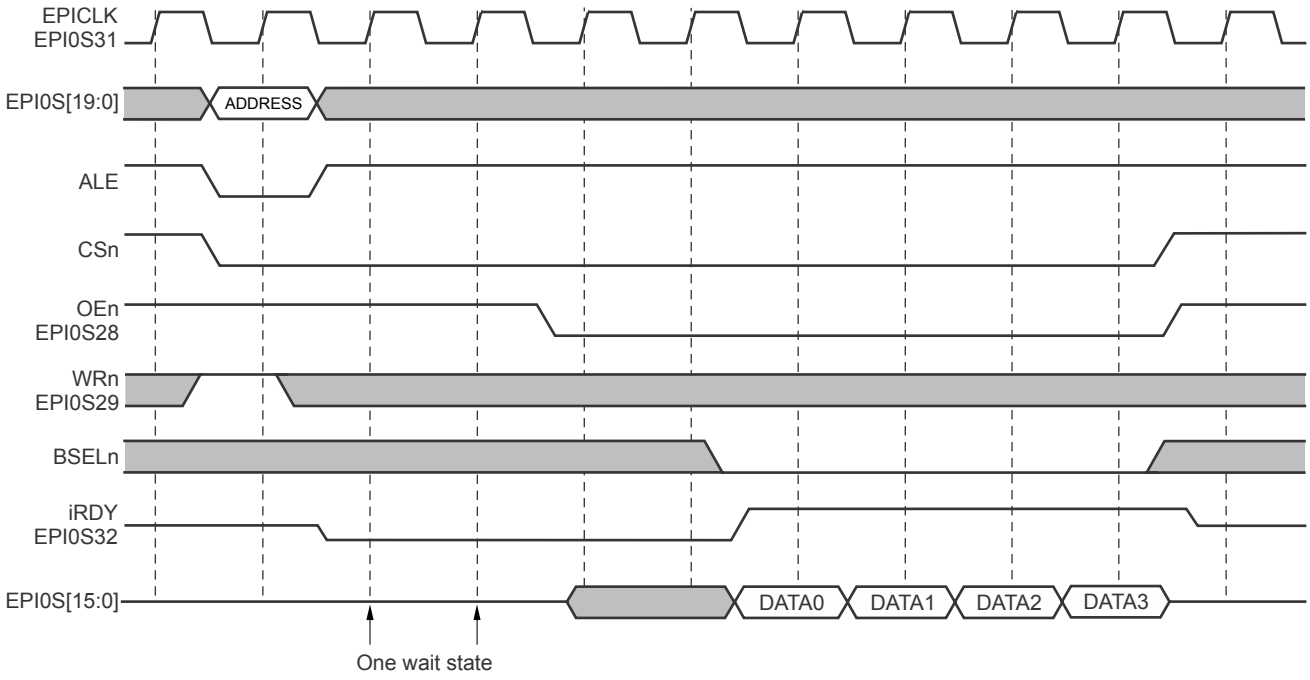
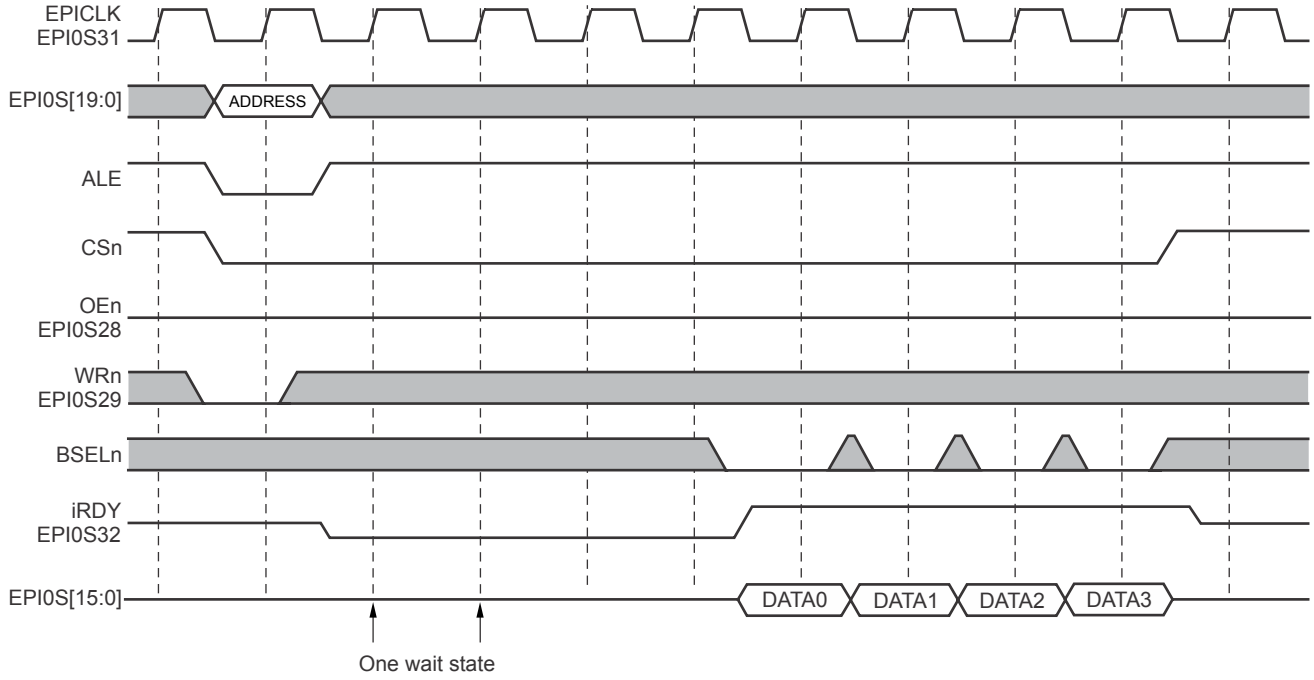


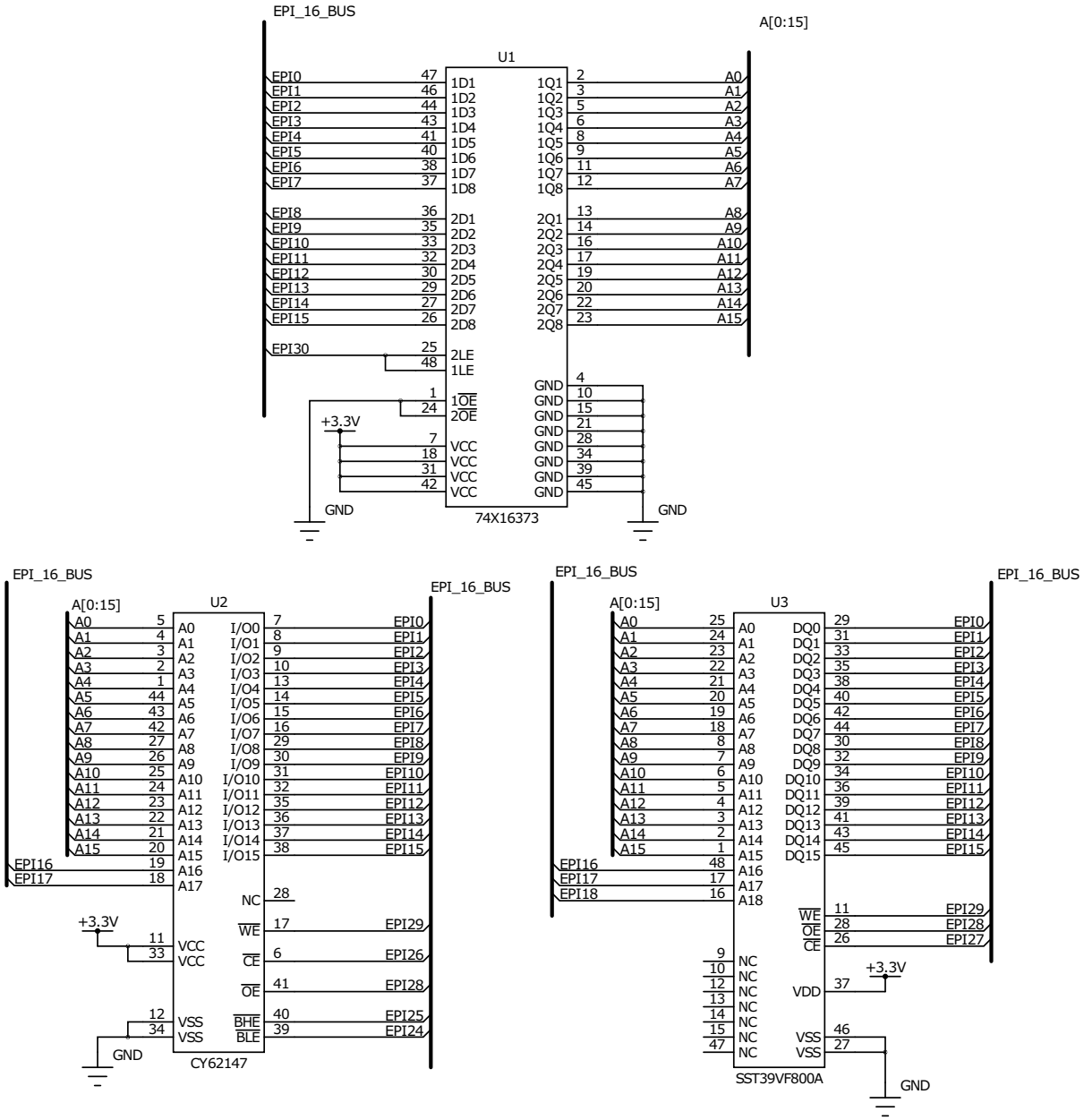

**Figure 11-10. Write Delay During Refresh Event**



### 11.4.3.3 Host Bus 16-bit Muxed Interface

Figure 11-11 on page 875 shows how to connect the EPI signals to a 16-bit SRAM and a 16-bit Flash memory with muxed address and memory using byte selects and dual chip selects with ALE. This schematic is just an example of how to connect the signals; timing and loading have not been analyzed. In addition, not all bypass capacitors are shown.

**Figure 11-11. Example Schematic for Muxed Host-Bus 16 Mode**



#### 11.4.3.4 Speed of Transactions

The COUNT0 field in the **EPIBAUD** register must be configured to set the main transaction rate based on what the slave device can support (including wiring considerations). The main control transitions are normally ½ the baud rate (COUNT0 = 1) because the EPI block forces data versus control to change on alternating clocks. When using dual chip selects, each chip select can access the bus using differing baud rates by setting the CSBAUD bit in the **EPIHBnCFG2** register. In this case, the COUNT0 field controls the CS0n transactions, and the COUNT1 field controls the CS1n transactions. When using quad chip select mode, the COUNT0 bit field of the **EPIBAUD2** register controls the baud rate of CS2n and the COUNT1 bit field is programmed to control the baud rate of CS3n.

Additionally, the Host-Bus mode provides read and write wait states for the data portion to support different classes of device. These wait states stretch the data period (hold the rising edge of data strobe) and may be used in all four sub-modes. The wait states are set using the WRWS and RDWS bits in the **EPI Host-Bus n Configuration (EPIHBnCFGn)** register. The WRWS and RDWS bits are enhanced with more precision by WRWSM and RDWSM bits in the **EPIHBnTIMEn** registers. Note none of the wait state configuration bits can be set concurrently with the BURST bit in the same **EPIHBnCFGn** register. See Table 11-11 on page 876 for programming information.

**Table 11-11. Data Phase Wait State Programming**

| RDWS or WRWS Encoding in EPIHBnCFGn Register | RDWSM or WRWSM Encoding in EPIHBnTIMEn Registers | Data Phase Wait States |
|---|---|---|
| 0x0 | 1 | 1 EPI clocks |
| 0x0 | 0 | 2 EPI clocks |
| 0x1 | 1 | 3 EPI clocks |
| 0x1 | 0 | 4 EPI clocks |
| 0x2 | 1 | 5 EPI clocks |
| 0x2 | 0 | 6 EPI clocks |
| 0x3 | 1 | 7 EPI clocks |
| 0x3 | 0 | 8 EPI clocks |

The CAPWIDTH bit in **EPIHBnTIMEn** registers controls the delay between Host-Bus transfers. When the CSBAUD bit is set and multi-chip selects have been configured in the **EPIHBnCFG2** registers, delay takes an additional clock cycle to adjust the clock rate of different chip selects.

Word read and write transactions can be enhanced through the enabling of the BURST bit in the **EPIHB16CFGn** registers.

#### 11.4.3.5 Sub-Modes of Host Bus 8/16

The EPI controller supports four variants of the Host-Bus model using 8 or 16 bits of data in all four cases. The four sub-modes are selected using the MODE bits in the **EPIHBnCFG** register, and are:

1. Address and data are muxed. This scheme is used by many 8051 devices, some Microchip PIC parts, and some ATmega parts. When used for standard SRAMs, a latch must be used between the microcontroller and the SRAM. This sub-mode is provided for compatibility with existing devices that support data transfers without a latch (that is, CPLDs). In general, the de-muxed sub-mode should normally be used. The ALE configuration should be used in this mode, as all Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold until the data phase. The ALE configuration is controlled by configuring the CSCFG and CSCFGEXT field to be 0x0 in the **EPIHBnCFG2** register.

The ALE can be enhanced to access two or four external devices with four separate CSn signals. By configuring the CSCFG field to be 0x3 and the CSCFGEXT bit to be 0 in the **EPIHBnCFG2** register, EPI0S30 functions as ALE, EPI0S27 functions as CS1n, and EPI0S26 functions as CS0n. When the CSCFG field is set to 0x0 and the CSCFGEXT bit is set to 1 in the **EPIHBnCFG2** register, EPI0S30 functions as ALE, EPIOS33 functions as CS3n, EPIOS34 functions as CS2n, EPI0S27 functions as CS1n, and EPI0S26 functions as CS0n. The CSn is best used for Host-Bus unmuxed mode, in which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access are occurring.

2. Address and data are separate with 8 or 16 bits of data and up to 20 bits of address (1 MB). This scheme is used by more modern 8051 devices, as well as some PIC and ATmega parts. This mode is generally used with SRAMs in continuous read modes, many EEPROMs, and many NOR Flash memory devices. Note that there is no hardware command write support for Flash memory devices; this mode should only be used for Flash memory devices programmed at manufacturing time. If a Flash memory device must be written and does not support a direct programming model, the command mechanism must be performed in software. The CSn configuration should be used in this mode. The CSn signal indicates when the address and data phases of a read or write access is occurring. The CSn configuration is controlled by configuring the CSCFG field to be 0x1 and the CSCFGEXT bit to be 0 in the **EPIHBnCFG2** register.

3. Continuous read mode where address and data are separate. This read sub-mode is used by some SRAMs and can read more quickly by only changing the address (and not using RDn/OEn strobing). In this sub-mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change. For example, to read data from address 0x100 and then 0x101, the EPI controller asserts the output-enable signal and then configures the address pins to 0x100; the EPI controller then captures what is on the data pins and increments A0 to 1 (so the address is now 0x101); the EPI controller then captures what is on the data pins. Note that this mode consumes higher power because the SRAM must continuously drive the data pins. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available. Writes are not permitted in this mode.

4. FIFO mode uses 8 or 16 bits of data, removes ALE and address pins and optionally adds external XFIFO FULL/EMPTY flag inputs. This scheme is used by many devices, such as radios, communication devices (including USB2 devices), and some FPGA configurations (FIFO through block RAM). This sub-mode provides the data side of the normal Host-Bus interface, but is paced by the FIFO control signals. It is important to consider that the XFIFO FULL/EMPTY control signals may stall the interface and could have an impact on blocking read latency from the processor or µDMA. Note that the EPI FIFO can only be used in asynchronous mode.

For the three modes above (1, 2, 4) that the Host-Bus 16 mode supports, byte select signals can be optionally implemented by setting the BSEL bit in the **EPIHB16CFG** register.

**Note:** Byte accesses should not be attempted if the BSEL bit has not been enabled in Host-Bus 16 Mode.

See “External Peripheral Interface (EPI)” on page 2130 for timing details for the Host-Bus mode.

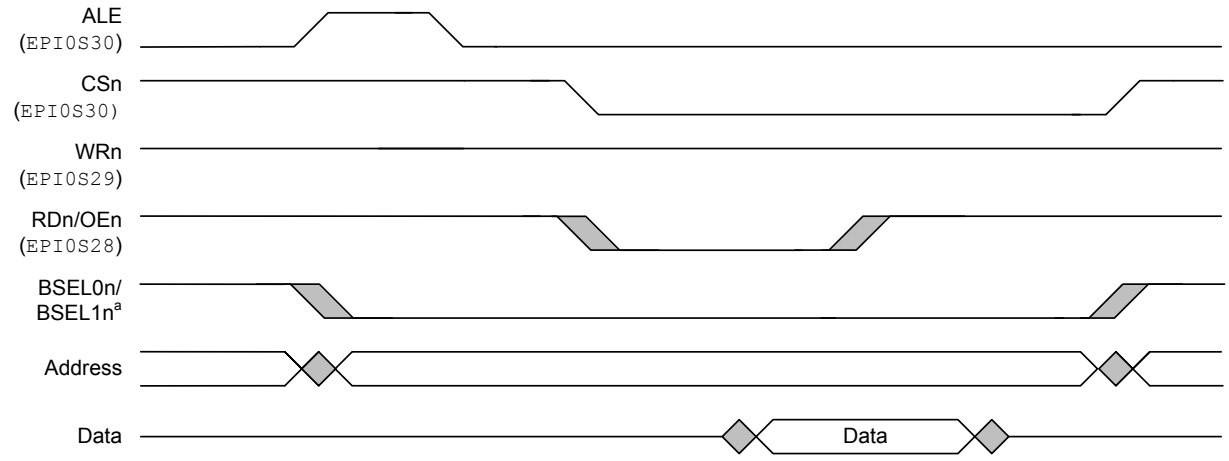
#### 11.4.3.6 Bus Operation

Bus operation is the same in Host-Bus 8 and Host-Bus 16 modes and is asynchronous. Timing diagrams show both ALE and CSn operation. The optional HB16 byte select signals have the same timing as the address signals. If wait states are required in the bus access, they can be inserted during the data phase of the access using the WRWS and RDWS bits in the **EPIHBnCFG2** register.

Each wait state adds 2 EPI clock cycles to the duration of the WRn or RDn strobe. During idle cycles, the address and muxed address data signals maintain the state of the last cycle.

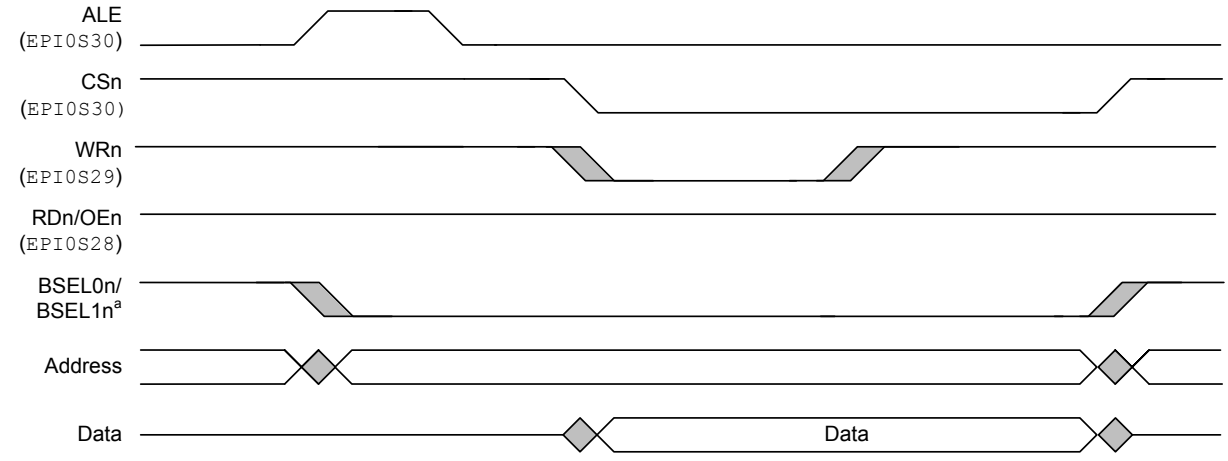
Figure 11-12 on page 878 shows a basic Host-Bus read cycle. Figure 11-13 on page 878 shows a basic Host-Bus write cycle. Both of these figures show address and data signals in the non-multiplexed mode (MODE field ix 0x1 in the **EPIHBnCFG** register).

**Figure 11-12. Host-Bus Read Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0**




[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

**Figure 11-13. Host-Bus Write Cycle, MODE = 0x1, WRHIGH = 0, RDHIGH = 0**




[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 11-14 on page 879 shows a write cycle with the address and data signals multiplexed (MODE field is 0x0 in the **EPIHBnCFG** register). A read cycle would look similar, with the RDn strobe being asserted along with CSn and data being latched on the rising edge of RDn.

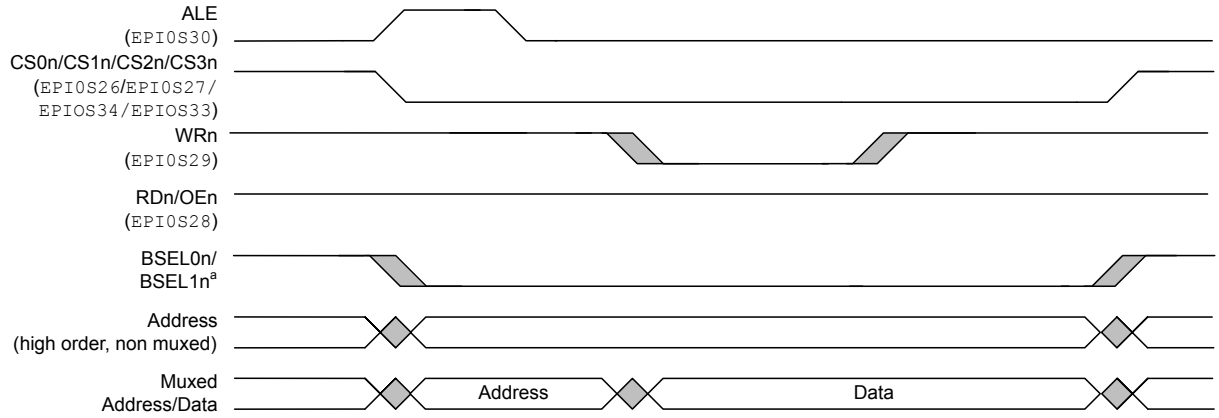
**Figure 11-14. Host-Bus Write Cycle with Multiplexed Address and Data, MODE = 0x0, WRHIGH = 0, RDHIGH = 0**




[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

When using ALE with dual CSn configuration (CSCFGEXT bit is 0 and the CSCFG field is 0x3 in the **EPIHBnCFG2** register) or quad chip select (CSCFGEXT bit is 1 and CSCSFG is 0x2), the appropriate CSn signal is asserted at the same time as ALE, as shown in Figure 11-15 on page 879.

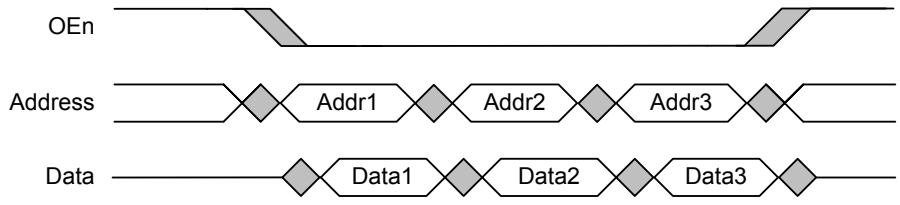
**Figure 11-15. Host-Bus Write Cycle with Multiplexed Address and Data and ALE with Dual or Quad CSn**




[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

Figure 11-16 on page 879 shows continuous read mode accesses. In this mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change.

**Figure 11-16. Continuous Read Mode Accesses**



FIFO mode accesses are the same as normal read and write accesses, except that the ALE signal and address pins are not present. Two input signals can be used to indicate when the XFIFO is full or empty to gate transactions and avoid overruns and underruns. The FFULL and FEMPTY signals are synchronized and must be recognized as asserted by the microcontroller for 2 system clocks before they affect transaction status. The MAXWAIT field in the **EPIHBnCFG** register defines the maximum number of EPI clocks to wait while the FEMPTY or FFULL signal is holding off a transaction. Figure 11-17 on page 880 shows how the FEMPTY signal should respond to a write and read from the XFIFO. Figure 11-18 on page 880 shows how the FEMPTY and FFULL signals should respond to 2 writes and 1 read from an external FIFO that contains two entries.

**Figure 11-17. Write Followed by Read to External FIFO**

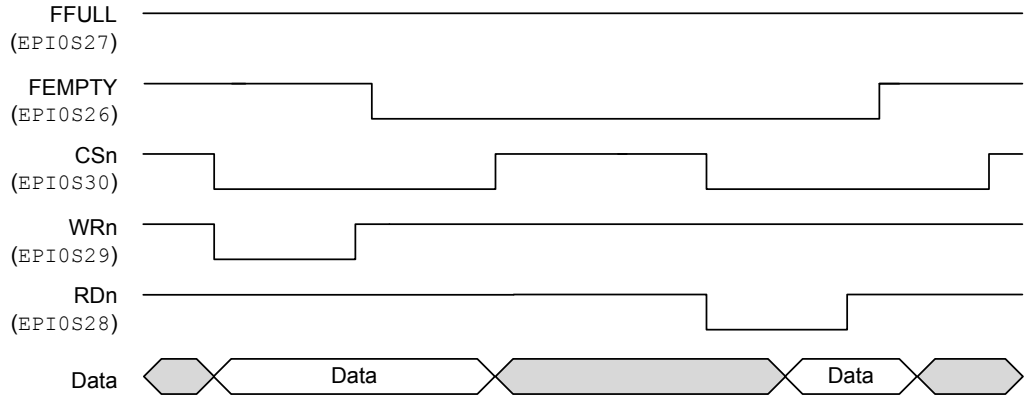
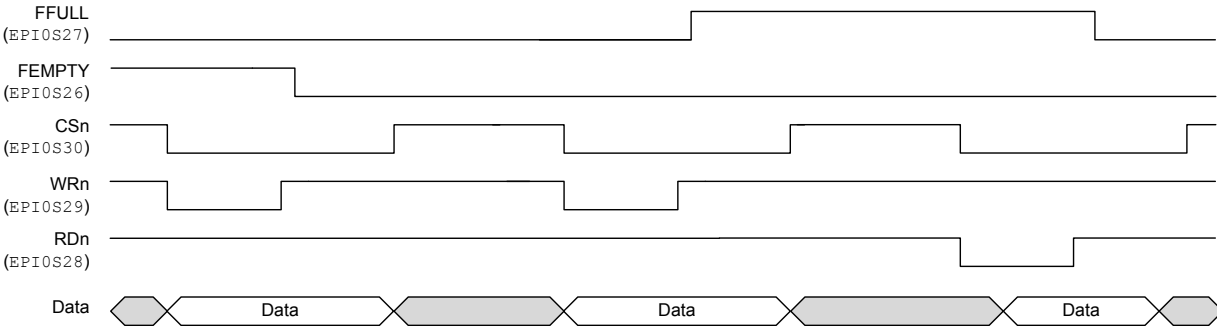



**Figure 11-18. Two-Entry FIFO**




## 11.4.4 General-Purpose Mode

The **General-Purpose Mode Configuration (EPIGPCFG)** register is used to configure the control, data, and address pins, if used. Any unused EPI controller signals can be used as GPIOs or another alternate function. The general-purpose configuration can be used for custom interfaces with FPGAs, CPLDs, and digital data acquisition and actuator control.

General-Purpose mode is designed for three general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs. Three sizes of data and optional address are supported. Framing and clock-enable functions permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the **EPIBAUD** register baud rate (when used with the WFIFO and/or the NBRFIFO) or by the rate of accesses from software or µDMA. Examples of this type of use include:

  - Reading 20 sensors at fixed time periods by configuring 20 pins to be inputs, configuring the COUNT0 field in the **EPIBAUD** register to some divider, and then using non-blocking reads.
  - Implementing a very wide ganged PWM/PCM with fixed frequency for driving actuators, LEDs, etc.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free-running or gated), a framing signal (with frame size), a ready input (to stretch transactions), an address (of varying sizes), and data (of varying sizes). Additionally, provisions are made for separating data and address phases.

The interface has the following optional features:

- Use of the EPI clock output is controlled by the CLKPIN bit in the **EPIGPCFG** register. Unclocked uses include general-purpose I/O and asynchronous interfaces (optionally using RD and WR strobes). Clocked interfaces allow for higher speeds and are much easier to connect to FPGAs and CPLDs (which usually include input clocks).
- EPI clock, if used, may be free running or gated depending on the CLKGATE bit in the **EPIGPCFG** register. A free-running EPI clock requires another method for determining when data is live, such as the frame pin or RD/WR strobes. A gated clock approach uses a setup-time model in which the EPI clock controls when transactions are starting and stopping. The gated clock is held high until a new transaction is started and goes high at the end of the cycle where RD/WR/FRAME and address (and data if write) are emitted.
- Use of the RD and WR outputs is controlled by the RW bit in the **EPIGPCFG** register. For interfaces where the direction is known (in advance, related to frame size, or other means), these strobes are not needed. For most other interfaces, RD and WR are used so the external peripheral knows what transaction is taking place, and if any transaction is taking place.
- Separation of address/request and data phases may be used on writes using the WR2CYC bit in the **EPIGPCFG** register. This configuration allows the external peripheral extra time to act. Address and data phases must be separated on reads. When configured to use an address as specified by the ASIZE field in the **EPIGPCFG** register, the address is emitted on the with the RD strobe (first cycle) and data is expected to be returned on the next cycle (when RD is not asserted). If no address is used, then RD is asserted on the first cycle and data is captured on the second cycle (when RD is not asserted), allowing more setup time for data.

  **Note:** When WR2CYC = 0, write data is valid when the WR strobe is asserted (High). When WR2CYC = 1, write data is valid when the WR strobe is Low after being asserted (High).

  For writes, the output may be in one or two cycles. In the two-cycle case, the address (if any) is emitted on the first cycle with the WR strobe and the data is emitted on the second cycle (with WR not asserted). Although split address and write data phases are not normally needed for logic reasons, it may be useful to make read and write timings match. If 2-cycle reads or writes are used, the RW bit is automatically set.
- Address may be emitted (controlled by the ASIZE field in the **EPIGPCFG** register). The address may be up to 4 bits (16 possible values), up to 12 bits (4096 possible values), or up to 20 bits (1 M possible values). Size of address limits size of data, for example, 4 bits of address support up to 24 bits data. 4-bit address uses EPI0S[27:24]; 12-bit address uses EPI0S[27:16]; 20-bit address uses EPI0S[27:8]. The address signals may be used by the external peripheral as an address, code (command), or for other unrelated uses (such as a chip enable). If the chosen address/data combination does not use all of the EPI signals, the unused pins can be

used as GPIOs or for other functions. For example, when using a 4-bit address with an 8-bit data, the pins assigned to EPIS0[23:8] can be assigned to other functions.

- Data may be 8 bits, 16 bits, 24 bits, or 32 bits (controlled by the DSIZE field in the **EPIGPCFG** register). By default, the EPI controller uses data bits [7:0] when the DSIZE field in the **EPIGPCFG** register is 0x0; data bits [15:0] when the DSIZE field is 0x1; data bits [23:0] when the DSIZE field is 0x2; and data bits [31:0] when the DSIZE field is 0x3.32-bit data cannot be used with address or EPI clock or any other signal. 24-bit data can only be used with 4-bit address or no address.
- When using the EPI controller as a GPIO interface, writes are FIFOed (up to 4 can be held at any time), and up to 32 pins are changed using the **EPIBAUD** clock rate specified by COUNT0. As a result, output pin control can be very precisely controlled as a function of time. By contrast, when writing to normal GPIOs, writes can only occur 8-bits at a time and take up to two clock cycles to complete. In addition, the write itself may be further delayed by the bus due to µDMA or draining of a previous write. With both GPIO and the EPI controller, reads may be performed directly, in which case the current pin states are read back. With the EPI controller, the non-blocking interface may also be used to perform reads based on a fixed time rule via the **EPIBAUD** clock rate.

Table 11-12 on page 882 shows how the EPI0S[31:0] signals function while in General-Purpose mode. Notice that the address connections vary depending on the data-width restrictions of the external peripheral.

**Table 11-12. EPI General-Purpose Signal Connections**

| EPI Signal | General-Purpose Signal (D8, A20) | General- Purpose Signal (D16, A12) | General- Purpose Signal (D24, A4) | General- Purpose Signal (D32) |
|---|---|---|---|---|
| EPI0S0 | D0 | D0 | D0 | D0 |
| EPI0S1 | D1 | D1 | D1 | D1 |
| EPI0S2 | D2 | D2 | D2 | D2 |
| EPI0S3 | D3 | D3 | D3 | D3 |
| EPI0S4 | D4 | D4 | D4 | D4 |
| EPI0S5 | D5 | D5 | D5 | D5 |
| EPI0S6 | D6 | D6 | D6 | D6 |
| EPI0S7 | D7 | D7 | D7 | D7 |
| EPI0S8 | A0 | D8 | D8 | D8 |
| EPI0S9 | A1 | D9 | D9 | D9 |
| EPI0S10 | A2 | D10 | D10 | D10 |
| EPI0S11 | A3 | D11 | D11 | D11 |
| EPI0S12 | A4 | D12 | D12 | D12 |
| EPI0S13 | A5 | D13 | D13 | D13 |
| EPI0S14 | A6 | D14 | D14 | D14 |
| EPI0S15 | A7 | D15 | D15 | D15 |
| EPI0S16 | A8 | A0[a] | D16 | D16 |
| EPI0S17 | A9 | A1 | D17 | D17 |
| EPI0S18 | A10 | A2 | D18 | D18 |
| EPI0S19 | A11 | A3 | D19 | D19 |
| EPI0S20 | A12 | A4 | D20 | D20 |

**Table 11-12. EPI General-Purpose Signal Connections *(continued)***

| EPI Signal | General-Purpose Signal (D8, A20) | General- Purpose Signal (D16, A12) | General- Purpose Signal (D24, A4) | General- Purpose Signal (D32) |
|---|---|---|---|---|
| EPI0S21 | A13 | A5 | D21 | D21 |
| EPI0S22 | A14 | A6 | D22 | D22 |
| EPI0S23 | A15 | A7 | D23 | D23 |
| EPI0S24 | A16 | A8 | A0[b] | D24 |
| EPI0S25 | A17 | A9 | A1 | D25 |
| EPI0S26 | A18 | A10 | A2 | D26 |
| EPI0S27 | A19 | A11 | A3 | D27 |
| EPI0S28 | WR | WR | WR | D28 |
| EPI0S29 | RD | RD | RD | D29 |
| EPI0S30 | Frame | Frame | Frame | D30 |
| EPI0S31 | Clock | Clock | Clock | D31 |

a. In this mode, half-word accesses are used. AO is the LSB of the address and is equivalent to the system A1 address.

b. In this mode, word accesses are used. AO is the LSB of the address and is equivalent to the system A2 address.

#### 11.4.4.1 Bus Operation

A basic access is 1 EPI clock for write cycles and 2 EPI clocks for read cycles. An additional EPI clock can be inserted into a write cycle by setting the WR2CYC bit in the **EPIGPCFG** register.

**Figure 11-19. Single-Cycle Single Write Access, FRM50=0, FRMCNT=0, WR2CYC=0**

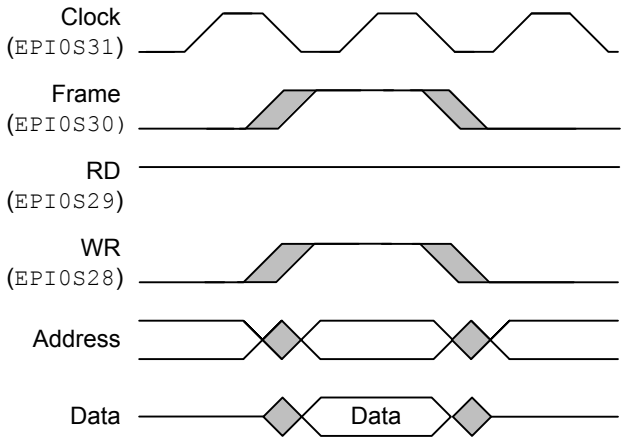


**Figure 11-20. Two-Cycle Read, Write Accesses, FRM50=0, FRMCNT=0, WR2CYC=1**

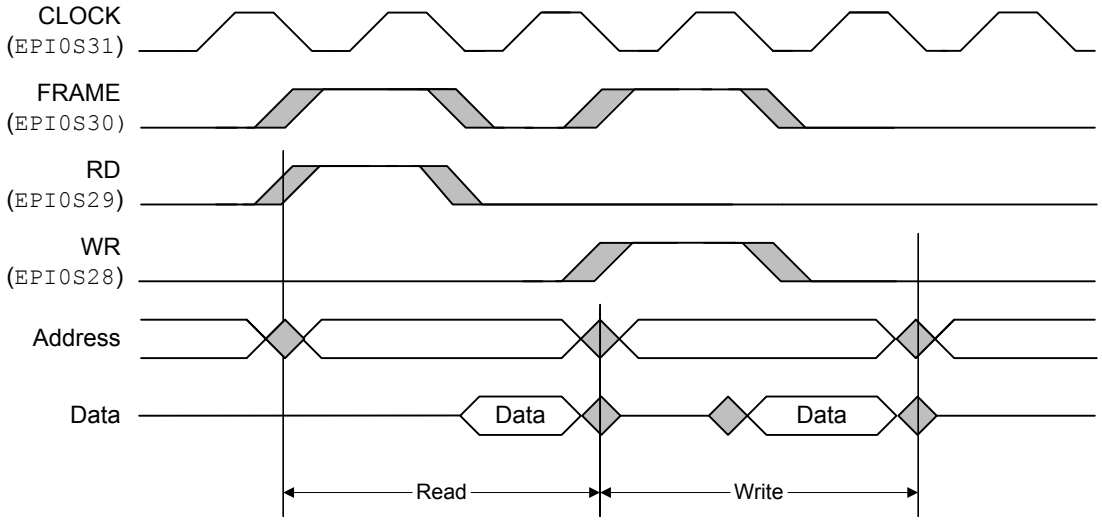
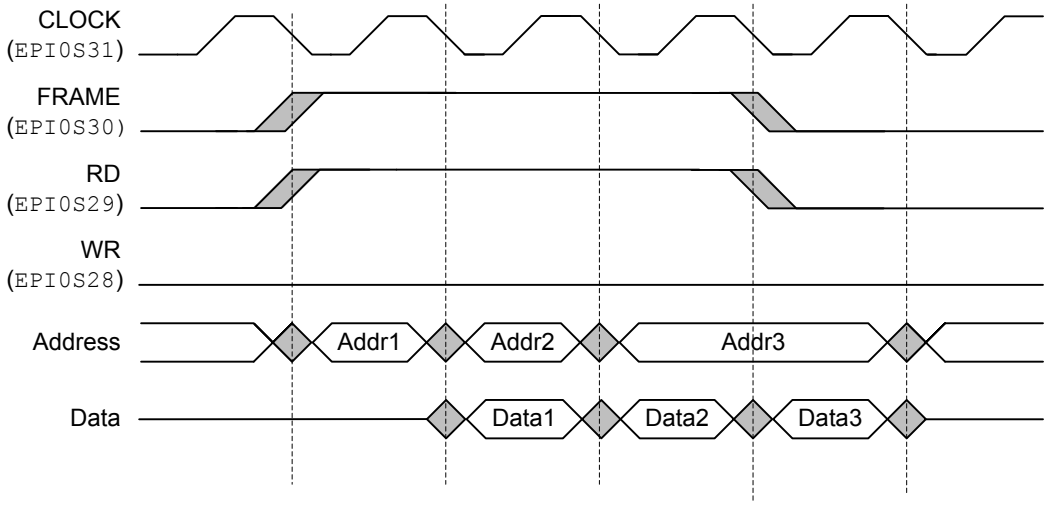


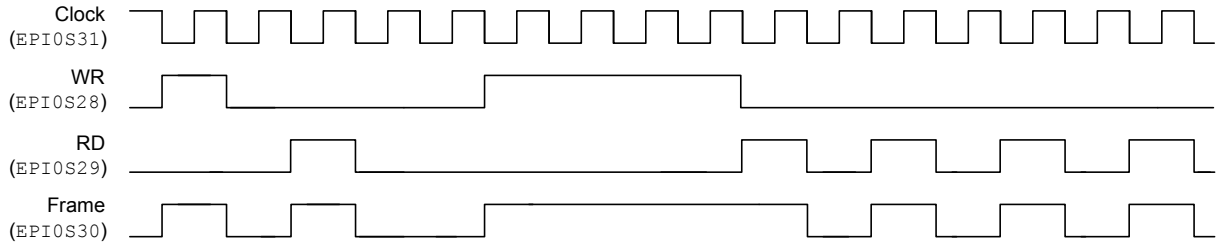

**Figure 11-21. Read Accesses, FRM50=0, FRMCNT=0**




### *FRAME Signal Operation*

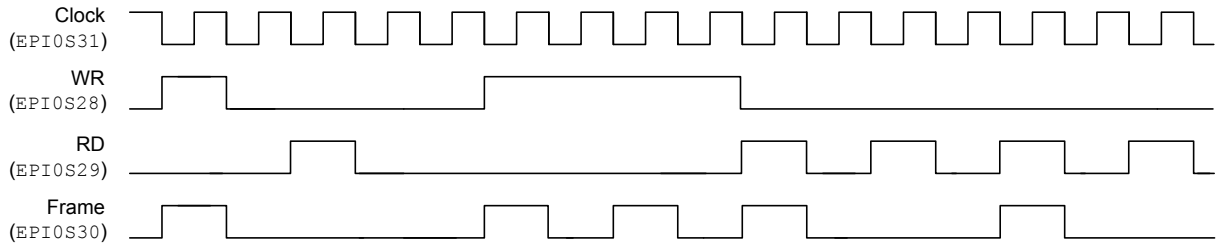
The operation of the FRAME signal is controlled by the FRMCNT and FRM50 bits. When FRM50 is clear, the FRAME signal is high whenever the WR or RD strobe is high. When FRMCNT is clear, the FRAME signal is simply the logical OR of the WR and RD strobes so the FRAME signal is high during every read or write access, see Figure 11-22 on page 885.

**Figure 11-22. FRAME Signal Operation, FRM50=0 and FRMCNT=0**



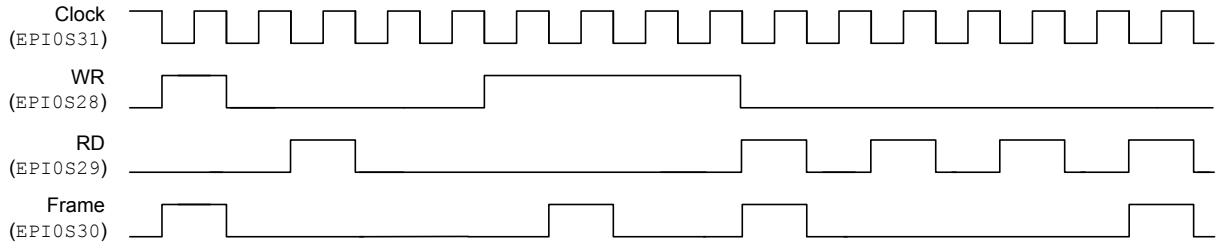

If the FRMCNT field is 0x1, then the FRAME signal pulses high during every other read or write access, see Figure 11-23 on page 885.

**Figure 11-23. FRAME Signal Operation, FRM50=0 and FRMCNT=1**



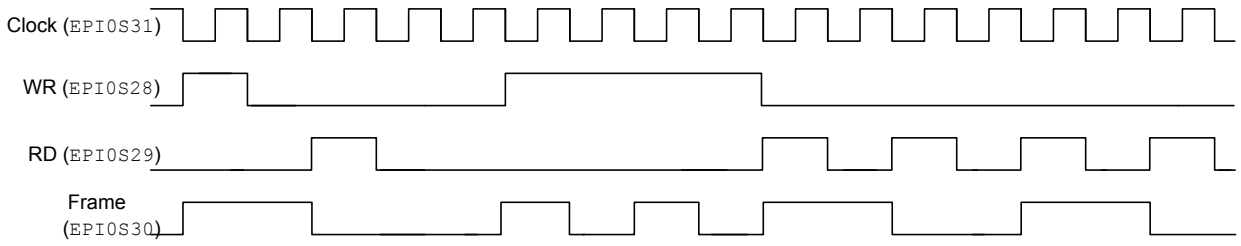

If the FRMCNT field is 0x2 and FRM50 is clear, then the FRAME signal pulses high during every third access, and so on for every value of FRMCNT, see Figure 11-24 on page 885.

**Figure 11-24. FRAME Signal Operation, FRM50=0 and FRMCNT=2**



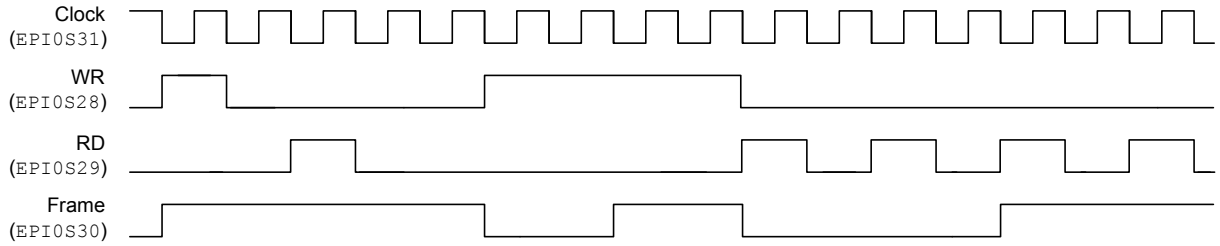

When FRM50 is set, the FRAME signal transitions on the rising edge of either the WR or RD strobes. When FRMCNT=0, the FRAME signal transitions on the rising edge of WR or RD for every access, see Figure 11-25 on page 885.

**Figure 11-25. FRAME Signal Operation, FRM50=1 and FRMCNT=0**



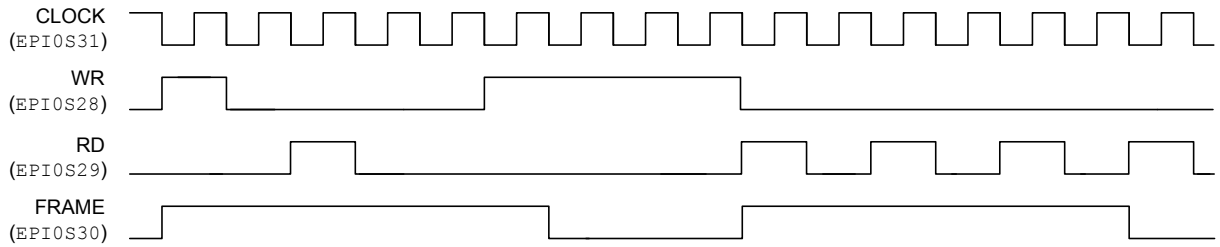

When FRMCNT=1, the FRAME signal transitions on the rising edge of the WR or RD strobes for every other access, see Figure 11-26 on page 886.

**Figure 11-26. FRAME Signal Operation, FRM50=1 and FRMCNT=1**




When FRMCNT=2, the FRAME signal transitions the rising edge of the WR or RD strobes for every third access, and so on for every value of FRMCNT, see Figure 11-27 on page 886.

**Figure 11-27. FRAME Signal Operation, FRM50=1 and FRMCNT=2**




### *EPI Clock Operation*

If the CLKGATE bit in the **EPIGPCFG** register is clear, the EPI clock always toggles when General-purpose mode is enabled. If CLKGATE is set, the clock is output only when a transaction is occurring, otherwise the clock is held high. If the WR2CYC bit is clear, the EPI clock begins toggling 1 cycle before the WR strobe goes High. If the WR2CYC bit is set, the EPI clock begins toggling when the WR strobe goes High. The clock stops toggling after the first rising edge after the WR strobe is deasserted. The RD strobe operates in the same manner as the WR strobe when the WR2CYC bit is set. See Figure 11-28 on page 886 and Figure 11-29 on page 887.

**Figure 11-28. EPI Clock Operation, CLKGATE=1, WR2CYC=0**

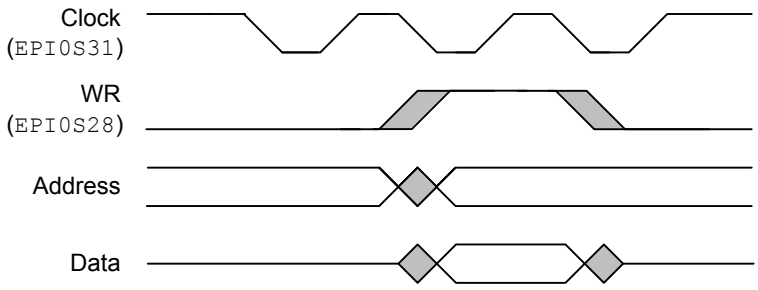
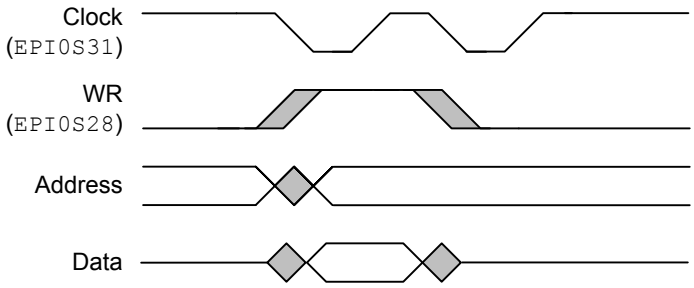


**Figure 11-29. EPI Clock Operation, CLKGATE=1, WR2CYC=1**




## 11.5 Register Map

Table 11-13 on page 887 lists the EPI registers. The offset listed is a hexadecimal increment to the register's address, relative to the base address of 0x400D.0000. Note that the EPI controller clock must be enabled before the registers can be programmed (see page 403). There must be a delay of 3 system clocks after the EPI module clock is enabled before any EPI module registers are accessed.

**Note:** A write immediately followed by a read of the same register, may not return correct data. A delay (instruction or NOP) must be inserted between the write and the read for correct operation. Read-write does not have this issue, so use of read-write for clear of error interrupt cause is not affected.

**Note:** For all versions of EPI, only WORD read and write accesses to registers are supported.

**Table 11-13. External Peripheral Interface (EPI) Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | EPICFG | RW | 0x0000.0000 | EPI Configuration | 890 |
| 0x004 | EPIBAUD | RW | 0x0000.0000 | EPI Main Baud Rate | 892 |
| 0x008 | EPIBAUD2 | RW | 0x0000.0000 | EPI Main Baud Rate | 894 |
| 0x010 | EPISDRAMCFG | RW | 0x82EE.0000 | EPI SDRAM Configuration | 896 |
| 0x010 | EPIHB8CFG | RW | 0x0008.FF00 | EPI Host-Bus 8 Configuration | 898 |
| 0x010 | EPIHB16CFG | RW | 0x0008.FF00 | EPI Host-Bus 16 Configuration | 903 |
| 0x010 | EPIGPCFG | RW | 0x0000.0000 | EPI General-Purpose Configuration | 909 |
| 0x014 | EPIHB8CFG2 | RW | 0x0008.0000 | EPI Host-Bus 8 Configuration 2 | 912 |
| 0x014 | EPIHB16CFG2 | RW | 0x0008.0000 | EPI Host-Bus 16 Configuration 2 | 918 |
| 0x01C | EPIADDRMAP | RW | 0x0000.0000 | EPI Address Map | 925 |
| 0x020 | EPIRSIZE0 | RW | 0x0000.0003 | EPI Read Size 0 | 928 |
| 0x024 | EPIRADDR0 | RW | 0x0000.0000 | EPI Read Address 0 | 929 |
| 0x028 | EPIRPSTD0 | RW | 0x0000.0000 | EPI Non-Blocking Read Data 0 | 930 |
| 0x030 | EPIRSIZE1 | RW | 0x0000.0003 | EPI Read Size 1 | 928 |

**Table 11-13. External Peripheral Interface (EPI) Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x034 | EPIRADDR1 | RW | 0x0000.0000 | EPI Read Address 1 | 929 |
| 0x038 | EPIRPSTD1 | RW | 0x0000.0000 | EPI Non-Blocking Read Data 1 | 930 |
| 0x060 | EPISTAT | RO | 0x0000.0000 | EPI Status | 932 |
| 0x06C | EPIRFIFOCNT | RO | - | EPI Read FIFO Count | 934 |
| 0x070 | EPIREADFIFO0 | RO | - | EPI Read FIFO | 935 |
| 0x074 | EPIREADFIFO1 | RO | - | EPI Read FIFO Alias 1 | 935 |
| 0x078 | EPIREADFIFO2 | RO | - | EPI Read FIFO Alias 2 | 935 |
| 0x07C | EPIREADFIFO3 | RO | - | EPI Read FIFO Alias 3 | 935 |
| 0x080 | EPIREADFIFO4 | RO | - | EPI Read FIFO Alias 4 | 935 |
| 0x084 | EPIREADFIFO5 | RO | - | EPI Read FIFO Alias 5 | 935 |
| 0x088 | EPIREADFIFO6 | RO | - | EPI Read FIFO Alias 6 | 935 |
| 0x08C | EPIREADFIFO7 | RO | - | EPI Read FIFO Alias 7 | 935 |
| 0x200 | EPIFIFOLVL | RW | 0x0000.0033 | EPI FIFO Level Selects | 936 |
| 0x204 | EPIWFIFOCNT | RO | 0x0000.0004 | EPI Write FIFO Count | 938 |
| 0x208 | EPIDMATXCNT | RW | 0x0000.0000 | EPI DMA Transmit Count | 939 |
| 0x210 | EPIIM | RW | 0x0000.0000 | EPI Interrupt Mask | 940 |
| 0x214 | EPIRIS | RO | 0x0000.0004 | EPI Raw Interrupt Status | 942 |
| 0x218 | EPIMIS | RO | 0x0000.0000 | EPI Masked Interrupt Status | 944 |
| 0x21C | EPIEISC | RW1C | 0x0000.0000 | EPI Error and Interrupt Status and Clear | 946 |
| 0x308 | EPIHB8CFG3 | RW | 0x0008.0000 | EPI Host-Bus 8 Configuration 3 | 948 |
| 0x308 | EPIHB16CFG3 | RW | 0x0008.0000 | EPI Host-Bus 16 Configuration 3 | 951 |
| 0x30C | EPIHB8CFG4 | RW | 0x0008.0000 | EPI Host-Bus 8 Configuration 4 | 955 |
| 0x30C | EPIHB16CFG4 | RW | 0x0008.0000 | EPI Host-Bus 16 Configuration 4 | 958 |
| 0x310 | EPIHB8TIME | RW | 0x0002.2000 | EPI Host-Bus 8 Timing Extension | 962 |
| 0x310 | EPIHB16TIME | RW | 0x0002.2000 | EPI Host-Bus 16 Timing Extension | 964 |
| 0x314 | EPIHB8TIME2 | RW | 0x0002.2000 | EPI Host-Bus 8 Timing Extension | 966 |
| 0x314 | EPIHB16TIME2 | RW | 0x0002.2000 | EPI Host-Bus 16 Timing Extension | 968 |
| 0x318 | EPIHB8TIME3 | RW | 0x0002.2000 | EPI Host-Bus 8 Timing Extension | 970 |
| 0x318 | EPIHB16TIME3 | RW | 0x0002.2000 | EPI Host-Bus 16 Timing Extension | 972 |
| 0x31C | EPIHB8TIME4 | RW | 0x0002.2000 | EPI Host-Bus 8 Timing Extension | 974 |
| 0x31C | EPIHB16TIME4 | RW | 0x0002.2000 | EPI Host-Bus 16 Timing Extension | 976 |
| 0x360 | EPIHBPSRAM | RW | 0x0000.0000 | EPI Host-Bus PSRAM | 978 |

## 11.6 Register Descriptions

This section lists and describes the EPI registers, in numerical order by address offset.

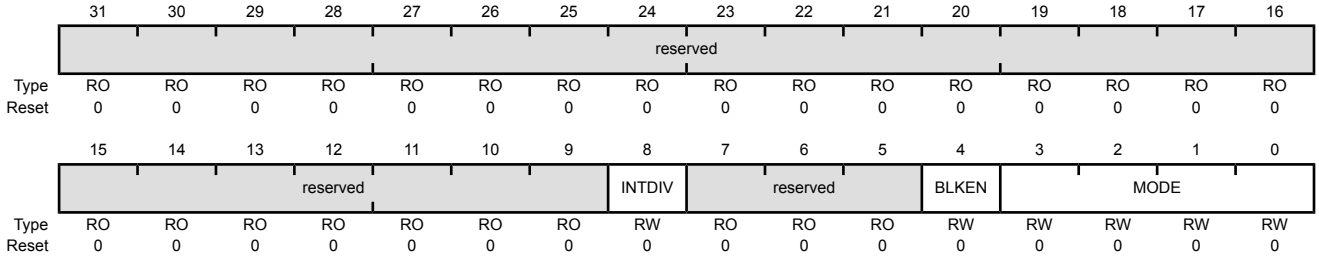
## Register 1: EPI Configuration (EPICFG), offset 0x000

**Important:** The MODE field determines which configuration register is accessed for offsets 0x010 and 0x014. Any write to the **EPICFG** register resets the register contents at offsets 0x010 and 0x014.

The configuration register is used to enable the block, select a mode, and select the basic pin use (based on the mode). Note that attempting to program an undefined MODE field clears the BLKEN bit and disables the EPI controller.

EPI Configuration (EPICFG)

Base 0x400D.0000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | 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. |
| 8 | INTDIV | RW | 0 | Integer Clock Divider Enable<br><br>Value Description<br>0 **EPIBAUD** register values create formula clock divide.<br>1 **EPIBAUD** register values create integer clock divide. |
| 7:5 | reserved | RO | 0x0 | 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. |
| 4 | BLKEN | RW | 0 | Block Enable<br><br>Value Description<br>0 The EPI controller is disabled.<br>1 The EPI controller is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | MODE | RW | 0x0 | Mode Select |

| Value | Description |
|---|---|
| 0x0 | General Purpose<br>General-Purpose mode. Control, address, and data pins are configured using the **EPIGPCFG** and **EPIGPCFG2** registers. |
| 0x1 | SDRAM<br>Supports SDR SDRAM. Control, address, and data pins are configured using the **EPISDRAMCFG** register. |
| 0x2 | 8-Bit Host-Bus (HB8)<br>Host-bus 8-bit interface (also known as the MCU interface). Control, address, and data pins are configured using the **EPIHB8CFG** and **EPIHB8CFG2** registers. |
| 0x3 | 16-Bit Host-Bus (HB16)<br>Host-bus 16-bit interface (standard SRAM). Control, address, and data pins are configured using the **EPIHB16CFG** and **EPIHB16CFG2** registers. |
| 0x3-0xF | Reserved |

## Register 2: EPI Main Baud Rate (EPIBAUD), offset 0x004

The system clock is used internally to the EPI Controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external EPI clock, this register defines the EPI clock emitted. If the mode selected does not use an EPI clock, this register controls the speed of changes on the external interface. Care must be taken to program this register properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 is the bit field used in all modes except in HB8 and HB16 modes with dual chip selects and quad chip selects when different baud rates are selected, see page 912 and page 918. If different baud rates are used, COUNT0 is associated with the address range specified by CS0n and COUNT1 is associated with the address range specified by CS1. The **EPIBAUD2** register configures the baud rates for CS2n and CS3n.

The COUNTn field is not a straight divider or count. The EPI Clock on EPI0S31 is related to the COUNTn field and the system clock as follows:

If COUNTn = 0,

$$EPIClockFreq = SystemClockFreq$$

otherwise:

$$EPIClockFreq = \frac{SystemClockFreq}{\left(\left\lfloor \frac{COUNTn}{2} \right\rfloor + 1\right) \times 2}$$

where the symbol around COUNTn/2 is the floor operator, meaning the largest integer less than or equal to COUNTn/2.

So, for example, a COUNTn of 0x0001 results in a clock rate of ½(system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of ¼(system clock).

The baud rate counter can also be configured as an integer divide by enabling INTDIV in the **EPICFG** register. When enabled, COUNTn of 0x0000 or 0x0001 results in a clock rate equal to system clock. COUNTn of 0x0002 results in a clock rate of 1/2 (system clock). COUNTn of 0x0003 results in a clock rate of 1/3 (system clock).

EPI Main Baud Rate (EPIBAUD)

Base 0x400D.0000
Offset 0x004
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COUNT1 | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COUNT0 | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | COUNT1 | RW | 0x0000 | Baud Rate Counter 1<br>This bit field is only valid with multiple chip selects which are enabled when the CSCFG field is 0x2 or 0x3 or the CSCFGEXT field is set to 1, with CSCFG field as 0x1 or 0x2 and the CSBAUD bit is set in the **EPIHBnCFG2** register.<br>This bit field contains a counter used to divide the system clock by the count.<br>A count of 0 means the system clock is used as is. |
| 15:0 | COUNT0 | RW | 0x0000 | Baud Rate Counter 0<br>This bit field contains a counter used to divide the system clock by the count.<br>A count of 0 means the system clock is used as is. |

## Register 3: EPI Main Baud Rate (EPIBAUD2), offset 0x008

The system clock is used internally to the EPI Controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external EPI clock, this register defines the EPI clock emitted. If the mode selected does not use an EPI clock, this register controls the speed of changes on the external interface. Care must be taken to program this register properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 and COUNT1 are used in quad chip select mode when different baud rates are selected, page 912 or page 918. If different baud rates are used, COUNT0 is associated with the address range specified by CS2n and COUNT1 is associated with the address range specified by CS3n.

The COUNTn field is not a straight divider or count. The EPI Clock on EPI0S31 is related to the COUNTn field and the system clock as follows:

If COUNTn = 0,

$$EPIClockFreq = SystemClockFreq$$

otherwise:

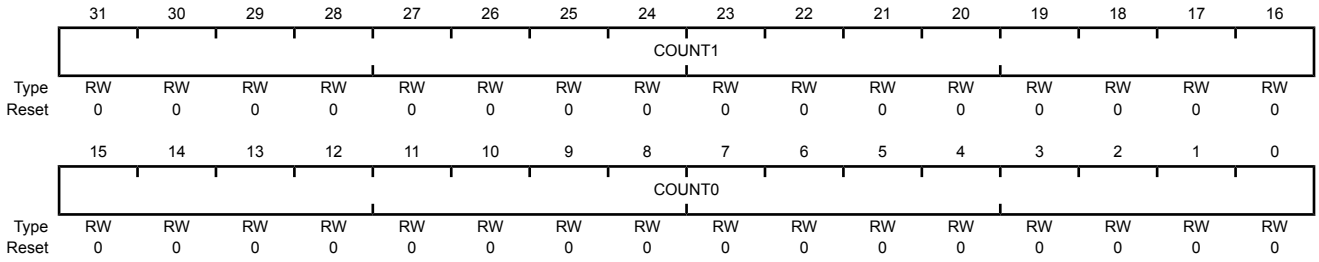
$$EPIClockFreq = \frac{SystemClockFreq}{\left(\left\lfloor \frac{COUNTn}{2} \right\rfloor + 1\right) \times 2}$$

where the symbol around COUNTn/2 is the floor operator, meaning the largest integer less than or equal to COUNTn/2.

So, for example, a COUNTn of 0x0001 results in a clock rate of ½(system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of ¼(system clock).

EPI Main Baud Rate (EPIBAUD2)

Base 0x400D.0000
Offset 0x008
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | COUNT1 | RW | 0x0000 | CS3n Baud Rate Counter 1<br>This bit field contains a counter used to divide the system clock by the count.<br>A count of 0 means the system clock is unchanged.<br>This bit field is only valid when quad chip selects are enabled by setting the CSCFGEXT bit to 1 and the CSCFG field to 0x1 or 0x2. In addition, the CSBAUD bit must be set in the **EPIHBnCFG2** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:0 | COUNT0 | RW | 0x0000 | CS2n Baud Rate Counter 0<br>This bit field contains a counter used to divide the system clock by the count.<br>A count of 0 means the system clock is unchanged.<br>This bit field is only valid when quad chip selects are enabled by setting the CSCFGEXT to 1 and the CSCFG field to 0x1 or 0x2. In addition, the CSBAUD bit must be set in the **EPIHBnCFG2** register. |

## Register 4: EPI SDRAM Configuration (EPISDRAMCFG), offset 0x010

**Important:** The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPISDRAMCFG**, the MODE field must be 0x1.

The SDRAM Configuration register is used to specify several parameters for the SDRAM controller. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized.

The SDRAM interface is designed to interface to x16 SDR SDRAMs of 64 MHz or higher, with the address and data pins overlapped (wire ORed on the board). See Table 11-3 on page 856 for pin assignments.

EPI SDRAM Configuration (EPISDRAMCFG)

Base 0x400D.0000
Offset 0x010
Type RW, reset 0x82EE.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FREQ | | reserved | | | RFSH | | | | | | | | | | |
| Type | RW | RW | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | SLEEP | reserved | | | | | | | SIZE | |
| Type | RO | RO | RO | RO | RO | RO | RW | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:30 | FREQ | RW | 0x2 | EPI Frequency Range<br>This field configures the frequency range used for delay references by internal counters. This EPI frequency is the system frequency with the divider programmed by the COUNT0 bit in the **EPIBAUDn** register bit. This field affects the power up, precharge, and auto refresh delays. This field does not affect the refresh counting, which is configured separately using the RFSH field (and is based on system clock rate and number of rows per bank). The ranges are:<br>Value Description<br>0x0 0 - 15 MHz<br>0x1 15 - 30 MHz<br>0x2 30 - 50 MHz<br>0x3 50 - 100 MHz |
| 29:27 | reserved | RO | 0x0 | 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. |
| 26:16 | RFSH | RW | 0x2EE | Refresh Counter<br>This field contains the refresh counter in EPI clocks. The reset value of 0x2EE provides a refresh period of 64 ms when using a 50 MHz EPI clock. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:10 | reserved | RO | 0x0 | 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. |
| 9 | SLEEP | RW | 0 | Sleep Mode<br><br>Value Description<br>0 No effect.<br>1 The SDRAM is put into low power state, but is self-refreshed. |
| 8:2 | reserved | RO | 0 | 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. |
| 1:0 | SIZE | RW | 0x0 | Size of SDRAM<br>The value of this field affects address pins and behavior.<br><br>Value Description<br>0x0 64 megabits (8MB)<br>0x1 128 megabits (16MB)<br>0x2 256 megabits (32MB)<br>0x3 512 megabits (64MB) |

## Register 5: EPI Host-Bus 8 Configuration (EPIHB8CFG), offset 0x010

**Important:** The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB8CFG**, the MODE field must be 0x2.

The Host Bus 8 Configuration register is activated when the HB8 mode is selected. The HB8 mode supports muxed address/data (overlay of lower 8 address and all 8 data pins), separate address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the HB8 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, CPLDs/FPGAs, and devices with an MCU/HostBus slave or 8-bit FIFO interface support.

Refer to Table 11-8 on page 864 for information on signal configuration controlled by this register and the **EPIHB8CFG2** register.

If less address pins are required, the corresponding AFSEL bit (page 803) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 8 Mode can be configured to use one to four chip selects with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for Device 1 and 0 respectively, then address 0x6800.0000 accesses Device 0 (Device 1 has its CEn high), and 0x6400.0000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WRn or RDn signals write or read when ALE is low thus providing CEn functionality.

EPI Host-Bus 8 Configuration (EPIHB8CFG)

Base 0x400D.0000
Offset 0x010
Type RW, reset 0x0008.FF00

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLKGATE | CLKGATEI | CLKINV | RDYEN | IRDYINV | reserved | | | XFFEN | XFEEN | WRHIGH | RDHIGH | ALEHIGH | reserved | | |
| Type RW | RW | RW | RW | RW | RO | RO | RO | RW | RW | RW | RW | RW | RO | RO | RO |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAXWAIT | | | | | | | | WRWS | | RDWS | | reserved | | MODE | |
| Type RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RO | RO | RW | RW |
| Reset 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CLKGATE | RW | 0 | Clock Gated<br><br>Value Description<br>0 The EPI clock is free running.<br>1 The EPI clock is held low.<br><br>**Note:** A software application should only set the CLKGATE bit when there are no pending transfers or no EPI register access has been issued. |
| 30 | CLKGATEI | RW | 0 | Clock Gated when Idle<br><br>Value Description<br>0 The EPI clock is free running.<br>1 The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low.<br><br>Note that EPI0S32 is an iRDY signal if RDYEN is set. CLKGATEI is ignored if CLKPIN is 0 or if the COUNT0 field in the **EPIBAUD** register is cleared. |
| 29 | CLKINV | RW | 0 | Invert Output Clock Enable<br><br>Value Description<br>0 No effect.<br>1 Invert EPI clock to ensure the rising edge is centered for outbound signal's setup and hold. Inbound signal is captured on rising edge EPI clock. |
| 28 | RDYEN | RW | 0 | Input Ready Enable<br><br>Value Description<br>0 No effect.<br>1 An external ready can be used to control the continuation of the current access. If this bit is set and the iRDY signal (EPIS032) is low, the current access is stalled. |
| 27 | IRDYINV | RW | 0 | Input Ready Invert<br><br>Value Description<br>0 No effect.<br>1 Invert the polarity of incoming external ready (iRDY signal). If this bit is set and the iRDY signal (EPIS032) is high the current access is stalled. |
| 26:24 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23 | XFFEN | RW | 0 | External FIFO FULL Enable<br>Value Description<br>0 No effect.<br>1 An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL full signal is high, XFIFO writes are stalled. |
| 22 | XFEEN | RW | 0 | External FIFO EMPTY Enable<br>Value Description<br>0 No effect.<br>1 An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled. |
| 21 | WRHIGH | RW | 0 | WRITE Strobe Polarity<br>Value Description<br>0 The WRITE strobe for CS0n is WRn (active Low).<br>1 The WRITE strobe for CS0n is WR (active High). |
| 20 | RDHIGH | RW | 0 | READ Strobe Polarity<br>Value Description<br>0 The READ strobe for CS0n is RDn (active Low).<br>1 The READ strobe for CS0n is RD (active High). |
| 19 | ALEHIGH | RW | 1 | ALE Strobe Polarity<br>Value Description<br>0 The address latch strobe for CS0n accesses is ALEn (active Low).<br>1 The address latch strobe for CS0n accesses is ALE (active High). |
| 18:16 | reserved | RO | 0x0 | 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. |
| 15:8 | MAXWAIT | RW | 0xFF | Maximum Wait<br>This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY).<br>When the MAXWAIT value is reached the ERRRIS interrupt status bit is set in the **EPIRIS** register. When this field is clear, the transaction can be held off forever without a system interrupt.<br>**Note:** When the MODE field is configured to be 0x2 and the **BLKEN** bit is set in the **EPICFG** register, enabling HB8 mode, this field defaults to 0xFF. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | WRWS | RW | 0x0 | Write Wait States<br>This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the **EPIHB8TIME** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active WRn is 2 EPI clocks.<br>0x1 Active WRn is 4 EPI clocks.<br>0x2 Active WRn is 6 EPI clocks.<br>0x3 Active WRn is 8 EPI clocks.<br>This field is used in conjunction with the **EPIBAUD** register. |
| 5:4 | RDWS | RW | 0x0 | Read Wait States<br>This field adds wait states to the data phase of CS0n (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB8TIME** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks.<br>0x1 Active RDn is 4 EPI clocks.<br>0x2 Active RDn is 6 EPI clocks.<br>0x3 Active RDn is 8 EPI clocks.<br>This field is used in conjunction with the **EPIBAUD** register |
| 3:2 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | MODE | RW | 0x0 | Host Bus Sub-Mode<br>This field determines which of four Host Bus 8 sub-modes to use. Sub-mode use is determined by the connected external peripheral. See Table 11-8 on page 864 for information on how this bit field affects the operation of the EPI signals. When used with multiple chip select option and the CSBAUD bit is set to 1 in the **EPIHB8CFG2** register, this configuration is for CS0n. If the multiple chip select option is enabled and CSBAUD is clear, all chip-selects use the MODE encoding programmed in this register.<br>Value Description<br>0x0 ADMUX – AD[7:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[7:0]<br>Data and address are separate.<br>0x2 Continuous Read - D[7:0]<br>This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing.<br>0x3 XFIFO – D[7:0]<br>This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.<br>Note that the XFIFO can only be used in asynchronous mode. |

## Register 6: EPI Host-Bus 16 Configuration (EPIHB16CFG), offset 0x010

**Important:** The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB16CFG**, the MODE field must be 0x3.

The Host Bus 16 sub-configuration register is activated when the HB16 mode is selected. The HB16 mode supports muxed address/data (overlay of lower 16 address and all 16 data pins), separated address/data, and address-less FIFO mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the HB16 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, and CPLDs/FPGAs, and devices with an MCU/HostBus slave or 16-bit FIFO interface support.

Refer to Table 11-9 on page 866 for information on signal configuration controlled by this register and the **EPIHB16CFG2** register.

If less address pins are required, the corresponding AFSEL bit (page 803) should not be enabled so the EPI controller does not drive those pins, and they are available as standard GPIOs.

EPI Host-Bus 16 Mode can be configured to use one to four chip selects with and without the use of ALE. If an alternative to chip selects are required, a chip enable can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins EPI0S27 and EPI0S26 are used for Device 1 and 0 respectively, then address 0x6800.0000 accesses Device 0 (Device 1 has its CEn high), and 0x6400.0000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WRn or RDn signals write or read when ALE is low thus providing CEn functionality.

EPI Host-Bus 16 Configuration (EPIHB16CFG)

Base 0x400D.0000
Offset 0x010
Type RW, reset 0x0008.FF00

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLKGATE | CLKGATEI | CLKINV | RDYEN | IRDYINV | reserved | | | XFFEN | XFEEN | WRHIGH | RDHIGH | ALEHIGH | WRCRE | RDCRE | BURST |
| Type RW | RW | RW | RW | RW | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAXWAIT | | | | | | | | WRWS | | RDWS | | reserved | BSEL | MODE | |
| Type RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RO | RW | RW | RW |
| Reset 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CLKGATE | RW | 0 | Clock Gated<br><br>Value Description<br>0 The EPI clock is free running.<br>1 The EPI clock is held low.<br><br>**Note:** A software application should only set the CLKGATE bit when there are no pending transfers or no EPI register access has been issued. |
| 30 | CLKGATEI | RW | 0 | Clock Gated Idle<br><br>Value Description<br>0 The EPI clock is free running.<br>1 The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low.<br><br>Note that EPI0S32 is an iRDY signal if RDYEN is set. CLKGATEI is ignored if CLKPIN is 0 or if the COUNT0 field in the **EPIBAUD** register is cleared. |
| 29 | CLKINV | RW | 0 | Invert Output Clock Enable<br><br>**Note:** If operating in asynchronous mode, CLKINV must be 0.<br><br>Value Description<br>0 No effect.<br>1 Invert EPI clock to ensure the rising edge is centered for outbound signal's setup and hold. Inbound signal is captured on rising edge EPI clock. |
| 28 | RDYEN | RW | 0 | Input Ready Enable<br><br>Value Description<br>0 No effect.<br>1 An external ready (iRDY) can be used to control the continuation of the current access. If this bit is set and the iRDY signal (EPIS032) is low, the current access is stalled. |
| 27 | IRDYINV | RW | 0 | Input Ready Invert<br><br>Value Description<br>0 No effect.<br>1 Invert polarity of incoming external ready. If this bit is set and the iRDY signal (EPIS032) is high the current access is stalled. |
| 26:24 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23 | XFFEN | RW | 0 | External FIFO FULL Enable<br>Value Description<br>0 No effect.<br>1 An external FIFO full signal can be used to control write cycles. If this bit is set and the FFULL signal is high, XFIFO writes are stalled. |
| 22 | XFEEN | RW | 0 | External FIFO EMPTY Enable<br>Value Description<br>0 No effect.<br>1 An external FIFO empty signal can be used to control read cycles. If this bit is set and the FEMPTY signal is high, XFIFO reads are stalled. |
| 21 | WRHIGH | RW | 0 | WRITE Strobe Polarity<br>Value Description<br>0 The WRITE strobe for CS0n is WRn (active Low).<br>1 The WRITE strobe for CS0n is WR (active High). |
| 20 | RDHIGH | RW | 0 | READ Strobe Polarity<br>Value Description<br>0 The READ strobe for CS0n is RDn (active Low).<br>1 The READ strobe for CS0n is RD (active High). |
| 19 | ALEHIGH | RW | 1 | ALE Strobe Polarity<br>Value Description<br>0 The address latch strobe for CS0n is ALEn (active Low).<br>1 The address latch strobe for CS0n is ALE (active High). |
| 18 | WRCRE | RW | 0 | PSRAM Configuration Register Write<br>Used for PSRAM configuration registers.<br>With WRCRE set, the next transaction by the EPI will be a write of the CR bit field in the **EPIHBPSRAM** register to the configuration register (CR) of the PSRAM. The WRCRE bit will self clear once the write-enabled CRE access is complete.<br>Value Description<br>0 No Action.<br>1 Start CRE write transaction for CS0n. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | RDCRE | RW | 0 | PSRAM Configuration Register Read<br>Enables read of PSRAM configuration registers.<br>With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR). This bit self clears once the read-enabled CRE access is complete. The address for the CRE access is located at **EPIHBPSRAM[19:18]**. The read data is returned on **EPIHBPSRAM[15:0].**<br>Value Description<br>0 No Action.<br>1 Start CRE read transaction for CS0n. |
| 16 | BURST | RW | 0 | Burst Mode<br>Burst mode must be used with an ALE-enabled interface. Burst mode must be used with ADMUX, which is configured by the MODE field in the **EPIHB16CFG** register.<br>**Note:** Burst mode is optimized for word-length accesses.<br>Value Description<br>0 Burst mode is disabled.<br>1 Burst mode is enabled for CS0n or single chip access. |
| 15:8 | MAXWAIT | RW | 0xFF | Maximum Wait<br>This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY).<br>When this field is clear, the transaction can be held off forever without a system interrupt.<br>**Note:** When the MODE field is configured to be 0x3 and the **BLKEN** bit is set in the **EPICFG** register, enabling HB16 mode, this field defaults to 0xFF. |
| 7:6 | WRWS | RW | 0x0 | Write Wait States<br>This field adds wait states to the data phase of CS0n (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit **EPIHB16TIME** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active WRn is 2 EPI clocks.<br>0x1 Active WRn is 4 EPI clocks.<br>0x2 Active WRn is 6 EPI clocks.<br>0x3 Active WRn is 8 EPI clocks.<br>This field is used in conjunction with the **EPIBAUD** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | RDWS | RW | 0x0 | Read Wait States<br>This field adds wait states to the data phase of CS0n (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB16TIME** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks.<br>0x1 Active RDn is 4 EPI clocks.<br>0x2 Active RDn is 6 EPI clocks.<br>0x3 Active RDn is 8 EPI clocks.<br>This field is used in conjunction with the **EPIBAUD** register |
| 3 | reserved | RO | 0 | 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. |
| 2 | BSEL | RW | 0 | Byte Select Configuration<br>This bit enables byte select operation.<br>Value Description<br>0 No Byte Selects<br>Data is read and written as 16 bits.<br>1 Enable Byte Selects<br>Two EPI signals function as byte select signals to allow 8-bit transfers. See Table 11-9 on page 866 for details on which EPI signals are used.<br>**Note:** If BSEL = 0, byte accesses cannot be executed. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | MODE | RW | 0x0 | Host Bus Sub-Mode<br>This field determines which of three Host Bus 16 sub-modes to use. Sub-mode use is determined by the connected external peripheral. See Table 11-9 on page 866 for information on how this bit field affects the operation of the EPI signals. When used with multiple chip select option and the CSBAUD bit is set to 1 in the **EPIHB16CFG2** register, this configuration is for CS0n. If the multiple chip select option is enabled and CSBAUD is clear, all chip-selects use the MODE encoding programmed in this register.<br>Value Description<br>0x0 ADMUX – AD[15:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[15:0]<br>Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.<br>0x2 Continuous Read - D[15:0]<br>This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available.<br>0x3 XFIFO – D[15:0]<br>This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.<br>Note that the XFIFO can only be used in asynchronous mode. |

## Register 7: EPI General-Purpose Configuration (EPIGPCFG), offset 0x010

**Important:** The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIGPCFG**, the MODE field must be 0x0.

The General-Purpose configuration register is used to configure the control, data, and address pins. This mode can be used for custom interfaces with FPGAs, CPLDs, and for digital data acquisition and actuator control. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the General-purpose mode is selected again, the register the values must be reinitialized.

This mode is designed for 3 general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs, with 3 sizes of data and optional address. Framing and clock-enable permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the baud rate in the **EPIBAUD** register (when used with the NBRFIFO and/or the WFIFO) or by rate of accesses from software or µDMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free running or gated), a framing signal (with frame size), a ready input (to stretch transactions), read and write strobes, address of varying sizes, and data of varying sizes. Additionally, provisions are made for splitting address and data phases on the external interface.

EPI General-Purpose Configuration (EPIGPCFG)

Base 0x400D.0000
Offset 0x010
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLKPIN | CLKGATE | reserved | | | FRM50 | FRMCNT | | | | reserved | | WR2CYC | reserved | | |
| Type | RW | RW | RO | RO | RO | RW | RW | RW | RW | RW | RO | RO | RW | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | ASIZE | | reserved | | DSIZE | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CLKPIN | RW | 0 | Clock Pin |

| Value | Description |
|---|---|
| 0 | No clock output. |
| 1 | EPI0S31 functions as the EPI clock output. |

The EPI clock is generated from the COUNT0 field in the **EPIBAUD** register (as is the system clock which is divided down from it).

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 30 | CLKGATE | RW | 0 | Clock Gated<br><br>Value Description<br>0 The EPI clock is free running.<br>1 The EPI clock is output only when there is data to write or read (current transaction); otherwise the EPI clock is held low.<br><br>CLKGATE is ignored if CLKPIN is 0 or if the COUNT0 field in the **EPIBAUD** register is cleared. |
| 29:27 | reserved | RO | 0 | 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. |
| 26 | FRM50 | RW | 0 | 50/50 Frame<br><br>Value Description<br>0 The FRAME signal is output as a single pulse, and then held low for the count.<br>1 The FRAME signal is output as 50/50 duty cycle using count (see FRMCNT). |
| 25:22 | FRMCNT | RW | 0x0 | Frame Count<br><br>This field specifies the size of the frame in EPI clocks. The frame counter is used to determine the frame size. The count is FRMCNT+1. So, a FRMCNT of 0 forms a pure transaction valid signal (held high during transactions, low otherwise).<br><br>A FRMCNT of 0 with FRM50 set inverts the FRAME signal on each transaction. A FRMCNT of 1 means the FRAME signal is inverted every other transaction; a value of 15 means every sixteenth transaction.<br><br>If FRM50 is set, the frame is held high for FRMCNT+1 transactions, then held low for that many transactions, and so on.<br><br>If FRM50 is clear, the frame is pulsed high for one EPI clock and then low for FRMCNT EPI clocks. |
| 21:20 | reserved | RO | 0 | 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. |
| 19 | WR2CYC | RW | 0 | 2-Cycle Writes<br><br>Value Description<br>0 Data is output on the same EPI clock cycle as the address.<br>EPI clock begins toggling one cycle before the WR strobe goes High.<br>1 Writes are two EPI clock cycles long, with address on one EPI clock cycle (with the WR strobe asserted) and data written on the following EPI clock cycle (with WR strobe deasserted). The next address (if any) is in the cycle following.<br>If the WR2CYC bit is set, the EPI clock begins toggling when the WR strobe goes High.<br><br>When this bit is set, then the RW bit is forced to be set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 18:6 | reserved | RO | 0 | 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 | ASIZE | RW | 0x0 | Address Bus Size<br>This field defines the size of the address bus. The address can be up to 4-bits wide with a 24-bit data bus, up to 12-bits wide with a 16-bit data bus, and up to 20-bits wide with an 8-bit data bus. If the full address bus is not used, use the least significant address bits. Any unused address bits can be used as GPIOs by clearing the AFSEL bit for the corresponding GPIOs.<br>The values are:<br>Value Description<br>0x0 No address<br>0x1 Up to 4 bits wide.<br>0x2 Up to 12 bits wide. This size cannot be used with 24-bit data.<br>0x3 Up to 20 bits wide. This size cannot be used with data sizes other than 8. |
| 3:2 | reserved | RO | 0x0 | 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. |
| 1:0 | DSIZE | RW | 0x0 | Size of Data Bus<br>This field defines the size of the data bus (starting at EPI0S0). Subsets of these numbers can be created by clearing the AFSEL bit for the corresponding GPIOs. Note that size 32 may not be used with clock, frame, address, or other control.<br>The values are:<br>Value Description<br>0x0 8 Bits Wide (EPI0S0 to EPI0S7)<br>0x1 16 Bits Wide (EPI0S0 to EPI0S15)<br>0x2 24 Bits Wide (EPI0S0 to EPI0S23)<br>0x3 32 Bits Wide (EPI0S0 to EPI0S31)<br>This size may not be used with an EPI clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction. |

## Register 8: EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2), offset 0x014

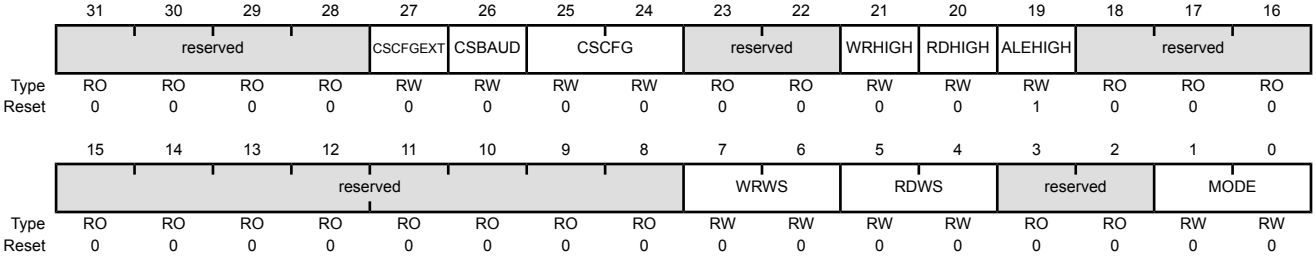
**Important:** The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB8CFG2**, the MODE field of the **EPICFG** register must be 0x2.

This register is used to configure operation while in Host-Bus 8 mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the Host-Bus 8 mode is selected again, the values must be reinitialized.

EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2)

Base 0x400D.0000
Offset 0x014
Type RW, reset 0x0008.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 27 | CSCFGEXT | RW | 0 | Chip Select Extended Configuration<br>This field is used in conjunction with CSCFG, to extend the chip select options, and ALE format. The values 0x0 through 0x3 are from the CSCFG field. The CSCFGEXT bit extends the values to 0x7.<br><br>Value Description<br>0 CSCFG bit field is used in chip select configuration.<br>1 The CSCFG bit field is extended with CSCFGEXT representing the MSB.<br><br>The possible chip select configurations when the CSCFGEXT bit is enabled are shown below: |

### Table 11-14. CSCFGEXT + CSCFG Encodings

| Value | Description |
|---|---|
| 0x0 | ALE Configuration<br>EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (MODE field in the **EPIHB8CFG** register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle. |
| 0x1 | CSn Configuration<br>EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (MODE field in the **EPIHB8CFG** register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low. |
| 0x2 | Dual CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral. |
| 0x3 | ALE with Dual CSn Configuration<br>EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. |
| 0x4 | ALE with Single CSn Configuration<br>EPI0S30 is used as address latch (ALE) and EPI0S27 is used as CSn. |
| 0x5 | Quad CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n. |
| 0x6 | ALE with Quad CSn Configuration<br>EPI0S30 is used as ALE, EPI0S26 is CS0n, and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n. |
| 0x7 | Reserved |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 26 | CSBAUD | RW | 0 | Chip Select Baud Rate and Multiple Sub-Mode Configuration enable<br>This bit is only valid when the CSCFGEXT + CSCFG field is programmed to 0x2 or 0x3, 0x5 or 0x6. This bit configures the baud rate settings for CS0n, CS1n, CS2n, and CS3n.<br>This bit must also be set to allow different sub-mode configurations on chip-selects. If this bit is clear, all chip-select sub-modes are based on the MODE encoding defined in the **EPI8HBCFG** register.<br>If the CSBAUD bit is set in the **EPIHBnCFG2** register and dual- or quad-chip selects are enabled, then the individual chip selects can use different clock frequencies, wait states and strobe polarity.<br>Value Description<br>0 Same Baud Rate and Same Sub-Mode<br>All CSn use the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register and the sub-mode programmed in the MODE field of the **EPIHB8CFG** register.<br>1 Different Baud Rates<br>CS0n uses the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register. CS1n uses the baud rate defined by the COUNT1 field in the **EPIBAUD** register.<br>CS2n uses the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD2** register. CS3n uses the baud rate defined by the COUNT1 field in the **EPIBAUD2** register.<br>In addition, the sub-modes for each chip select are individually programmed in their respective **EPIHB8CFGn** registers. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25:24 | CSCFG | RW | 0x0 | Chip Select Configuration<br>This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects. These bits are also used in combination with the CSCFGEXT bit for further configurations, including quad- chip select.<br><br>Value Description<br>0x0 ALE Configuration<br>EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (HB8MODE field in the **EPIHB8CFG** register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle.<br>0x1 CSn Configuration<br>EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (HB8MODE field in the **EPIHB8CFG** register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low.<br>0x2 Dual CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by two methods. If only external RAM or external PER is enabled in the address map, the most significant address bit for a respective external address map controls CS0n or CS1n. If both external RAM and external PER is enabled, CS0n is mapped to PER and CS1n is mapped to RAM. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.<br>0x3 ALE with Dual CSn Configuration<br>EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. |
| 23:22 | reserved | RO | 0x0 | 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. |
| 21 | WRHIGH | RW | 0 | CS1n WRITE Strobe Polarity<br>This field is used if the CSBAUD bit in the **EPIHB8CFG2** register is enabled.<br><br>Value Description<br>0 The WRITE strobe for CS1n accesses is WRn (active Low).<br>1 The WRITE strobe for CS1n accesses is WR (active High). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 20 | RDHIGH | RW | 0 | CS1n READ Strobe Polarity<br>This field is used if the CSBAUD bit in the **EPIHB8CFG2** register is enabled.<br><br>Value Description<br>0 The READ strobe for CS1n accesses is RDn (active Low).<br>1 The READ strobe for CS1n accesses is RD (active High). |
| 19 | ALEHIGH | RW | 1 | CS1n ALE Strobe Polarity<br>This field is used if the CSBAUD bit in the **EPIHB8CFG2** register is enabled.<br><br>Value Description<br>0 The address latch strobe for CS1n accesses is ALEn (active Low).<br>1 The address latch strobe for CS1n accesses is ALE (active High). |
| 18:8 | reserved | RO | 0 | 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. |
| 7:6 | WRWS | RW | 0x0 | CS1n Write Wait States<br>This field adds wait states to the data phase of CS1n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. The WRWSM bit in the **EPIHB8TIME2** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity.<br>This field is used if the CSBAUD bit is enabled in the **EPIHB8CFG2** register. This field is used in conjunction with the **EPIBAUD** register and is not applicable in BURST mode.<br><br>Value Description<br>0x0 Active WRn is 2 EPI clocks.<br>0x1 Active WRn is 4 EPI clocks<br>0x2 Active WRn is 6 EPI clocks<br>0x3 Active WRn is 8 EPI clocks |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | RDWS | RW | 0x0 | CS1n Read Wait States<br>This field adds wait states to the data phase of CS1n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state encoding adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB8TIME2** register can decrease the number of states by 1 EPI clock cycle for greater granularity.<br>This field is used if the CSBAUD bit is enabled in the **EPIHB8CFG2** register. This field is used in conjunction with the **EPIBAUD** register and is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks<br>0x1 Active RDn is 4 EPI clocks<br>0x2 Active RDn is 6 EPI clocks<br>0x3 Active RDn is 8 EPI clocks |
| 3:2 | reserved | RO | 0x0 | 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. |
| 1:0 | MODE | RW | 0x0 | CS1n Host Bus Sub-Mode<br>This field determines which Host Bus 8 sub-mode to use for CS1n. Sub-mode use is determined by the externally connected peripheral or memory. See Table 11-8 on page 864 for information on how this bit field affects the operation of the EPI signals.<br>**Note:** The CSBAUD bit must be set to enable this CS1n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the **EPIHB8CFG** register.<br>Value Description<br>0x0 ADMUX – AD[7:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[7:0]<br>Data and address are separate.<br>0x2-0x3 reserved |

## Register 9: EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2), offset 0x014

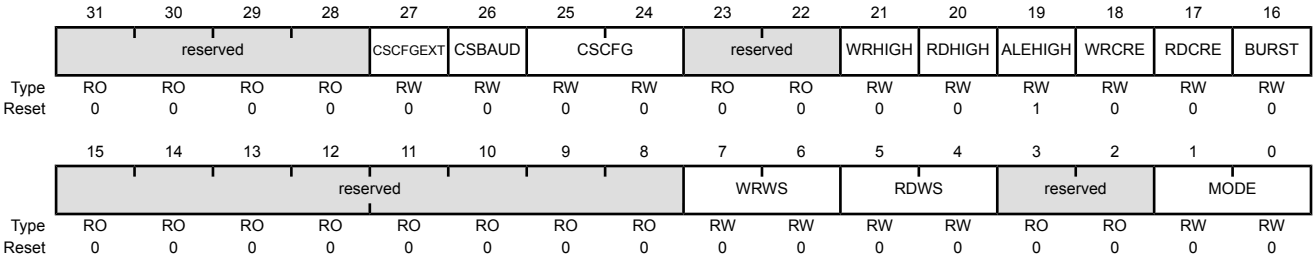
**Important:** The MODE field in the **EPICFG** register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access **EPIHB16CFG2**, the MODE field must be 0x3.

This register is used to configure operation while in Host-Bus 16 mode. Note that this register is reset when the MODE field in the **EPICFG** register is changed. If another mode is selected and the Host-Bus 16 mode is selected again, the values must be reinitialized.

EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2)

Base 0x400D.0000
Offset 0x014
Type RW, reset 0x0008.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 27 | CSCFGEXT | RW | 0 | Chip Select Extended Configuration<br>This field is used in conjunction with CSCFG, to extend the chip select options, and ALE format. The values 0x0 through 0x3 are from the CSCFG field. The CSCFGEXT bit extends the values to 0x7.<br><br>Value Description<br>0 CSCFG bit field is used in chip select configuration.<br>1 The CSCFG bit field is extended with CSCFGEXT representing the MSB.<br><br>The possible chip select configurations when the CSCFGEXT bit is enabled are shown below: |

### Table 11-15. CSCFGEXT + CSCFG Encodings

| Value | Description |
|---|---|
| 0x0 | ALE Configuration<br>EPI0S30 is used as an address latch (ALE). The ALE signal is generally used when the address and data are muxed (MODE field in the **EPIHB16CFG** register is 0x0). The ALE signal is used by an external latch to hold the address through the bus cycle. |
| 0x1 | CSn Configuration<br>EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data are generally not muxed (MODE field in the **EPIHB16CFG** register is 0x1). However, if address and data muxing is needed, the WR signal (EPI0S29) and the RD signal (EPI0S28) can be used to latch the address when CSn is low. |
| 0x2 | Dual CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral. |
| 0x3 | ALE with Dual CSn Configuration<br>EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. |
| 0x4 | ALE with Single CSn Configuration<br>EPI0S30 is used as address latch (ALE) and EPI0S27 is used as CSn. |
| 0x5 | Quad CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n. |
| 0x6 | ALE with Quad CSn Configuration<br>EPI0S30 is used as ALE, EPI0S26 is CS0n, and EPI0S27 is used as CS1n. EPI0S34 is used as CS2n and EPI0S33 is used as CS3n. |
| 0x7 | Reserved |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 26 | CSBAUD | RW | 0 | Chip Select Baud Rate and Multiple Sub-Mode Configuration enable<br>This bit is only valid when the CSCFGEXT + CSCFG field is programmed to 0x2 or 0x3, 0x5 or 0x6. This bit configures the baud rate settings for CS0n, CS1n, CS2n, and CS3n.<br>This bit must also be set to allow different sub-mode configurations on chip-selects. If this bit is clear, all chip-select sub-modes are based on the MODE encoding defined in the **EPI8HBCFG** register.<br>If the CSBAUD bit is set in the **EPIHBnCFG2** register and dual- or quad-chip selects are enabled, then the individual chip selects can use different clock frequencies, wait states and strobe polarity.<br>Value Description<br>0 Same Baud Rate and Same Sub-Mode<br>All CSn use the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register and the sub-mode programmed in the MODE field of the EPIHB16CFG register.<br>1 Different Baud Rates<br>CS0n uses the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD** register. CS1n uses the baud rate defined by the COUNT1 field in the **EPIBAUD** register.<br>CS2n uses the baud rate for the external bus that is defined by the COUNT0 field in the **EPIBAUD2** register. CS3n uses the baud rate defined by the COUNT1 field in the **EPIBAUD2** register.<br>In addition, the sub-modes for each chip select are individually programmed in their respective **EPIHB16CFGn** registers. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25:24 | CSCFG | RW | 0x0 | Chip Select Configuration<br>This field controls the chip select options, including an ALE format, a single chip select, two chip selects, and an ALE combined with two chip selects. These bits are also used in combination with the CSCFGEXT bit for further configurations, including quad- chip select.<br><br>Value Description<br>0x0 ALE Configuration<br>EPI0S30 is used as an address latch (ALE). When using this mode, the address and data should be muxed (HB16MODE field in the **EPIHB16CFG** register should be configured to 0x0). If needed, the address can be latched by external logic.<br>0x1 CSn Configuration<br>EPI0S30 is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (MODE field in the **EPIHB16CFG** register should be configured to 0x1). In this mode, the WR signal (EPI0S29) and the RD signal (EPI0S28) are used to latch the address when CSn is low.<br>0x2 Dual CSn Configuration<br>EPI0S30 is used as CS0n and EPI0S27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.<br>0x3 ALE with Dual CSn Configuration<br>EPI0S30 is used as address latch (ALE), EPI0S27 is used as CS1n, and EPI0S26 is used as CS0n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. |
| 23:22 | reserved | RO | 0x0 | 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. |
| 21 | WRHIGH | RW | 0 | CS1n WRITE Strobe Polarity<br>This field is used if CSBAUD bit of the **EPIHB16CFG2** register is enabled.<br><br>Value Description<br>0 The WRITE strobe for CS1n accesses is WRn (active Low).<br>1 The WRITE strobe for CS1n accesses is WR (active High). |
| 20 | RDHIGH | RW | 0 | CS1n READ Strobe Polarity<br>This field is used if CSBAUD bit of the **EPIHB16CFG2** register is enabled.<br><br>Value Description<br>0 The READ strobe for CS1n accesses is RDn (active Low).<br>1 The READ strobe for CS1n accesses is RD (active High). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 19 | ALEHIGH | RW | 1 | CS1n ALE Strobe Polarity<br>This field is used if CSBAUD bit of the **EPIHB16CFG2** register is enabled.<br><br>Value Description<br>0 The address latch strobe for CS1n accesses is ALEn (active Low).<br>1 The address latch strobe for CS1n accesses is ALE (active High). |
| 18 | WRCRE | RW | 0 | CS1n PSRAM Configuration Register Write<br>Used for the PSRAM configuration registers (CR).<br>With WRCRE set, the next transaction by the EPI is a write of the CR bit field in the **EPIHBPSRAM** register to the configuration register (CR) of the PSRAM. The WRCRE bit self clears once the write-enabled CRE access is complete.<br><br>Value Description<br>0 No Action.<br>1 Start CRE write transaction for CS1n. |
| 17 | RDCRE | RW | 0 | CS1n PSRAM Configuration Register Read<br>Used for the PSRAM configuration registers (CR).<br>With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR). This bit self clears once the CRE access is complete. The address for the CRE access is located at **EPIHBPSRAM[19:18]**. The read data is returned on **EPIHBPSRAM[15:0].**<br><br>Value Description<br>0 No Action.<br>1 Start CRE read transaction for CS1n. |
| 16 | BURST | RW | 0 | CS1n Burst Mode<br>Burst mode must be used with an ALE which is configured by programming the CSCFG and CSCFGEXT fields in the **EPIHB16CFG2** register. Burst mode must be used in ADMUX, which is set by the MODE field in **EPIHB16CFG2**.<br>**Note:** Burst mode is optimized for word-length accesses.<br><br>Value Description<br>0 Burst mode is disabled.<br>1 Burst mode is enabled for CS1n. |
| 15:8 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | WRWS | RW | 0x0 | CS1n Write Wait States<br>This field adds wait states to the data phase of CS1n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state encoding adds 2 EPI clock cycles to the access time. The WRWSM bit in the **EPIHB16TIME2** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity.<br>This field is used if the CSBAUD bit is enabled in the **EPIHB16CFG2** register. This field is used in conjunction with the **EPIBAUD** register and is not applicable in BURST mode.<br>Value Description<br>0x0 Active WRn is 2 EPI clocks<br>0x1 Active WRn is 4 EPI clocks.<br>0x2 Active WRn is 6 EPI clocks<br>0x3 Active WRn is 8 EPI clocks |
| 5:4 | RDWS | RW | 0x0 | CS1n Read Wait States<br>This field adds wait states to the data phase of CS1n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state encoding adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB16TIME2** register can decrease the number of states by 1 EPI clock cycle for greater granularity.<br>This field is used if the CSBAUD bit is enabled in the **EPIHB16CFG2** register. This field is used in conjunction with the **EPIBAUD** register and is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks<br>0x1 Active RDn is 4 EPI clocks<br>0x2 Active RDn is 6 EPI clocks<br>0x3 Active RDn is 8 EPI clocks |
| 3:2 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | MODE | RW | 0x0 | CS1n Host Bus Sub-Mode<br>This field determines which Host Bus 16 sub-mode to use for CS1n.<br>Sub-mode use is determined by the connected external peripheral. See Table 11-9 on page 866 for information on how this bit field affects the operation of the EPI signals. When used with multiple chip select option this configuration is for CS1n.<br>**Note:** The CSBAUD bit must be set to enable this CS1n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the **EPIHB16CFG** register.<br>Value Description<br>0x0 ADMUX – AD[15:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[15:0]<br>Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.<br>0x2-0x3 reserved |

## Register 10: EPI Address Map (EPIADDRMAP), offset 0x01C

This register enables address mapping. The EPI controller can directly address memory and peripherals. In addition, the EPI controller supports address mapping to allow indirect accesses in the External RAM and External Peripheral areas.

If the external device is a peripheral, including a FIFO or a directly addressable device, the EPSZ and EPADR bit fields should be configured for the address space. If the external device is SDRAM, SRAM, or NOR Flash memory, the ERADR and ERSZ bit fields should be configured for the address space.

If one of the dual chip select modes is selected (CSCFGEXT is 0x0 and CSCFG is 0x2 or 0x3 in the **EPIHBnCFG2** register), both chip selects can share the peripheral or the memory space, or one chip select can use the peripheral space and the other can use the memory space. In the **EPIADDRMAP** register, if the EPADR field is not 0x0, the ECADR field is 0x0, and the ERADR field is 0x0, then the address specified by EPADR is used for both chip selects, with CS0n being asserted when the MSB of the address range is 0 and CS1n being asserted when the MSB of the address range is 1. If the ERADR field is not 0x0, the ECADR field is 0x0, and the EPADR field is 0x0, then the address specified by ERADR is used for both chip selects, with the MSB performing the same delineation. If both the EPADR and the ERADR are not 0x0 and the ECADR field is 0x0, then CS0n is asserted for either address range defined by EPADR and CS1n is asserted for either address range defined by ERADR. The two chip selects can also be shared between the code space and memory or peripheral space. If the ECADR field is 0x1, ERADR field is 0x0, and the EPADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by EPADR. If the ECADR field is 0x1, EPADR field is 0x0, and the ERADR field is not 0x0, then CS0n is asserted for the address range defined by ECADR and CS1n is asserted for either address range defined by ERADR.

If one of the Quad-Chip-Select modes is selected (CSCFGEXT is 0x1 and CSCFG is 0x2 or 0x3 in the **EPIHBnCFG2** register), both the peripheral and the memory space must be enabled. In the **EPIADDRMAP** register, the EPADR field is 0x3, the ERADR field is 0x3, and the ECADR field is 0x0. In this case, CS0n maps to 0x6000.0000; CS1n maps to 0x8000.0000; CS2n maps to 0xA000.0000; and CS3n maps to 0xC000.0000. The MODE field of the **EPIHBnCFGn** registers configures the interface for the individual chip selects, which support ADMUX or ADNOMUX. If the CSBAUD bit is clear, all chip selects use the mode configured in the MODE bit field of the **EPIHBnCFG** register. Table 11-5 on page 861 gives a detailed explanation of chip select address range mappings based on which combinations of peripheral and memory space are enabled.

EPI Address Map (EPIADDRMAP)

Base 0x400D.0000
Offset 0x01C
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | ECSZ | | ECADR | | EPSZ | | EPADR | | ERSZ | | ERADR | |
| RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |
| 11:10 | ECSZ | RW | 0x0 | External Code Size<br>This field selects the size of the external code. If the size of the external code is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused).<br>**Note:** When not using byte selects in Host-Bus 16, data is accessed on 2-byte boundaries. As a result, the available address space is double the amount shown below.<br>Value Description<br>0x0 256 bytes; lower address range: 0x00 to 0xFF<br>0x1 64 KB; lower address range: 0x0000 to 0xFFFF<br>0x2 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF<br>0x3 256MB; lower address range: 0x000.0000 to 0x0FFF.FFFF |
| 9:8 | ECADR | RW | 0x0 | External Code Address<br>This field selects address mapping for the external code area.<br>Value Description<br>0x0 Not mapped<br>0x1 At 0x1000.0000<br>0x2 reserved<br>0x3 reserved |
| 7:6 | EPSZ | RW | 0x0 | External Peripheral Size<br>This field selects the size of the external peripheral. If the size of the external peripheral is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused).<br>**Note:** When not using byte selects in Host-Bus 16, data is accessed on 2-byte boundaries. As a result, the available address space is double the amount shown below.<br>Value Description<br>0x0 256 bytes; lower address range: 0x00 to 0xFF<br>0x1 64 KB; lower address range: 0x0000 to 0xFFFF<br>0x2 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF<br>0x3 256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | EPADR | RW | 0x0 | External Peripheral Address<br>This field selects address mapping for the external peripheral area.<br><br>Value Description<br>0x0 Not mapped<br>0x1 At 0xA000.0000<br>0x2 At 0xC000.0000<br>0x3 Only to be used with Host Bus quad chip select. In quad chip select mode, CS2n maps to 0xA000.0000 and CS3n maps to 0xC000.0000. |
| 3:2 | ERSZ | RW | 0x0 | External RAM Size<br>This field selects the size of mapped RAM. If the size of the external memory is larger, a bus fault occurs. If the size of the external memory is smaller, it wraps (upper address bits unused):<br><br>Value Description<br>0x0 256 bytes; lower address range: 0x00 to 0xFF<br>0x1 64 KB; lower address range: 0x0000 to 0xFFFF<br>0x2 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF<br>0x3 256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF |
| 1:0 | ERADR | RW | 0x0 | External RAM Address<br>Selects address mapping for external RAM area:<br><br>Value Description<br>0x0 Not mapped<br>0x1 At 0x6000.0000<br>0x2 At 0x8000.0000<br>0x3 Only to be used with Host Bus quad chip select. In quad chip select mode, CS0n maps to 0x6000.0000 and CS1n maps to 0x8000.0000. |

## Register 11: EPI Read Size 0 (EPIRSIZE0), offset 0x020

## Register 12: EPI Read Size 1 (EPIRSIZE1), offset 0x030

This register selects the size of transactions when performing non-blocking reads with the **EPIRPSTDn** registers. This size affects how the external address is incremented.

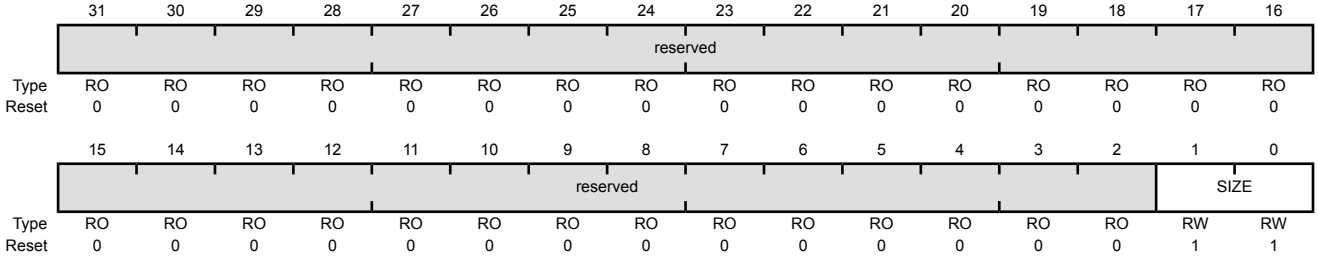
The SIZE field must match the external data width as configured in the **EPIHBnCFG** or **EPIGPCFG** register.

SDRAM mode uses a 16-bit data interface. If SIZE is 0x1, data is returned on the least significant bits (D[7:0]), and the remaining bits D[31:8] are all zeros, therefore the data on bits D[15:8] is lost. If SIZE is 0x2, data is returned on the least significant bits (D[15:0]), and the remaining bits D[31:16] are all zeros.

Note that changing this register while a read is active has an unpredictable effect.

EPI Read Size n (EPIRSIZEn)

Base 0x400D.0000
Offset 0x020
Type RW, reset 0x0000.0003




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | 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. |
| 1:0 | SIZE | RW | 0x3 | Current Size |

| Value | Description |
|---|---|
| 0x0 | reserved |
| 0x1 | Byte (8 bits) |
| 0x2 | Half-word (16 bits) |
| 0x3 | Word (32 bits) |

## Register 13: EPI Read Address 0 (EPIRADDR0), offset 0x024

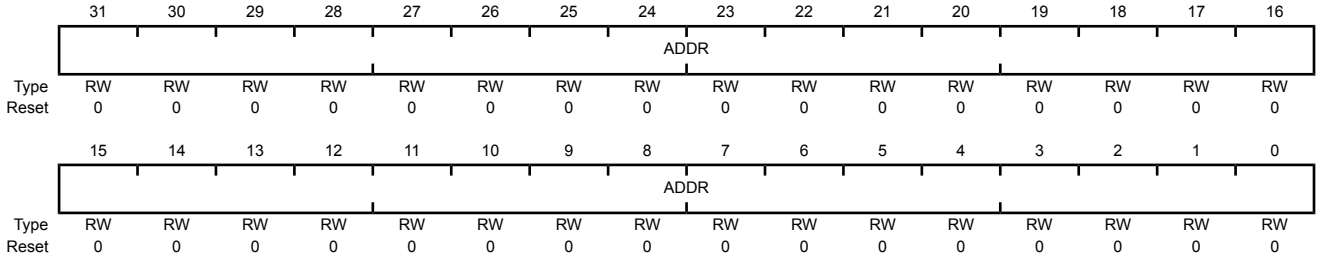
## Register 14: EPI Read Address 1 (EPIRADDR1), offset 0x034

This register holds the current address value. When performing non-blocking reads via the **EPIRPSTDn** registers, this register's value forms the address (when used by the mode). That is, when an **EPIRPSTDn** register is written with a non-0 value, this register is used as the first address. After each read, it is incremented by the size specified by the corresponding **EPIRSIZEn** register. Thus at the end of a read, this register contains the next address for the next read. For example, if the last read was 0x20, and the size is word, then the register contains 0x24. When a non-blocking read is cancelled, this register contains the next address that would have been read had it not been cancelled. For example, if reading by bytes and 0x103 had been read but not 0x104, this register contains 0x104. In this manner, the system can determine the number of values in the NBRFIFO to drain.

Note that changing this register while a read is active has an unpredictable effect due to race condition.

EPI Read Address n (EPIRADDRn)

Base 0x400D.0000
Offset 0x024
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | RW | 0x0000.0000 | Current Address<br>Next address to read. |

## Register 15: EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028

## Register 16: EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038

This register sets up a non-blocking read via the external interface. A non-blocking read is started by writing to this register with the count (other than 0). Clearing this register terminates an active non-blocking read as well as cancelling any that are pending. This register should always be cleared before writing a value other than 0; failure to do so can cause improper operation. Note that both NBR channels can be enabled at the same time, but NBR channel 0 has the highest priority and channel 1 does not start until channel 0 is finished.

The first address is based on the corresponding **EPIRADDRn** register. The address register is incremented by the size specified by the **EPIRSIZEn** register after each read. If the size is less than a word, only the least significant bits of data are filled into the NBRFIFO; the most significant bits are cleared.

Note that all three registers may be written using one STM instruction, such as with a structure copy in C/C++.

The data may be read from the **EPIREADFIFO** register after the read cycle is completed. The interrupt mechanism is normally used to trigger the FIFO reads via ISR or µDMA.

If the countdown has not reached 0 and the NBRFIFO is full, the external interface waits until a NBRFIFO entry becomes available to continue.

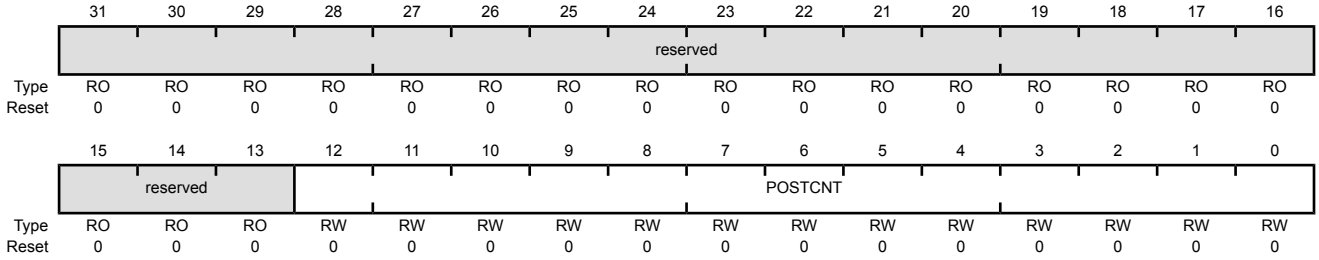
Note: if a blocking read or write is performed through the address mapped area (at 0x6000.0000 through 0xDFFF.FFFF), any current non-blocking read is paused (at the next safe boundary), and the blocking request is inserted. After completion of any blocking reads or writes, the non-blocking reads continue from where they were paused.

The other way to read data is via the address mapped locations (see the **EPIADDRMAP** register), but this method is blocking (core or µDMA waits until result is returned).

To cancel a non-blocking read, clear this register. To make sure that all values read are drained from the NBRFIFO, the **EPISTAT** register must be consulted to be certain that bits NBRBUSY and ACTIVE are cleared. One of these registers should not be cleared until either the other **EPIRPSTDn** register becomes active or the external interface is not busy. At that point, the corresponding **EPIRADDRn** register indicates how many values were read.

EPI Non-Blocking Read Data n (EPIRPSTDn)

Base 0x400D.0000
Offset 0x028
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:13 | reserved | RO | 0x0000.0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12:0 | POSTCNT | RW | 0x000 | Post Count<br>A write of a non-zero value starts a read operation for that count. Note that it is the software's responsibility to handle address wrap-around.<br>Reading this register provides the current count.<br>A write of 0 cancels a non-blocking read (whether active now or pending).<br>Prior to writing a non-zero value, this register must first be cleared. |

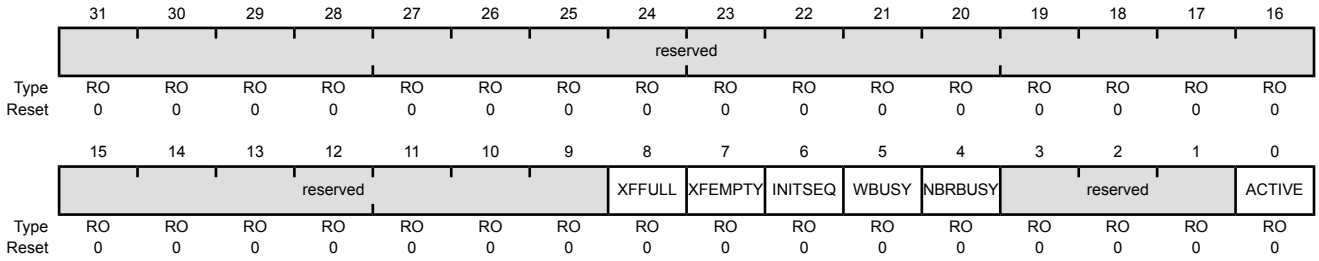
## Register 17: EPI Status (EPISTAT), offset 0x060

This register indicates which non-blocking read register is currently active; it also indicates whether the external interface is busy performing a write or non-blocking read (it cannot be performing a blocking read, as the bus would be blocked and as a result, this register could not be accessed).

This register is useful to determining which non-blocking read register is active when both are loaded with values and when implementing sequencing or sharing.

This register is also useful when canceling non-blocking reads, as it shows how many values were read by the canceled side.

EPI Status (EPISTAT)
Base 0x400D.0000
Offset 0x060
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | 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. |
| 8 | XFFULL | RO | 0 | External FIFO Full<br>This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFFEN bit set in the **EPIHBnCFG** register. The EPI0S26 signal reflects the status of this bit.<br>Value Description<br>0 The external device is not gating the clock.<br>1 The XFIFO is signaling as full (the FIFO full signal is high).<br>Attempts to write in this case are stalled until the XFIFO full signal goes low or the counter times out as specified by the MAXWAIT field. |
| 7 | XFEMPTY | RO | 0 | External FIFO Empty<br>This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFEEN bit set in the **EPIHBnCFG** register. The EPI0S27 signal reflects the status of this bit.<br>Value Description<br>0 The external device is not gating the clock.<br>1 The XFIFO is signaling as empty (the FIFO empty signal is high).<br>Attempts to read in this case are stalled until the XFIFO empty signal goes low or the counter times out as specified by the MAXWAIT field. |

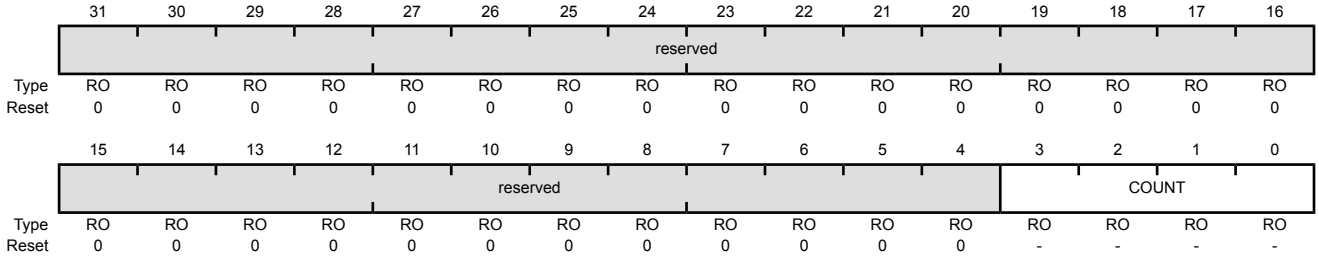
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | INITSEQ | RO | 0 | Initialization Sequence<br><br>Value Description<br>0 The SDRAM interface is not in the wakeup period.<br>1 The SDRAM interface is running through the wakeup period (greater than 100 µs).<br>If an attempt is made to read or write the SDRAM during this period, the access is held off until the wakeup period is complete. |
| 5 | WBUSY | RO | 0 | Write Busy<br><br>Value Description<br>0 The external interface is not performing a write.<br>1 The external interface is performing a write. |
| 4 | NBRBUSY | RO | 0 | Non-Blocking Read Busy<br><br>Value Description<br>0 The external interface is not performing a non-blocking read.<br>1 The external interface is performing a non-blocking read, or if the non-blocking read is paused due to a write. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | ACTIVE | RO | 0 | Register Active<br><br>Value Description<br>0 If NBRBUSY is set, the **EPIRPSTD0** register is active.<br>If the NBRBUSY bit is clear, then neither **EPIRPSTDx** register is active.<br>1 The **EPIRPSTD1** register is active. |

## Register 18: EPI Read FIFO Count (EPIRFIFOCNT), offset 0x06C

This register returns the number of values in the NBRFIFO (the data in the NBRFIFO can be read via the **EPIREADFIFO** register). A race is possible, but that only means that more values may come in after this register has been read.

EPI Read FIFO Count (EPIRFIFOCNT)

Base 0x400D.0000
Offset 0x06C
Type RO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | COUNT | RO | - | FIFO Count<br>Number of filled entries in the NBRFIFO. |

## Register 19: EPI Read FIFO (EPIREADFIFO0), offset 0x070

## Register 20: EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074

## Register 21: EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078

## Register 22: EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C

## Register 23: EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080

## Register 24: EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084

## Register 25: EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088

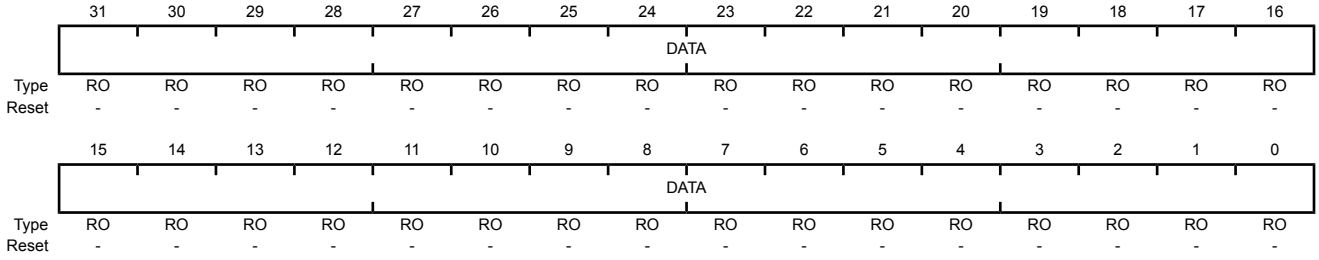
## Register 26: EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C

**Important:** This register is read-sensitive. See the register description for details.

This register returns the contents of the NBRFIFO or 0 if the NBRFIFO is empty. Each read returns the data that is at the top of the NBRFIFO, and then empties that value from the NBRFIFO. The alias registers can be used with the LDMIA instruction for more efficient operation (for up to 8 registers). See *Cortex™-M3/M4 Instruction Set Technical User's Manual (literature number SPMU159)* for more information on the LDMIA instruction.

EPI Read FIFn (EPIREADFIFOn)

Base 0x400D.0000
Offset 0x070
Type RO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RO | - | Reads Data<br>This field contains the data that is at the top of the NBRFIFO. After being read, the NBRFIFO entry is removed. |

## Register 27: EPI FIFO Level Selects (EPIFIFOLVL), offset 0x200

This register allows selection of the FIFO levels which trigger an interrupt to the interrupt controller or, more efficiently, a DMA request to the µDMA. The NBRFIFO select triggers on fullness such that it triggers on match or above (more full) in order for the processor or the µDMA to extract the read data. The WFIFO triggers on emptiness such that it triggers on match or below (less entries) in order for the processor or the µDMA to insert more write data.

It should be noted that the FIFO triggers are not identical to other such FIFOs in TM4C129XKCZAD peripherals. In particular, empty and full triggers are provided to avoid wait states when using blocking operations.

The settings in this register are only meaningful if the µDMA is active or the interrupt is enabled.

Additionally, this register allows protection against writes stalling and notification of performing blocking reads which stall for extra time due to preceding writes. The two functions behave in a non-orthogonal way because read and write are not orthogonal.

The write error bit configures the system such that an attempted write to an already full WFIFO abandons the write and signals an error interrupt to prevent accidental latencies due to stalling writes.

The read error bit configures the system such that after a read has been stalled due to any preceding writes in the WFIFO, the error interrupt is generated. Note that the excess stall is not prevented, but an interrupt is generated after the fact to notify that it has happened.

EPI FIFO Level Selects (EPIFIFOLVL)

Base 0x400D.0000
Offset 0x200
Type RW, reset 0x0000.0033

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | WFERR | RSERR |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | WRFIFO | | | reserved | | RDFIFO | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RO | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0x0000 | 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. |
| 17 | WFERR | RW | 0 | Write Full Error |

| Value | Description |
|---|---|
| 0 | The Write Full error interrupt is disabled. Writes are stalled when the WFIFO is full until a space becomes available but an error is not generated. Note that the Cortex-M3 write buffer may hide that stall if no other memory transactions are attempted during that time. |
| 1 | This bit enables the Write Full error interrupt (WTFULL in the **EPIEISC** register) to be generated when a write is attempted and the WFIFO is full. The write stalls until a WFIFO entry becomes available. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | RSERR | RW | 0 | Read Stall Error<br><br>Value Description<br>0 The Read Stalled error interrupt is disabled. Reads behave as normal and are stalled until any preceding writes have completed and the read has returned a result.<br>1 This bit enables the Read Stalled error interrupt (RSTALL in the **EPIEISC** register) to be generated when a read is attempted and the WFIFO is not empty. The read is still stalled during the time the WFIFO drains, but this error notifies the application that this excess delay has occurred.<br><br>Note that the configuration of this bit has no effect on non-blocking reads. |
| 15:7 | reserved | RO | 0x00 | 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. |
| 6:4 | WRFIFO | RW | 0x3 | Write FIFO<br><br>Value Description<br>0x0 Interrupt is triggered while WRFIFO is empty. It will be deasserted when not empty. This encoding is optimized for burst of 4 writes.<br>0x1 reserved<br>0x2 Interrupt is triggered until there are only two slots available. Thus, trigger is deasserted when there are two WRFIFO entries present. This configuration is optimized for bursts of 2.<br>0x3 Interrupt is triggered until there is one WRFIFO entry available. This configuration expects only single writes.<br>0x4 Trigger interrupt when WRFIFO is not full, meaning trigger will continue to assert until there are four entries in the WRFIFO.<br>0x5-0x7 reserved |
| 3 | reserved | RO | 0 | 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. |
| 2:0 | RDFIFO | RW | 0x3 | Read FIFO<br>This field configures the trigger point for the NBRFIFO.<br><br>Value Description<br>0x0 reserved<br>0x1 Trigger when there are 1 or more entries in the NBRFIFO.<br>0x2 Trigger when there are 2 or more entries in the NBRFIFO.<br>0x3 Trigger when there are 4 or more entries in the NBRFIFO.<br>0x4 Trigger when there are 6 or more entries in the NBRFIFO.<br>0x5 Trigger when there are 7 or more entries in the NBRFIFO.<br>0x6 Trigger when there are 8 entries in the NBRFIFO.<br>0x7 reserved |

## Register 28: EPI Write FIFO Count (EPIWFIFOCNT), offset 0x204

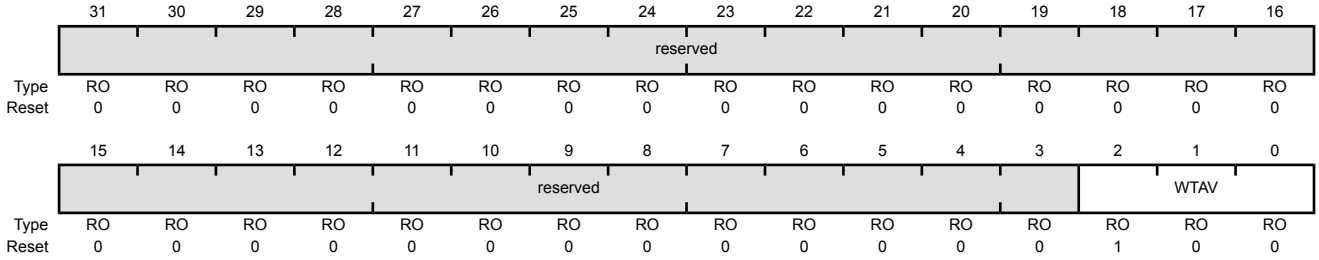
This register contains the number of slots currently available in the WFIFO. This register may be used for polled writes to avoid stalling and for blocking reads to avoid excess stalling (due to undrained writes). An example use for writes may be:

```
for (idx = 0; idx < cnt; idx++) {
while (EPIWFIFOCNT == 0) ;
*ext_ram = *mydata++;
}
```

The above code ensures that writes to the address mapped location do not occur unless the WFIFO has room. Although polling makes the code wait (spinning in the loop), it does not prevent interrupts being serviced due to bus stalling.

EPI Write FIFO Count (EPIWFIFOCNT)

Base 0x400D.0000
Offset 0x204
Type RO, reset 0x0000.0004



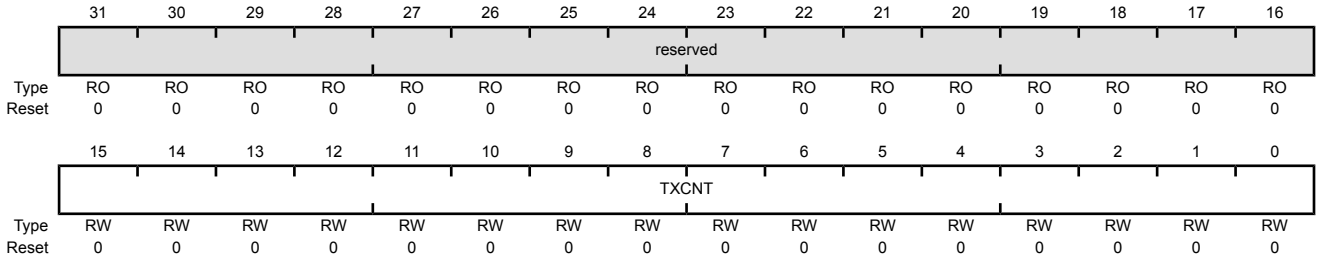

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2:0 | WTAV | RO | 0x4 | Available Write Transactions<br>The number of write transactions available in the WFIFO.<br>When clear, a write is stalled waiting for a slot to become free (from a preceding write completing). |

## Register 29: EPI DMA Transmit Count (EPIDMATXCNT), offset 0x208

This register is used to program the total number of transfers (byte, halfword or word) by the µDMA to WRFIFO. As each transfer is processed by the EPI, the TXCNT bit field value is decreased by 1. When TXCNT = 0, the EPI's uDMA request signal is deasserted.

EPI DMA Transmit Count (EPIDMATXCNT)

Base 0x400D.0000
Offset 0x208
Type RW, reset 0x0000.0000



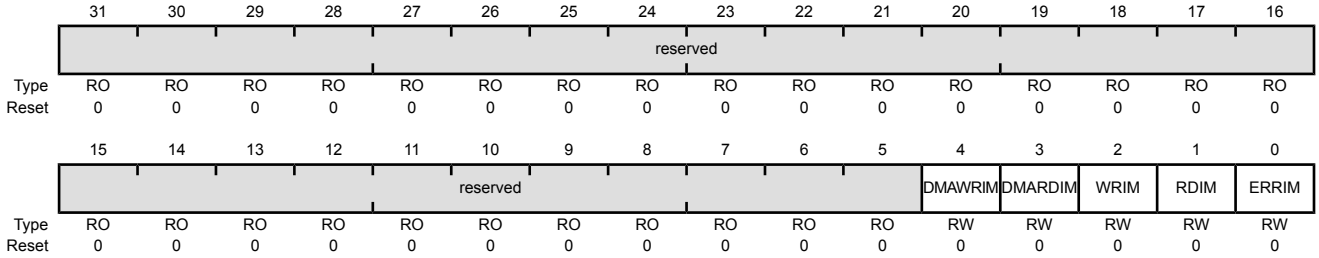

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TXCNT | RW | 0x0000 | DMA Count<br>This field is used to program the total number of transfers (byte, halfword or word) from the µDMA to the EPI WRFIFO. |

## Register 30: EPI Interrupt Mask (EPIIM), offset 0x210

This register is the interrupt mask set or clear register. For each interrupt source (read, write, and error), a mask value of 1 allows the interrupt source to trigger an interrupt to the interrupt controller; a mask value of 0 prevents the interrupt source from triggering an interrupt.

EPI Interrupt Mask (EPIIM)

Base 0x400D.0000
Offset 0x210
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMAWRIM | RW | 0 | Write uDMA Interrupt Mask<br>Value Description<br>0 DMAWRRIS in the **EPIRIS** register is masked and does not cause an interrupt.<br>1 DMAWRRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller. |
| 3 | DMARDIM | RW | 0 | Read uDMA Interrupt Mask<br>Value Description<br>0 DMARDRIS in the **EPIRIS** register is masked and does not cause an interrupt.<br>1 DMARDRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller. |
| 2 | WRIM | RW | 0 | Write FIFO Empty Interrupt Mask<br>Value Description<br>0 WRRIS in the **EPIRIS** register is masked and does not cause an interrupt.<br>1 WRRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | RDIM | RW | 0 | Read FIFO Full Interrupt Mask<br>Value Description<br>0 RDRIS in the **EPIRIS** register is masked and does not cause an interrupt.<br>1 RDRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller. |
| 0 | ERRIM | RW | 0 | Error Interrupt Mask<br>Value Description<br>0 ERRIS in the **EPIRIS** register is masked and does not cause an interrupt.<br>1 ERRIS in the **EPIRIS** register is not masked and can trigger an interrupt to the interrupt controller. |

## Register 31: EPI Raw Interrupt Status (EPIRIS), offset 0x214

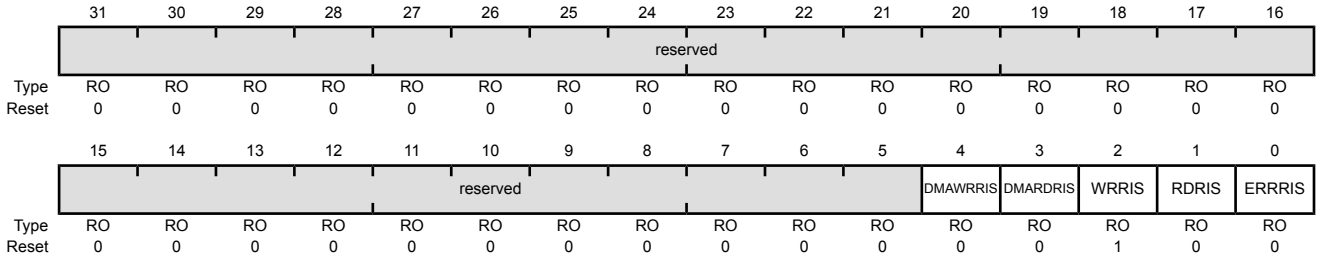
This register is the raw interrupt status register. On a read, it gives the current state of each interrupt source. A write has no effect.

Note that raw status for read and write is set or cleared based on FIFO fullness as controlled by **EPIFIFOLVL**.

Raw status for error is held until the error is cleared by writing to the **EPIEISC** register.

EPI Raw Interrupt Status (EPIRIS)

Base 0x400D.0000
Offset 0x214
Type RO, reset 0x0000.0004




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMAWRRIS | RO | 0 | Write uDMA Raw Interrupt Status<br><br>Value Description<br>0 The write uDMA has not completed.<br>1 The write uDMA has completed.<br><br>This bit is cleared by writing a 1 to the DMAWRIC bit in the **EPIEISC** register. |
| 3 | DMARDRIS | RO | 0 | Read uDMA Raw Interrupt Status<br><br>Value Description<br>0 The read uDMA has not completed.<br>1 The read uDMA has completed.<br><br>This bit is cleared by writing a 1 to the DMARDIC bit in the **EPIEISC** register. |

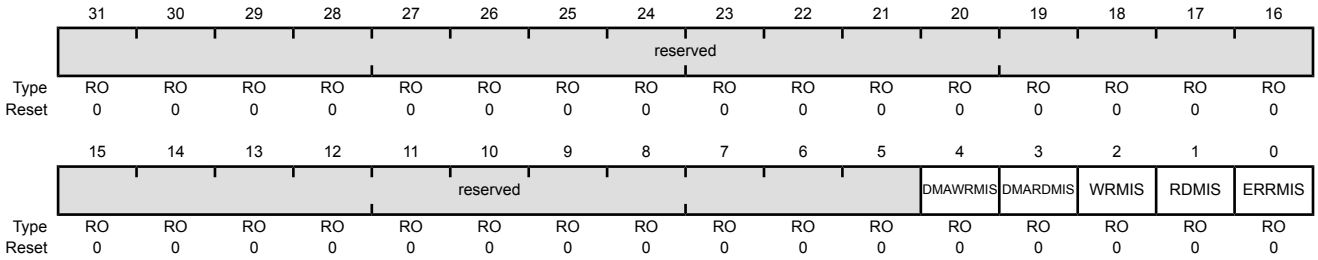
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | WRRIS | RO | 1 | Write Raw Interrupt Status<br><br>Value Description<br>0 The number of available entries in the WFIFO is above the range specified by the WRFIFO field in the **EPIFIFOLVL** register.<br>1 The number of available entries in the WFIFO is within the trigger range specified by the WRFIFO field in the **EPIFIFOLVL** register.<br><br>This bit is cleared when the level in the WFIFO is above the trigger point programmed by the WRFIFO field. |
| 1 | RDRIS | RO | 0 | Read Raw Interrupt Status<br><br>Value Description<br>0 The number of valid entries in the NBRFIFO is below the trigger range specified by the RDFIFO field in the **EPIFIFOLVL** register.<br>1 The number of valid entries in the NBRFIFO is in the trigger range specified by the RDFIFO field in the **EPIFIFOLVL** register.<br><br>This bit is cleared when the level in the NBRFIFO is below the trigger point programmed by the RDFIFO field. |
| 0 | ERRRIS | RO | 0 | Error Raw Interrupt Status<br>The error interrupt occurs in the following situations:<br>■ WFIFO Full. For a full WFIFO to generate an error interrupt, the WFERR bit in the **EPIFIFOLVL** register must be set.<br>■ Read Stalled. For a stalled read to generate an error interrupt, the RSERR bit in the **EPIFIFOLVL** register must be set.<br>■ Timeout. If the MAXWAIT field in the **EPIHBnCFG** register is configured to a value other than 0, a timeout error occurs when XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field.<br><br>Value Description<br>0 An error has not occurred.<br>1 A WFIFO Full, a Read Stalled, or a Timeout error has occurred.<br><br>To determine which error occurred, read the status of the **EPI Error Interrupt Status and Clear (EPIEISC)** register. This bit is cleared by writing a 1 to the bit in the **EPIEISC** register that caused the interrupt. |

## Register 32: EPI Masked Interrupt Status (EPIMIS), offset 0x218

This register is the masked interrupt status register. On read, it gives the current state of each interrupt source (read, write, and error) after being masked via the **EPIIM** register. A write has no effect.

The values returned are the ANDing of the **EPIIM** and **EPIRIS** registers. If a bit is set in this register, the interrupt is sent to the interrupt controller.

EPI Masked Interrupt Status (EPIMIS)
Base 0x400D.0000
Offset 0x218
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMAWRMIS | RO | 0 | Write uDMA Masked Interrupt Status<br><br>Value Description<br>0 The write uDMA has not completed or the interrupt is masked.<br>1 The write uDMA has completed and the DMAWRIM bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1 to the DMAWRIC bit in the **EPIEISC** register. |
| 3 | DMARDMIS | RO | 0 | Read uDMA Masked Interrupt Status<br><br>Value Description<br>0 The read uDMA has not completed or the interrupt is masked.<br>1 The read uDMA has completed and the DMAWRIM bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1 to the DMARDIC bit in the **EPIEISC** register. |

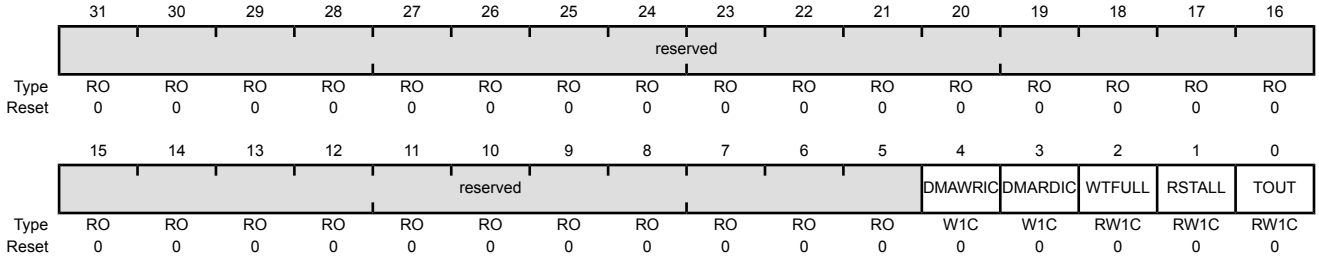
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | WRMIS | RO | 0 | Write Masked Interrupt Status<br>Value Description<br>0 The number of available entries in the WFIFO is above the range specified by the trigger level or the interrupt is masked.<br>1 The number of available entries in the WFIFO is within the range specified by the trigger level (the WRFIFO field in the **EPIFIFOLVL** register) and the WRIM bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller. |
| 1 | RDMIS | RO | 0 | Read Masked Interrupt Status<br>Value Description<br>0 The number of valid entries in the NBRFIFO is below the range specified by the trigger level or the interrupt is masked.<br>1 The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the RDFIFO field in the **EPIFIFOLVL** register) and the RDIM bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller. |
| 0 | ERRMIS | RO | 0 | Error Masked Interrupt Status<br>Value Description<br>0 An error has not occurred or the interrupt is masked.<br>1 A WFIFO Full, a Read Stalled, or a Timeout error has occurred and the ERIM bit in the **EPIIM** register is set, triggering an interrupt to the interrupt controller. |

## Register 33: EPI Error and Interrupt Status and Clear (EPIEISC), offset 0x21C

This register is used to clear a pending error interrupt. Clearing any defined bit in the **EPIEISC** has no effect; setting a bit clears the error source and the raw error returns to 0. When any of bits[2:0] of this register are read as set, it indicates that the ERRRIS bit in the **EPIRIS** register is set and an EPI controller error is sent to the interrupt controller if the ERIM bit in the **EPIIM** register is set. If any of bits [2:0] are written as 1, the register bit being written to, as well as the ERRIS bit in the **EPIRIS** register and the ERIM bit in the **EPIIM** register are cleared.If the DMAWRIC or DMARDIC bit in this register is set, then the corresponding bit in the **EPIRIS** and **EPIMIS** register is cleared. Note that writing to this register and reading back immediately (pipelined by the processor) returns the old register contents. One cycle is needed between write and read.

EPI Error and Interrupt Status and Clear (EPIEISC)

Base 0x400D.0000
Offset 0x21C
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMAWRIC | W1C | 0 | Write uDMA Interrupt Clear<br>Writing a 1 to this bit clears the DMAWRRIS bit in the **EPIRIS** register and the DMAWRMIS bit in the **EPIMIS** register. |
| 3 | DMARDIC | W1C | 0 | Read uDMA Interrupt Clear<br>Writing a 1 to this bit clears the DMARDRIS bit in the **EPIRIS** register and the DMARDMIS bit in the **EPIMIS** register. |
| 2 | WTFULL | RW1C | 0 | Write FIFO Full Error<br>Value Description<br>0 The WFERR bit is not enabled or no writes are stalled.<br>1 The WFERR bit is enabled and a write is stalled due to the WFIFO being full.<br>Writing a 1 to this bit clears it, as well as the ERRRIS and ERIM bits. |
| 1 | RSTALL | RW1C | 0 | Read Stalled Error<br>Value Description<br>0 The RSERR bit is not enabled or no pending reads are stalled.<br>1 The RSERR bit is enabled and a pending read is stalled due to writes in the WFIFO.<br>Writing a 1 to this bit clears it, as well as the ERRRIS and ERIM bits. |

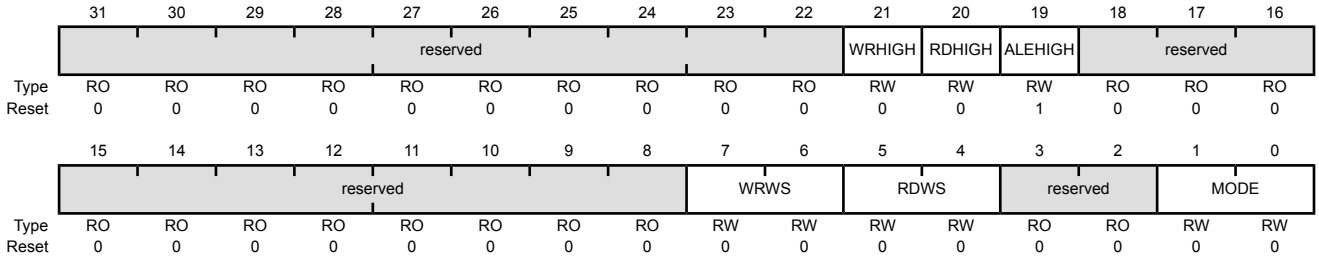
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TOUT | RW1C | 0 | Timeout Error<br>This bit is the timeout error source. The timeout error occurs when the XFIFO not-ready signals hold a transaction for more than the count in the MAXWAIT field (when not 0).<br><br>Value Description<br>0 No timeout error has occurred.<br>1 A timeout error has occurred.<br><br>Writing a 1 to this bit clears it, as well as the ERRRIS and ERIM bits. |

## Register 34: EPI Host-Bus 8 Configuration 3 (EPIHB8CFG3), offset 0x308

**Important:** The MODE field in the **EPICFG** register configures whether EPI Host Bus mode is enabled. For **EPIHB8CFG3** to be valid, the MODE field must be 0x2.

EPI Host-Bus 8 Configuration 3 (EPIHB8CFG3)

Base 0x400D.0000
Offset 0x308
Type RW, reset 0x0008.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:22 | reserved | RO | 0x000 | 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. |
| 21 | WRHIGH | RW | 0 | CS2n WRITE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB8CFG2**.<br><br>Value Description<br>0 The WRITE strobe for CS2n accesses is WRn (active Low).<br>1 The WRITE strobe for CS2n accesses is WR (active High). |
| 20 | RDHIGH | RW | 0 | CS2n READ Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB8CFG2**.<br><br>Value Description<br>0 The READ strobe for CS2n accesses is RDn (active Low).<br>1 The READ strobe for CS2n accesses is RD (active High). |
| 19 | ALEHIGH | RW | 1 | CS2n ALE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB8CFG2**.<br><br>Value Description<br>0 The address latch strobe for CS2n accesses is ADVn (active Low).<br>1 The address latch strobe for CS2n accesses is ALE (active High). |
| 18:8 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | WRWS | RW | 0x0 | CS2n Write Wait States<br>This field adds wait states to the data phase of CS2n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the **EPIHB8TIME3** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the **EPIHB8CFG2** register. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active WRn is 2 EPI clocks<br>0x1 Active WRn is 4 EPI clocks<br>0x2 Active WRn is 6 EPI clocks<br>0x3 Active WRn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 5:4 | RDWS | RW | 0x0 | CS2n Read Wait States<br>This field adds wait states to the data phase of CS2n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB8TIME3** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the **EPIHB8CFG2** register.<br>This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks<br>0x1 Active RDn is 4 EPI clocks<br>0x2 Active RDn is 6 EPI clocks<br>0x3 Active RDn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 3:2 | reserved | RO | 0x0 | 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. |

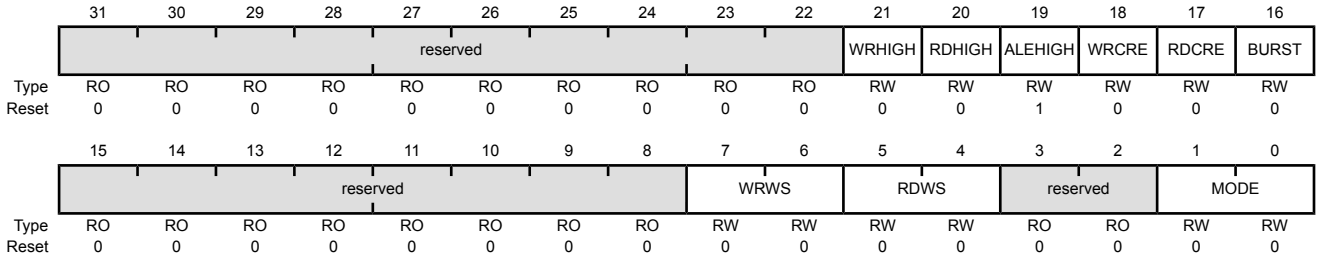
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | MODE | RW | 0x0 | CS2n Host Bus Sub-Mode<br>This field determines which Host Bus 8 sub-mode to use for CS2n in multiple chip-select mode.<br>Sub-mode use is determined by the connected external peripheral. See Table 11-8 on page 864 for information on how this bit field affects the operation of the EPI signals.<br>**Note:** The CSBAUD bit must be set to enable this CS2n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the **EPIHB8CFG** register.<br>Value Description<br>0x0 ADMUX – AD[7:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[7:0]<br>Data and address are separate.<br>0x2-0x3 reserved |

## Register 35: EPI Host-Bus 16 Configuration 3 (EPIHB16CFG3), offset 0x308

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB16CFG3** to be valid, the MODE field must be 0x3.

EPI Host-Bus 16 Configuration 3 (EPIHB16CFG3)

Base 0x400D.0000
Offset 0x308
Type RW, reset 0x0008.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:22 | reserved | RO | 0x000 | 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. |
| 21 | WRHIGH | RW | 0 | CS2n WRITE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB16CFG2**.<br><br>Value Description<br>0 The WRITE strobe for CS2n accesses is WRn (active Low).<br>1 The WRITE strobe for CS2n accesses is WR (active High). |
| 20 | RDHIGH | RW | 0 | CS2n READ Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB16CFG2**.<br><br>Value Description<br>0 The READ strobe for CS2n accesses is RDn (active Low).<br>1 The READ strobe for CS2n accesses is RD (active High). |
| 19 | ALEHIGH | RW | 1 | CS2n ALE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB16CFG2**.<br><br>Value Description<br>0 The address latch strobe for CS2n accesses is ADVn (active Low).<br>1 The address latch strobe for CS2n accesses is ALE (active High). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 18 | WRCRE | RW | 0 | CS2n PSRAM Configuration Register Write<br>Used for PSRAM configuration registers.<br>With WRCRE set, the next transaction by the EPI is a write of the CR bit field in the **EPIHBPSRAM** register to the configuration register (CR) of the PSRAM. The WRCRE bit self clears once the write-enabled CRE access is complete.<br>Value Description<br>0 No Action.<br>1 Start CRE write transaction for CS2n. |
| 17 | RDCRE | RW | 0 | CS2n PSRAM Configuration Register Read<br>Used for PSRAM configuration registers.<br>With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR). This bit self clears once the CRE access is complete. The address for the CRE access is located at **EPIHBPSRAM[19:18]**. The read data is returned on **EPIHBPSRAM[15:0].**<br>Value Description<br>0 No Action.<br>1 Start CRE read transaction for CS2n. |
| 16 | BURST | RW | 0 | CS2n Burst Mode<br>Burst mode must be used with an ALE, which is configured by programming the CSCFG and CSCFGEXT fields in the **EPIHB16CFG2** register. Burst mode must be used in ADMUX, which is set by the MODE field in **EPIHB16CFG3**.<br>**Note:** Burst mode is optimized for word-length accesses.<br>Value Description<br>0 Burst mode is disabled.<br>1 Burst mode is enabled for CS2n. |
| 15:8 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | WRWS | RW | 0x0 | CS2n Write Wait States<br>This field adds wait states to the data phase of CS2n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the **EPIHB16TIME3** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the **EPIHB16CFG2** register. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active WRn is 2 EPI clocks<br>0x1 Active WRn is 4 EPI clocks<br>0x2 Active WRn is 6 EPI clocks<br>0x3 Active WRn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 5:4 | RDWS | RW | 0x0 | CS2n Read Wait States<br>This field adds wait states to the data phase of CS2n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB16TIME3** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the **EPIHB16CFG2** register.<br>This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks<br>0x1 Active RDn is 4 EPI clocks<br>0x2 Active RDn is 6 EPI clocks<br>0x3 Active RDn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 3:2 | reserved | RO | 0 | 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. |

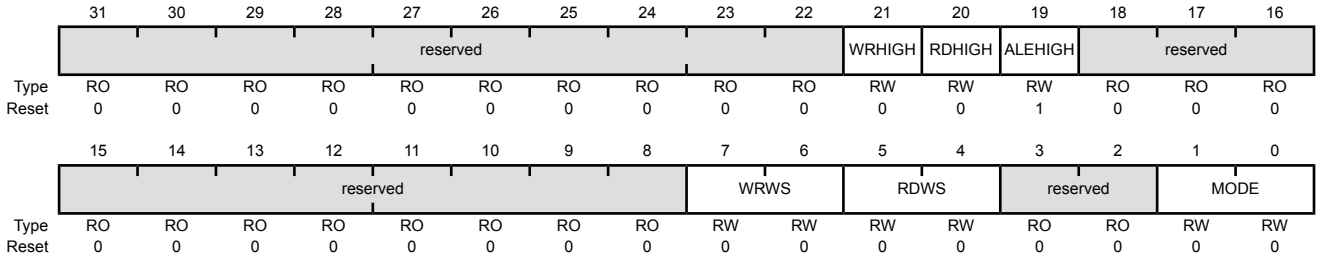
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | MODE | RW | 0x0 | CS2n Host Bus Sub-Mode<br>This field determines which Host Bus 16 sub-mode to use for CS2n in multiple chip select mode.<br>Sub-mode use is determined by the connected external peripheral. See Table 11-9 on page 866 for information on how this bit field affects the operation of the EPI signals.<br>**Note:** The CSBAUD bit must be set to enable this CS2n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the **EPIHB16CFG** register.<br>Value Description<br>0x0 ADMUX – AD[15:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[15:0]<br>Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.<br>0x2-0x3 reserved |

## Register 36: EPI Host-Bus 8 Configuration 4 (EPIHB8CFG4), offset 0x30C

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB8CFG4** to be valid, the MODE field must be 0x2.

EPI Host-Bus 8 Configuration 4 (EPIHB8CFG4)

Base 0x400D.0000
Offset 0x30C
Type RW, reset 0x0008.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:22 | reserved | RO | 0x0 | 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. |
| 21 | WRHIGH | RW | 0 | CS3n WRITE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB8CFG2**.<br><br>Value Description<br>0 The WRITE strobe for CS3n accesses is WRn (active Low).<br>1 The WRITE strobe for CS3n accesses is WR (active High). |
| 20 | RDHIGH | RW | 0 | CS2n READ Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB8CFG2.**<br><br>Value Description<br>0 The READ strobe for CS3n accesses is RDn (active Low).<br>1 The READ strobe for CS3n accesses is RD (active High). |
| 19 | ALEHIGH | RW | 1 | CS3n ALE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB8CFG2**<br><br>Value Description<br>0 The address latch strobe for CS3n accesses is ADVn (active Low).<br>1 The address latch strobe for CS3n accesses is ALE (active High). |
| 18:8 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | WRWS | RW | 0x0 | CS3n Write Wait States<br>This field adds wait states to the data phase of CS3n accesses (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the **EPIHB8TIME4** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is enabled in the **EPIHB8CFG2** register. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active WRn is 2 EPI clocks<br>0x1 Active WRn is 4 EPI clocks<br>0x2 Active WRn is 6 EPI clocks<br>0x3 Active WRn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 5:4 | RDWS | RW | 0x0 | CS3n Read Wait States<br>This field adds wait states to the data phase of CS3n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB8TIME4** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used when the CSBAUD bit is set in the **EPIHB8CFG2** register.<br>This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks<br>0x1 Active RDn is 4 EPI clocks<br>0x2 Active RDn is 6 EPI clocks<br>0x3 Active RDn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 3:2 | reserved | RO | 0x0 | 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. |

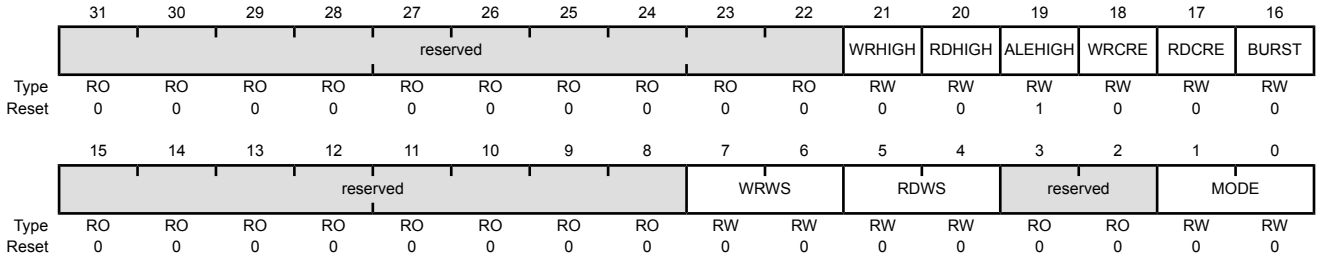
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | MODE | RW | 0x0 | CS3n Host Bus Sub-Mode<br>This field determines which Host Bus 8 sub-mode to use for CS3n in multiple chip select mode. Sub-mode use is determined by the connected external peripheral. See Table 11-8 on page 864 for information on how this bit field affects the operation of the EPI signals.<br>**Note:** The CSBAUD bit must be set to enable this CS3n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the **EPIHB8CFG** register.<br>Value Description<br>0x0 ADMUX – AD[7:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[7:0]<br>Data and address are separate.<br>0x2-0x3 reserved |

## Register 37: EPI Host-Bus 16 Configuration 4 (EPIHB16CFG4), offset 0x30C

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB16CFG4** to be valid, the MODE field must be 0x3.

EPI Host-Bus 16 Configuration 4 (EPIHB16CFG4)

Base 0x400D.0000
Offset 0x30C
Type RW, reset 0x0008.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:22 | reserved | RO | 0x00 | 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. |
| 21 | WRHIGH | RW | 0 | CS3n WRITE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB16CFG2**.<br><br>Value Description<br>0 The WRITE strobe for CS3n accesses is WRn (active Low).<br>1 The WRITE strobe for CS3n accesses is WR (active High). |
| 20 | RDHIGH | RW | 0 | CS3n READ Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB16CFG2**.<br><br>Value Description<br>0 The READ strobe for CS3n accesses is RDn (active Low).<br>1 The READ strobe for CS3n accesses is RD (active High). |
| 19 | ALEHIGH | RW | 1 | CS3n ALE Strobe Polarity<br>This field is used if the CSBAUD bit is enabled in **EPIHB16CFG2**.<br><br>Value Description<br>0 The address latch strobe for CS3n accesses is ADVn (active Low).<br>1 The address latch strobe for CS3n accesses is ALE (active High). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 18 | WRCRE | RW | 0 | CS3n PSRAM Configuration Register Write<br>Used for PSRAM configuration registers.<br>With WRCRE set, the next transaction by the EPI will be a write of the CR bit field in the **EPIHBPSRAM** register to the configuration register (CR) of the PSRAM. The WRCRE bit will self clear once the write-enabled CRE access is complete.<br>Value Description<br>0 No Action.<br>1 Start CRE write transaction for CS3n. |
| 17 | RDCRE | RW | 0 | CS3n PSRAM Configuration Register Read<br>Used for PSRAM configuration registers.<br>With the RDCRE set, the next access is a read of the PSRAM's Configuration Register (CR). This bit self clears once the CRE access is complete. The address for the CRE access is located at **EPIHBPSRAM[19:18]**. The read data is returned on **EPIHBPSRAM[15:0].**<br>Value Description<br>0 No Action.<br>1 Start CRE read transaction for CS3n. |
| 16 | BURST | RW | 0 | CS3n Burst Mode<br>Burst mode must be used with an ALE, which is configured by programming the CSCFG and CSCFGEXT fields in the **EPIHB16CFG2** register. Burst mode must be used in ADMUX, which is set by the MODE field in **EPIHB16CFG4**.<br>**Note:** Burst mode is optimized for word-length accesses.<br>Value Description<br>0 Burst mode is disabled.<br>1 Burst mode is enabled for CS3n. |
| 15:8 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | WRWS | RW | 0x0 | CS3n Write Wait States<br>This field adds wait states to the data phase of CS2n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of WRn (or the falling edge of WR). Each wait state adds 2 EPI clock cycles to the access time. The WRWSM bit in the **EPIHB16TIME4** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used if the CSBAUD bit is set in the **EPIHB16CFG2** register. This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active WRn is 2 EPI clocks<br>0x1 Active WRn is 4 EPI clocks<br>0x2 Active WRn is 6 EPI clocks<br>0x3 Active WRn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 5:4 | RDWS | RW | 0x0 | CS3n Read Wait States<br>This field adds wait states to the data phase of CS3n accesses (the address phase is not affected).<br>The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD). Each wait state adds 2 EPI clock cycles to the access time. The RDWSM bit in the **EPIHB16TIME4** register can decrease the number of wait states by 1 EPI clock cycle for greater granularity. This field is used when the CSBAUD bit is set in the **EPIHB16CFG2** register.<br>This field is not applicable in BURST mode.<br>Value Description<br>0x0 Active RDn is 2 EPI clocks<br>0x1 Active RDn is 4 EPI clocks<br>0x2 Active RDn is 6 EPI clocks<br>0x3 Active RDn is 8 EPI clocks<br>This field is used in conjunction with the **EPIBAUD2** register. |
| 3:2 | reserved | RO | 0 | 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. |

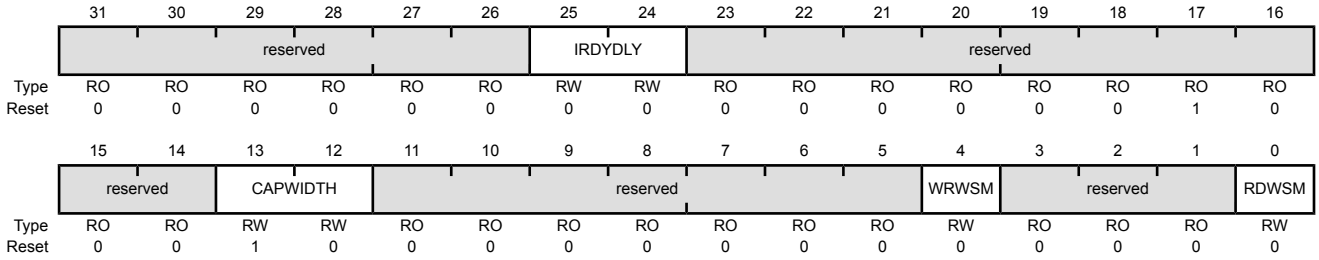
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | MODE | RW | 0x0 | CS3n Host Bus Sub-Mode<br>This field determines which Host Bus 16 sub-mode to use for CS3n in multiple chip select mode.<br>Sub-mode use is determined by the connected external peripheral. See Table 11-9 on page 866 for information on how this bit field affects the operation of the EPI signals.<br>**Note:** The CSBAUD bit must be set to enable this CS3n MODE field. If CSBAUD is clear, all chip-selects use the MODE configuration defined in the **EPIHB16CFG** register.<br>Value Description<br>0x0 ADMUX – AD[15:0]<br>Data and Address are muxed.<br>0x1 ADNONMUX – D[15:0]<br>Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.<br>0x2-0x3 reserved |

## Register 38: EPI Host-Bus 8 Timing Extension (EPIHB8TIME), offset 0x310

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB8TIME** to be valid, the MODE field must be 0x2.

EPI Host-Bus 8 Timing Extension (EPIHB8TIME)

Base 0x400D.0000
Offset 0x310
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS0n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:14 | reserved | RO | 0x008 | 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. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS0n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br><br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |
| 11:5 | reserved | RO | 0x00 | 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. |

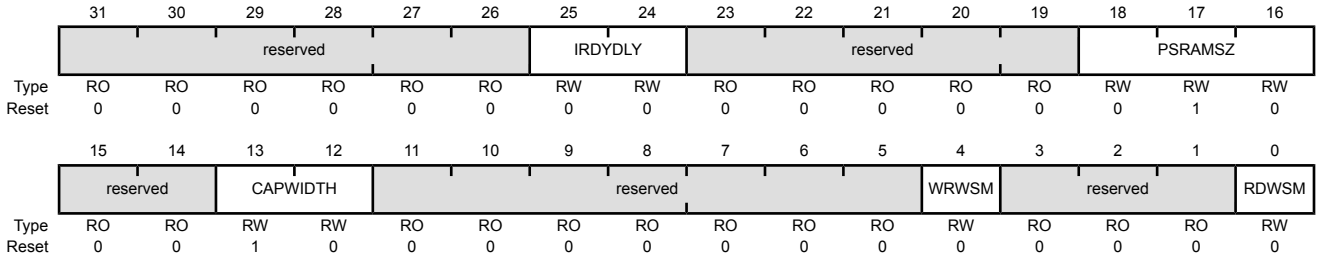
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | WRWSM | RW | 0 | Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB8CFG**. This field is not applicable in BURST mode.<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB8CFG** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB8CFG**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | Read Wait State Minus One<br>Use with RDWS field in the **EPIHB8CFG** register. This field is not applicable in BURST mode.<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB8CFG**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB8CFG**. |

## Register 39: EPI Host-Bus 16 Timing Extension (EPIHB16TIME), offset 0x310

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB16TIME** to be valid, the MODE field must be 0x3.

EPI Host-Bus 16 Timing Extension (EPIHB16TIME)

Base 0x400D.0000
Offset 0x310
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS0n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:19 | reserved | RO | 0x000 | 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. |
| 18:16 | PSRAMSZ | RW | 0x2 | PSRAM Row Size<br>Defines the row size for the PSRAM controlled by CS0n<br><br>Value Description<br>0x0 No row size limitation<br>0x1 128 B<br>0x2 256 B<br>0x3 512 B<br>0x4 1024 B<br>0x5 2048 B<br>0x6 4096 B<br>0x7 8192 B |

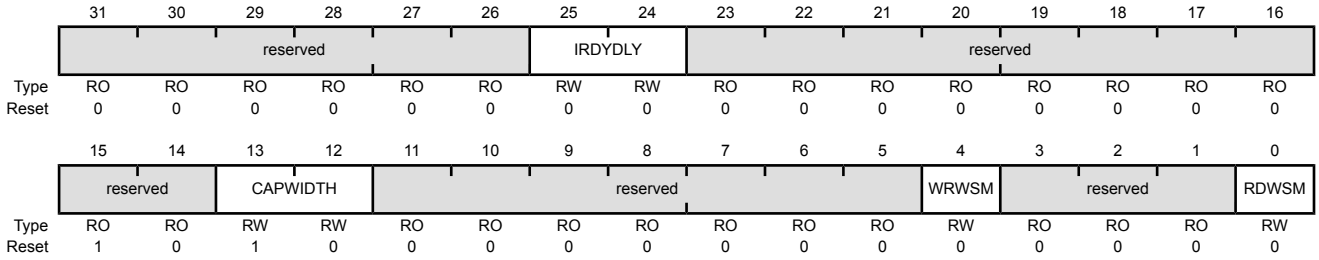
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS0n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |
| 11:5 | reserved | RO | 0x0 | 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. |
| 4 | WRWSM | RW | 0 | Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB16CFG**. This field is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB16CFG** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB16CFG**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | Read Wait State Minus One<br>Use with RDWS field in the **EPIHB16CFG** register. This field is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB16CFG**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB16CFG**. |

## Register 40: EPI Host-Bus 8 Timing Extension (EPIHB8TIME2), offset 0x314

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB8TIME2** to be valid, the MODE field must be 0x2.

EPI Host-Bus 8 Timing Extension (EPIHB8TIME2)

Base 0x400D.0000
Offset 0x314
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS1n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:14 | reserved | RO | 0x002 | 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. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS1n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br><br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |
| 11:5 | reserved | RO | 0x00 | 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. |

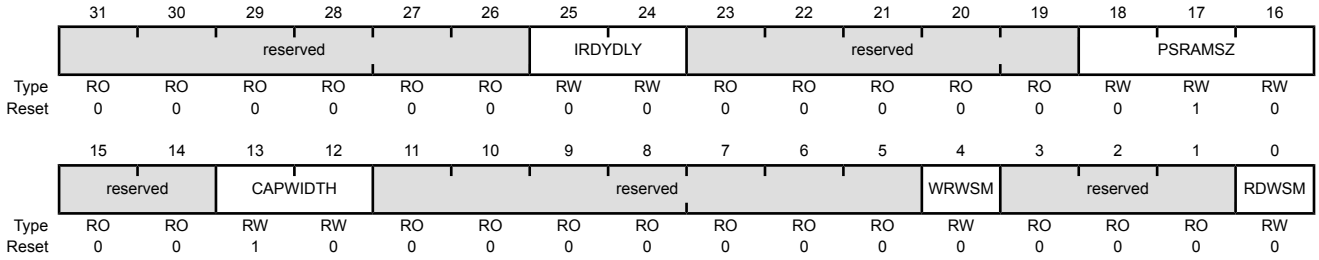
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | WRWSM | RW | 0 | CS1n Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB8CFG2**. This field is not applicable in BURST mode.<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB8CFG2** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB8CFG2**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | CS1n Read Wait State Minus One<br>This field is used with RDWS field in **EPIHB8CFG2**. This bit is not applicable in BURST mode.<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB8CFG2**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB8CFG2**. |

## Register 41: EPI Host-Bus 16 Timing Extension (EPIHB16TIME2), offset 0x314

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB16TIME2** to be valid, the MODE field must be 0x3.

EPI Host-Bus 16 Timing Extension (EPIHB16TIME2)

Base 0x400D.0000
Offset 0x314
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS1n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:19 | reserved | RO | 0x000 | 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. |
| 18:16 | PSRAMSZ | RW | 0x2 | PSRAM Row Size<br>Defines the row size for the PSRAM controlled by CS1n<br><br>Value Description<br>0x0 No row size limitation<br>0x1 128 B<br>0x2 256 B<br>0x3 512 B<br>0x4 1024 B<br>0x5 2048 B<br>0x6 4096 B<br>0x7 8192 B |

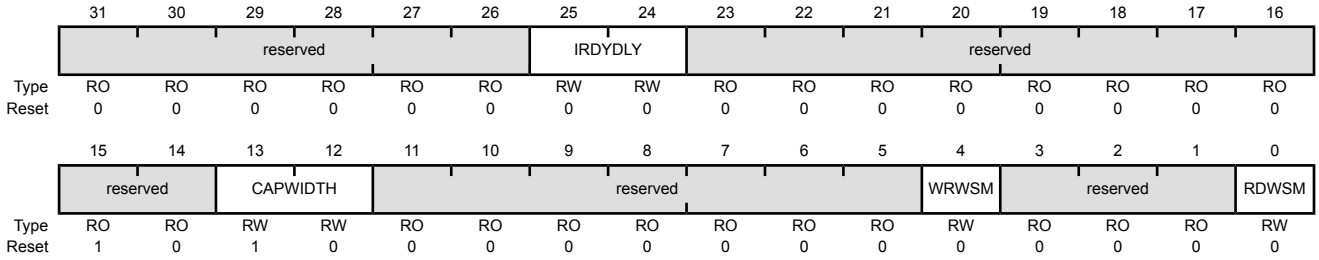
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS1n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br><br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |
| 11:5 | reserved | RO | 0x00 | 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. |
| 4 | WRWSM | RW | 0 | CS1n Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB16CFG2**. This field is not applicable in BURST mode..<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB16CFG2** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB16CFG2**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | CS1n Read Wait State Minus One<br>This field is used with RDWS field in **EPIHB16CFG2**. This bit is not applicable in BURST mode.<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB16CFG2**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB16CFG2**. |

## Register 42: EPI Host-Bus 8 Timing Extension (EPIHB8TIME3), offset 0x318

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB8TIME**3 to be valid, the MODE field must be 0x2.

EPI Host-Bus 8 Timing Extension (EPIHB8TIME3)

Base 0x400D.0000
Offset 0x318
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS2n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:14 | reserved | RO | 0x002 | 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. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS2n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br><br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |
| 11:5 | reserved | RO | 0x00 | 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. |

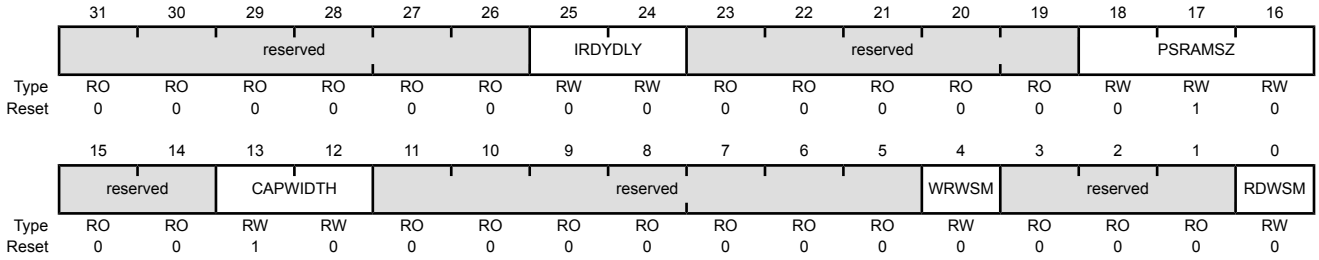
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | WRWSM | RW | 0 | CS2n Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB8CFG3**. This field is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB8CFG3** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB8CFG3**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | CS2n Read Wait State Minus One<br>This field is used with RDWS field in **EPIHB8CFG3**. This bit is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB8CFG3**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB8CFG3**. |

## Register 43: EPI Host-Bus 16 Timing Extension (EPIHB16TIME3), offset 0x318

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB16TIME**3 to be valid, the MODE field must be 0x3.

EPI Host-Bus 16 Timing Extension (EPIHB16TIME3)

Base 0x400D.0000
Offset 0x318
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS2n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:19 | reserved | RO | 0x000 | 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. |
| 18:16 | PSRAMSZ | RW | 0x2 | PSRAM Row Size<br>Defines the row size for the PSRAM controlled by CS2n<br><br>Value Description<br>0x0 No row size limitation<br>0x1 128 B<br>0x2 256 B<br>0x3 512 B<br>0x4 1024 B<br>0x5 2048 B<br>0x6 4096 B<br>0x7 8192 B |

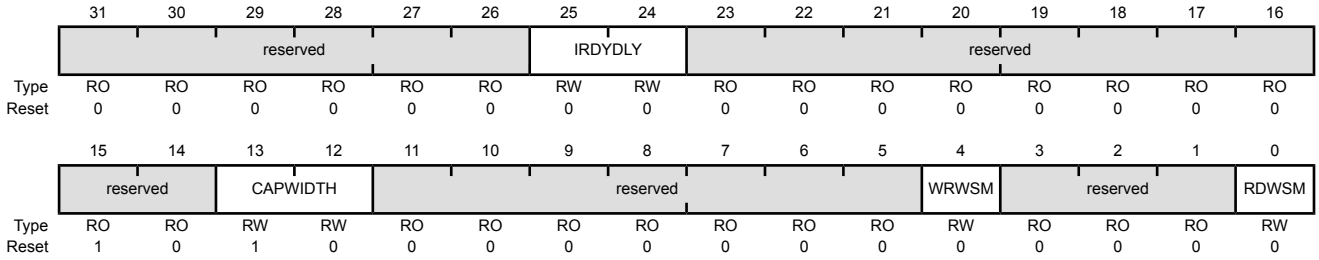
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS2n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |
| 11:5 | reserved | RO | 0x00 | 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. |
| 4 | WRWSM | RW | 0 | CS2n Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB16CFG3**. This field is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB16CFG3** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB16CFG3**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | CS2n Read Wait State Minus One<br>This field is used with RDWS field in **EPIHB16CFG3**. This bit is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB16CFG3**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB16CFG3**. |

## Register 44: EPI Host-Bus 8 Timing Extension (EPIHB8TIME4), offset 0x31C

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB8TIME4** to be valid, the MODE field must be 0x2.

EPI Host-Bus 8 Timing Extension (EPIHB8TIME4)

Base 0x400D.0000
Offset 0x31C
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS3n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:14 | reserved | RO | 0x002 | 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.<br>Bits [18:16] have the same RTL implementation as the **HB16TIMEn** register, even though this is not used in HB8 mode. Thus, the reset value of 0x2 is carried over from the PSRAMSZ bits of **HB16TIMEn**. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS3n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br><br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |

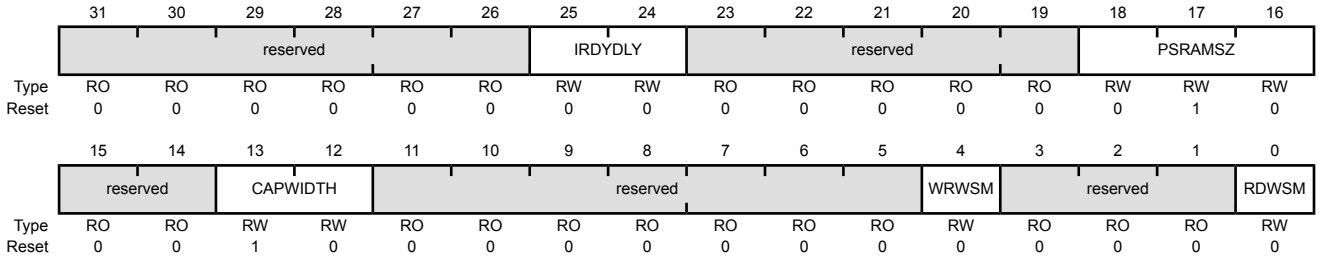
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:5 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | WRWSM | RW | 0 | CS3n Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB8CFG4**. This field is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB8CFG4** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB8CFG4**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | CS3n Read Wait State Minus One<br>This field is used with RDWS field in **EPIHB8CFG4**. This bit is not applicable in BURST mode.<br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB8CFG4**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB8CFG4**. |

## Register 45: EPI Host-Bus 16 Timing Extension (EPIHB16TIME4), offset 0x31C

**Important:** The MODE field in the **EPICFG** register determines which configuration is enabled. For **EPIHB16TIME**4 to be valid, the MODE field must be 0x3.

EPI Host-Bus 16 Timing Extension (EPIHB16TIME4)

Base 0x400D.0000
Offset 0x31C
Type RW, reset 0x0002.2000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25:24 | IRDYDLY | RW | 0x0 | CS3n Input Ready Delay<br><br>Value Description<br>0 reserved<br>1 Stall begins one EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>2 Stall begins two EPI clocks past iRDY low being sampled on the rising edge of EPIO clock.<br>3 Stall begins three EPI clocks past iRDY low being sampled on the rising edge of EPIO clock. |
| 23:19 | reserved | RO | 0x000 | 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. |
| 18:16 | PSRAMSZ | RW | 0x2 | PSRAM Row Size<br>Defines the row size for the PSRAM controlled by CS3n<br><br>Value Description<br>0x0 No row size limitation<br>0x1 128 B<br>0x2 256 B<br>0x3 512 B<br>0x4 1024 B<br>0x5 2048 B<br>0x6 4096 B<br>0x7 8192 B |

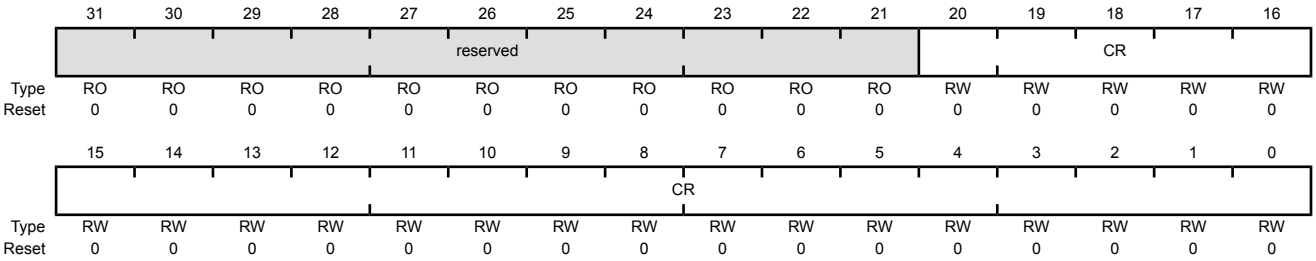
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 13:12 | CAPWIDTH | RW | 0x2 | CS3n Inter-transfer Capture Width<br>Controls the delay between Host-Bus transfers.<br><br>Value Description<br>0x0 Reserved<br>0x1 1 EPI clock.<br>0x2 2 EPI clock.<br>0x3 Reserved |
| 11:5 | reserved | RO | 0x00 | 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. |
| 4 | WRWSM | RW | 0 | CS3n Write Wait State Minus One<br>This bit is used with the WRWS field in **EPIHB16CFG4**. This field is not applicable in BURST mode.<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the in WRWS field in **EPIHB16CFG4** register.<br>1 Wait state value is now:<br>WRWS - 1<br>WRWS field is programmed in **EPIHB16CFG4**. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | RDWSM | RW | 0 | CS3n Read Wait State Minus One<br>This field is used with RDWS field in **EPIHB16CFG4**. This bit is not applicable in BURST mode.<br><br>Value Description<br>0 No change in the number of wait state clock cycles programmed in the RDWS field of **EPIHB16CFG4**.<br>1 Wait state value is now:<br>RDWS - 1<br>RDWS field is programmed in **EPIHB16CFG4**. |

## Register 46: EPI Host-Bus PSRAM (EPIHBPSRAM), offset 0x360

This register holds the PSRAM configuration register value. When the WRCRE bit in the **EPIHB16CFGn** register is set, all 21 bits of the **EPIHBPSRAM** register's CR value are written to the PSRAM's configuration register. When the RDCRE bit is set in the **EPIHB16CFGn** register, a read of the PSRAM's configuration register takes place and the value is written to bits[15:0] of the **EPIHBPSRAM**. Bits[20:16] will not contain any valid data.

EPI Host-Bus PSRAM (EPIHBPSRAM)

Base 0x400D.0000
Offset 0x360
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:21 | reserved | RO | 0x000 | 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. |
| 20:0 | CR | RW | 0x000000 | PSRAM Config Register<br>During a configuration write, all 21 bits of the CR bit field are written to the PSRAM. During configuration reads, CR bits[15:0] of this register contain the configuration read of the PSRAM. CR[20:16] will not contain valid data. |

# 12 Cyclical Redundancy Check (CRC)

The Cyclical Redundancy Check (CRC) computation module can be used for message transfer and safety system checks, as well as in conjunction with the AES and DES modules. The following features are supported:

- Support four major CRC forms:
  - CRC16-CCITT as used by CCITT/ITU X.25
  - CRC16-IBM as used by USB and ANSI
  - CRC32-IEEE as used by IEEE802.3 and MPEG2
  - CRC32C as used by G.Hn
- Allows word and byte feed
- Supports auto-initialization and manual initialization
- Supports MSb and LSb
- Supports CCITT post-processing
- Can be fed by µDMA, Flash memory and code

## 12.1 Functional Description

The following sections describe the features of CRC.

### 12.1.1 CRC Support

The purpose of the CRC engine is to accelerate CRC and TCP checksum operation. The result of the CRC operation is a 32- and 16-bit signature which can be used to check the sanity of data. The required mode of operation is selected through the TYPE bit in the **CRC Control (CRCCTRL)** register, offset 0x400. A µDMA software channel can be used to burst data into the CRC module. CRCs are computed combinatorially in one clock.

The CRC module contains all of the control registers to which the input context interfaces. Because CRC calculations are a single cycle, as soon as data is written to **CRC Data Input (CRCDIN)** register, the result of CRC/CSUM is updated in the **CRC SEED/Context (CRCSEED)** register, offset 0x410. The input data is computed by the selected CRC polynomial or CSUM.

#### 12.1.1.1 CRC Checksum engine

Software can offload the CRC and checksum task to the CRC checksum engine accelerator. The accelerator has registers that need to be programmed to initiate processing. These registers should be fed with data in order to calculate CRC/CSUM. Software should configure the µDMA channel for data movement through the **DMA Channel Map Select n (DMACHMAPn)** register in the µDMA module. Further µDMA configuration guidelines are available in the “Micro Direct Memory Access (µDMA)” on page 708.

The starting seed for the CRC and checksum operation is programmed in the **CRC SEED/Context (CRCSEED)** register at offset 0x410. Depending on the encoding of the INIT field in the **CRCCTRL** register, the value of the SEED field can initialized to any one of the following:

- A unique context value written to the **CRCSEED** register (INIT=0x0)
- All 0s (INIT=0x2)
- All 1s (INIT=0x3)

Once the operation is done, software should read the result from the **CRC Post Processing Result (CRCRSLTPP)** register, offset 0x418, and a software channel µDMA interrupt should be used to identify completion.

#### 12.1.1.2 Data Size

The CRC module supports data being fed 32-bit words and 8 bits at a time and can dynamically switch back and forth. The data size is configured by programming the SIZE bit in the **CRCCTRL** register, offset 0x400.

Because CRC is a division on a long stream of bits, the application must take into consideration the bit order. When processing message data that is read out by words, bit order is not an issue. For example, if the data value in the message is 0x12345678, the most significant eight byte is 0x12 (00010010 in binary). If the data is processed as bytes, 0x12, 0x34, 0x56, and 0x78 are copied into memory in that order and the word is stored as 0x78563412, where 0x12 is written as byte 0, 0x34 is written as byte 1, and so on.

#### 12.1.1.3 Endian Configuration

The following endian configuration is provided by the ENDIAN field in the **CRCCTRL** register:

- Swap byte in half-word
- Swap half word

Input data width is four bytes, hence the configuration only affects the four-byte word. The ENDIAN bit field supports the following configurations, assuming the input word is {B3, B2, B1, B0}

**Table 12-1. Endian Configuration**

| ENDIAN Encoding | Definition | Configuration |
|---|---|---|
| 0x0 | Configuration unchanged. | {B3, B2, B1, B0} |
| 0x1 | Bytes are swapped in half-words but half-words are not swapped | {B2, B3, B0, B1} |
| 0x2 | Half-words are swapped but bytes are not swapped in half-word. | {B1, B0, B3, B2} |
| 0x3 | Bytes are swapped in half-words and half-words are swapped. | {B0, B1, B2, B3} |

Bit reversal is supported by the BR bit in the **CRCCTRL** register. The bit reversal operation works in tandem with endian control. For example, the above table with the BR option set would look like this:

**Table 12-2. Endian Configuration with Bit Reversal**

| ENDIAN Encoding | Initial Endian Configuration | Configuration with Bit Reversal (BR = 1) |
|---|---|---|
| 0x0 | Configuration unchanged.<br>{B3[31:24], B2[23:16] , B1[15:8], B0[7:0]} | B3[24:31],B2[16:23],B1[8:15],B0[0:7] |

**Table 12-2. Endian Configuration with Bit Reversal *(continued)***

| ENDIAN Encoding | Initial Endian Configuration | Configuration with Bit Reversal (BR = 1) |
|---|---|---|
| 0x1 | Bytes are swapped in half-words but half-words are not swapped<br>{B2[23:16], B3[31:24], B0[7:0], B1[15:8]} | B2[16:23],B3[24:31],B0[0:7],B1[8:15] |
| 0x2 | Half-words are swapped but bytes are not swapped in half-word.<br>{B1[15:8], B0[7:0], B3[31:24], B2[23:16]} | B1[8:15],B0[0:7],B3[24:31],B2[16:23] |
| 0x3 | Bytes are swapped in half-words and half-words are swapped<br>{B0[7:0], B1[15:8], B2[23:16], B3[31:24]} | B0[0:7],B1[8:15],B2[16:23],B3[24:31] |

## 12.2 Initialization and Configuration

The following describes the initialization and configuration procedures of the CRC module.

### 12.2.1 CRC Initialization and Configuration

The CRC engine works in push through mode, which means it works on streaming data. This section describes the steps for initializing the CRC module:

1. Enable the CRC by setting the R0 bit in the CRC and Cryptographic Modules (RCGCCM) register, System Control offset 0x674.
2. Configure the desired CRC data size, bit order, endian configuration and CRC type by programming the **CRC Control (CRCCTRL)** register, offset 0x400.
3. If the CRC value has not been initialized to all 0s or all 1s using the INIT field in the **CRCCTRL** register, program the initial value in the **CRC SEED/Context (CRCSEED)** register, offset 0x410.
4. Repeatedly write the DATAIN field in the **CRC Data Input (CRCDIN)** register, offset 0x414. If the SIZE bit in the **CRCCTRL** register is set to select byte, the CRC engine operates in byte mode and only the least significant byte is used for CRC calculation.
5. When CRC is finished, read the **CRCSEED** register for the final result. If using post-processing, the raw CRC result is stored in the **CRCSEED** register and the final, post-processed result can be read from the **CRC Post-Processing Result (CRCRSLTPP)** register, offset 0x418. Post-processing options are selectable through the OBR and OLNV bits of the **CRCCTRL** register.

Alternatively a software µDMA channel can be configured to copy data from the source into the **CRCDIN** register. When configuring the µDMA, the destination should be configured to not increment. For more information on how to configure the µDMA, refer to “Micro Direct Memory Access (µDMA)” on page 708.

#### 12.2.1.1 Data Endian Convention for the CRC Engine

If the input stream is expressed as a byte stream, Din, where Din = {D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D11, D12, D13, D14, D15, D16.....}, then data should be fed to the CRC engine as follows:

- If operating in Byte mode, the **CRCDIN** register should be written in the following order:
  1. {00, 00, 00, D0}

2. {00, 00, 00, D1}
3. {00, 00, 00, D2}
4. {00, 00, 00, D3}
5. {00, 00, 00, D4}
6. {00, 00, 00, D5}
7. {00, 00, 00, D6}
8. .....
9. .....

- If operating in word mode, the **CRCDIN** register should be written in the following order:
  1. {D3, D2, D1, D0}
  2. {D7, D6, D5, D4}
  3. {D11, D10, D9, D8}
  4. .....
  5. .....

## 12.3 Register Map

Table 12-3 on page 982 lists the CRC Module registers. The offset listed is a hexadecimal increment to the register's address, relative to the base address 0x4403.0000.

**Table 12-3. CCM Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x400 | CRCCTRL | RW | 0x0000.0000 | CRC Control | 983 |
| 0x410 | CRCSEED | RW | 0x0000.0000 | CRC SEED/Context | 985 |
| 0x414 | CRCDIN | RW | 0x0000.0000 | CRC Data Input | 986 |
| 0x418 | CRCRSLTPP | RO | 0x0000.0000 | CRC Post Processing Result | 987 |

## 12.4 CRC Module Register Descriptions

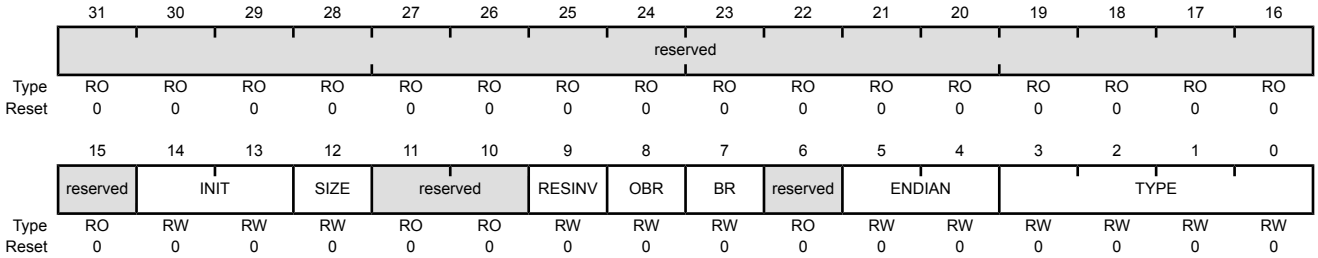
This section lists and describes the CRC registers, in numerical order by address offset.

**Note:** The CRC module can only be accessed through privileged mode. If the µDMA is used for CRC transfers, then the µDMA's **DMA Channel Control (DMACHCTL)** register also needs to be programmed to allow for privileged accesses.

## Register 1: CRC Control (CRCCTRL), offset 0x400

The **CRC Control (CRCCTRL)** register is used to configure control of the CRC.

CRC Control (CRCCTRL)
Base 0x4403.0000
Offset 0x400
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:15 | reserved | RO | 0x0000 | 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. |
| 14:13 | INIT | RW | 0x0 | CRC Initialization<br>Determines initialization value of CRC. This field is self-clearing. With the first write to the **CRC Data Input (CRCDIN)** register, this value clears to zero and remains zero for the rest of the operation unless written again.<br>Value Description<br>0x0 Use the **CRCSEED** register context as the starting value<br>0x1 reserved<br>0x2 Initialize to all '0s'<br>0x3 Initialize to all '1s' |
| 12 | SIZE | RW | 0 | Input Data Size<br>Value Description<br>0 32-bit (word)<br>1 8-bit (byte) |
| 11:10 | reserved | RO | 0x0 | 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. |
| 9 | RESINV | RW | 0 | Result Inverse Enable<br>Value Description<br>0 No effect<br>1 Invert the result bits before storing in the **CRCRSLTPP** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | OBR | RW | 0 | Output Reverse Enable<br>Refer to Table 12-2 on page 980 for more information regarding bit reversal.<br>Value Description<br>0 No change to result.<br>1 Bit reverse the output result byte before storing to **CRCRSLTPP** register. The reversal is applied to all bytes in a word. |
| 7 | BR | RW | 0 | Bit reverse enable<br>Refer to Table 12-2 on page 980 for more information regarding bit reversal.<br>Value Description<br>0 No change to result.<br>1 Bit reverse the input byte for all bytes in a word. |
| 6 | reserved | RO | 0 | 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 | ENDIAN | RW | 0 | Endian Control<br>This field is used to program the endian configuration. The encodings below are with respect to an input word = (B3, B2, B1, B0)<br>Refer to Table 12-1 on page 980 for more information regarding endian configuration and control.<br>Value Description<br>0x0 Configuration unchanged. (B3, B2, B1, B0)<br>0x1 Bytes are swapped in half-words but half-words are not swapped (B2, B3, B0, B1)<br>0x2 Half-words are swapped but bytes are not swapped in half-word. (B1, B0, B3, B2)<br>0x3 Bytes are swapped in half-words and half-words are swapped. (B0, B1, B2, B3) |
| 3:0 | TYPE | RW | 0 | Operation Type<br>The TYPE value in the **CRCCTRL** register should be exclusive.<br>Value Description<br>0x0 Polynomial 0x8005<br>0x1 Polynomial 0x1021<br>0x2 Polynomial 0x4C11DB7<br>0x3 Polynomial 0x1EDC6F41<br>0x4-0x7 reserved<br>0x8 TCP checksum<br>0x9-0xF reserved |

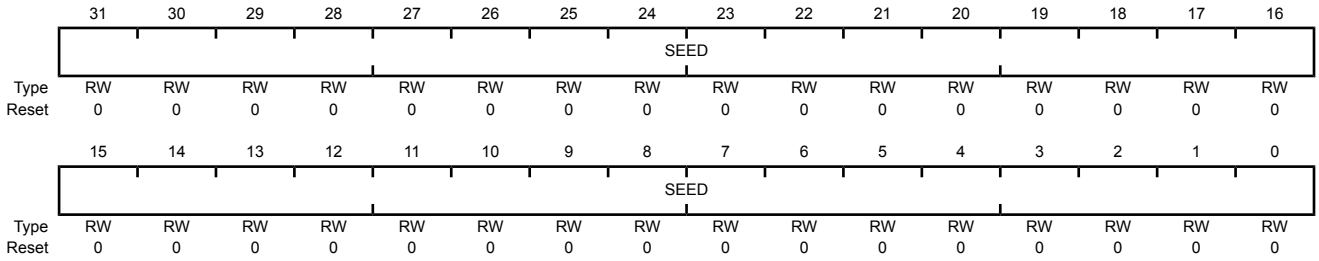
## Register 2: CRC SEED/Context (CRCSEED), offset 0x410

The **CRC SEED/Context (CRCSEED)** register is initially written with one of the following three values depending on the encoding of the INIT field in the **CRCCTRL** register:

- The context value written to the **CRCSEED** register. This encoding is for SEED values from a previous CRC calculation or a specific protocol. (INIT=0x0)
- 0x0000.0000 (INIT=0x2)
- 0x1111.1111 (INIT=0x3)

CRC SEED/Context (CRCSEED)

Base 0x4403.0000
Offset 0x410
Type RW, reset 0x0000.0000



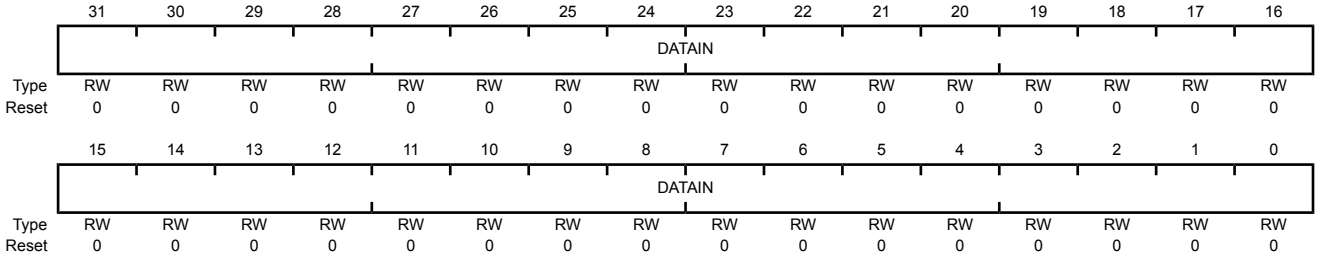

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SEED | RW | 0x0000.0000 | SEED/Context Value<br>This register contains the starting seed of the CRC and checksum operation. This register also holds the latest result of CRC or checksum operation. |

## Register 3: CRC Data Input (CRCDIN), offset 0x414

The application or µDMA writes the **CRC Data Input (CRCDIN)** register with the next byte or word to compute.

CRC Data Input (CRCDIN)

Base 0x4403.0000
Offset 0x414
Type RW, reset 0x0000.0000



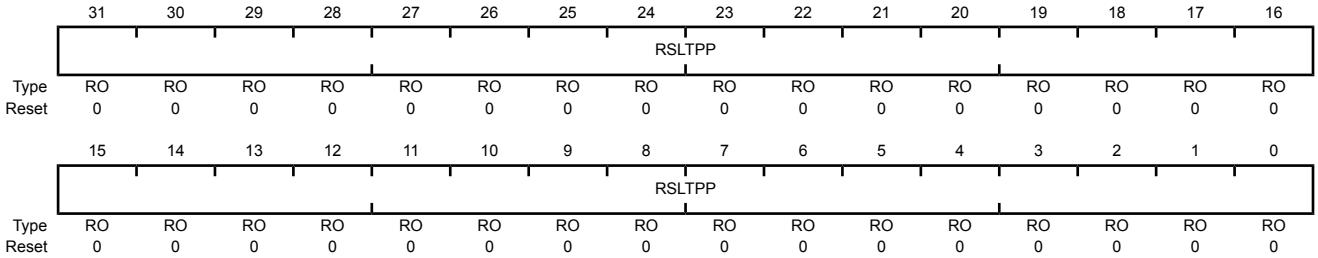

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATAIN | RW | 0x0000.0000 | Data Input<br>This register contains the input data value for the CRC or checksum operation. |

## Register 4: CRC Post Processing Result (CRCRSLTPP), offset 0x418

This register contains the post-processed CRC result as configured by the **CRCCTRL** register.

CRC Post Processing Result (CRCRSLTPP)

Base 0x4403.0000
Offset 0x418
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RSLTPP | RO | 0x0000.0000 | Post Processing Result<br>This register contains the post-processed CRC result. |

# 13 Advance Encryption Standard Accelerator (AES)

This section describes the Advanced Encryption Standard (AES) cryptographic hardware-accelerated module.

## 13.1 AES Overview

This section introduces the AES and describes the AES main functions and connections in the device.

The advanced encryption standard (AES) security modules provide hardware-accelerated data encryption and decryption operations based on a binary key. The AES is a symmetric cipher module that supports a 128-bit, 192-bit, or 256-bit key in hardware for both encryption and decryption. The AES module is based on a symmetric algorithm, meaning that the encryption and decryption keys are identical. To encrypt data means to convert it from plain text to an unintelligible form called cipher text. Decrypting cipher text converts previously encrypted data back to its original plain text form.

The main features of the AES accelerator are:

- Support for basic AES encrypt and decrypt operations:
  - Galois/Counter Mode (GCM), with basic GHASH operation
  - Counter Mode with CBC-MAC (CCM)
  - XTS Mode
- Availability of the following feedback operating modes:
  - Electronic Code Book Mode (ECB)
  - Cipher Block Chaining Mode (CBC)
  - Counter Mode (CTR)
  - Cipher Feedback Mode (CFB), 128-bit
  - F8 Mode
- Key sizes 128-, 192- and 256-bits
- Support for CBC_MAC and Fedora 9 (F9) authentication modes
- Basic GHASH operation (when selecting no encryption)
- Key scheduling in hardware
- Support for µDMA transfers
- Fully synchronous design

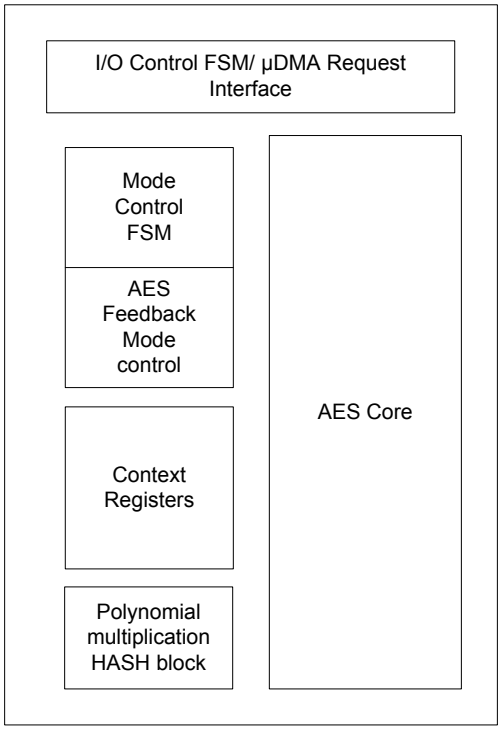
## 13.2 AES Functional Description

The following sections describe the features of the AES Module.

### 13.2.1 AES Block Diagram

Figure 13-1 on page 989 shows the AES block diagram. A single-core/dual-interface architecture is used.

**Figure 13-1. AES Block Diagram**




AES is an efficient implementation of the Rijndael cipher (the AES algorithm) and a 128-bit polynomial multiplication (referred to here as GHASH, as per the AES-GCM specification). Rijndael is a block cipher in which each data block is 128 bits. The polynomial multiplication multiplies two 128-bit vectors using the smallest 128-bit irreducible polynomial, represented by the following 128-bit string: $\{0^{120}\}||10000111$. The two implementations are combined into the AES wide-bus engine.

Depending on the availability of context and data, the AES wide-bus engine is automatically triggered to process the data. The AES wide-bus engine is directly connected to the context and data registers so that it can immediately start processing when all data is available. The AES wide-bus engine also interfaces to the I/O Control FSM/µDMA Request Interface.

AES comprises the following major functional blocks:

- Global control FSM and µDMA interface
- Register interface module
- The AES wide-bus engine

The AES wide-bus 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 wide-bus engine and starts each encryption/decryption operation.

- Feedback modes: The logic that implements the various feedback modes supported by AES.
- GHASH core: The polynomial multiplication algorithm used for AES-GCM
- 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

AES encryption requires a specific number of rounds, depending on the key length. The supported key lengths are 128-, 192-, and 256-bit, which require 10, 12, and 14 rounds, respectively, or 32, 38, and 44 clock cycles, respectively, because {number of clock cycles} = 2 + 3 x {number of rounds}.

The larger key lengths provide greater encryption strength at the expense of additional rounds and therefore reduced throughput. The overall throughput of the AES executing polynomial multiplication is adjusted based on the overall cryptographic performance. The AES module contains one Electronic Codebook (ECB) core and a dedicated 32-cycle polynomial multiplication module for performing GHASH operations. Polynomial multiplication operates in parallel with the AES core, if data is available for both modules.

Depending on the key size (128/192/256 bits), this core requires 32, 38, or 44 clock cycles to process one 128-bit data block. 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, 38, or 44 clock cycles.

#### 13.2.1.1 Interfaces

The interface signals to the AES module can be grouped into the following four categories:

- Clock and reset signals
- Register interface
- µDMA/INT interface, used to request new context and packet data or to indicate available result data (encrypted/decrypted data or authentication result)

#### 13.2.1.2 Register Interface

The register interface block performs all address decoding and control. However, not all registers are located in this block. The context and data input registers are in the AES wide-bus engine. The data output registers are available in this block.

#### 13.2.1.3 AES Wide-Bus Engine

The AES wide-bus engine performs the cryptographic operations. The composition of the AES core is:

- 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.

### AES Key Scheduler

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.

### AES Encryption Core

The AES encryption core implements the Rijndael algorithm as specified in [ FIPS-197]. This 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. This implementation of the AES encryption core has a 64-bit data path.

### AES Decryption Core

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. Once a decryption key is generated, subsequent decryption operations with the same key use this generated decryption key directly.

### AES Feedback Mode Block

The AES feedback mode block buffers the feedback parameters and controls the various feedback modes. For more information about the ECB, CBC, CTR, and CFB modes of operation, see the NIST-SP800-38A specification.

CTR implements the standard incrementing function, as described in the NIST-SP800-38A specification, with $m$ set to 16 or a multiple of 32.

AES-XTS mode requires a polynomial multiplication for initialization vector (IV) generation of the AES operation. This multiplication can be simplified when the first result is available due to the definition and use of the block number within a unit. The input for the polynomial multiplication is not directly j, but $\alpha^j$, where $\alpha = x^2$ in the $GF(2^{128})$ domain.

In addition, f8 encryption/decryption mode and f9 and (X)CBC-MAC authentication modes are available.

### GHASH Block

The data sequencer manages the data flow to and from the AES core. For data input, the data sequencer monitors the input buffer until a 16-byte block is available. If the AES core is idle, the data sequencer writes this data block to the internal working registers of the AES core, thus clearing the buffer for the next block.

After completing an encryption or decryption operation, the data sequencer writes the AES output to the output buffer. If the output buffer is full at the time of completion, the AES core is held until the buffer clears. Although the data sequencer is designed to support uninterrupted packet encryption, the host must properly manage the input and output packet buffers to achieve optimal performance.

### 13.2.2 AES Algorithm

The AES algorithm generates block ciphers. The AES block size is 16 bytes. The AES key(s) can be coded on 128, 192, or 256 bits. The larger key sizes provide a higher level of security, but at the cost of a moderate decrease in throughput.

For the AES algorithm:

- The length of the input and output blocks is 128 bits, which is represented by Nb = 4, which reflects the number of 32-bit words.
- The length of the cipher key (K) is 128, 192, or 256 bits. The key length is represented by Nk = 4, 6, or 8, which reflects the number of 32-bit words in the cipher key.
- The number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4 (128-bit key); Nr = 12 when Nk = 6 (192-bit key); and Nr = 14 when Nk = 8 (256-bit key).

Table 13-1 on page 992 lists the combinations of keys, blocks, and rounds.

**Table 13-1. Key-Block-Round Combinations**

| Key | Key Length (Nk) | Block Size (Nb) | Number of Rounds (Nr) |
|---|---|---|---|
| 128 bits | 4 | 4 | 10 |
| 192 bits | 6 | 4 | 12 |
| 256 bits | 8 | 4 | 14 |

The AES algorithm for cipher and inverse cipher uses a round function composed of four different byte-oriented transformations:

- Byte substitution using a substitution table (S-Box)

  This transformation is a non-linear byte substitution that operates independently on each byte of the state (the state is an intermediate processed block of 128 bits inside the AES; the state is arranged as an array of [4 x Nk] bytes) using an S-Box. This S-Box transformation is reversible.

- Shifting rows of the state array by different offsets

  In this transformation, the bytes in the last three rows of the state are cyclically shifted over different numbers of bytes (offsets). The first row (r = 0) is not shifted.

- Mixing the data within each column of the state array

  This transformation operates on the state column-by-column, treating each column as a four-term polynomial. The columns are considered polynomials over GF($2^8$) and multiplied modulo $x^4 + 1$ with a fixed polynomial a(x).

- Adding a round key to the state

  In this transformation, a round key is added to the state by a simple bitwise XOR operation. Each round key consists of Nb words from the key schedule.

The AES algorithm takes the cipher key (K) and performs a key expansion routine to generate a key schedule. The key expansion generates a total of Nb x (Nr + 1) words: The algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted $[w_i]$, with i in the range $0 \le i < Nb \times (Nr + 1)$.

### 13.2.3 AES Operating Modes

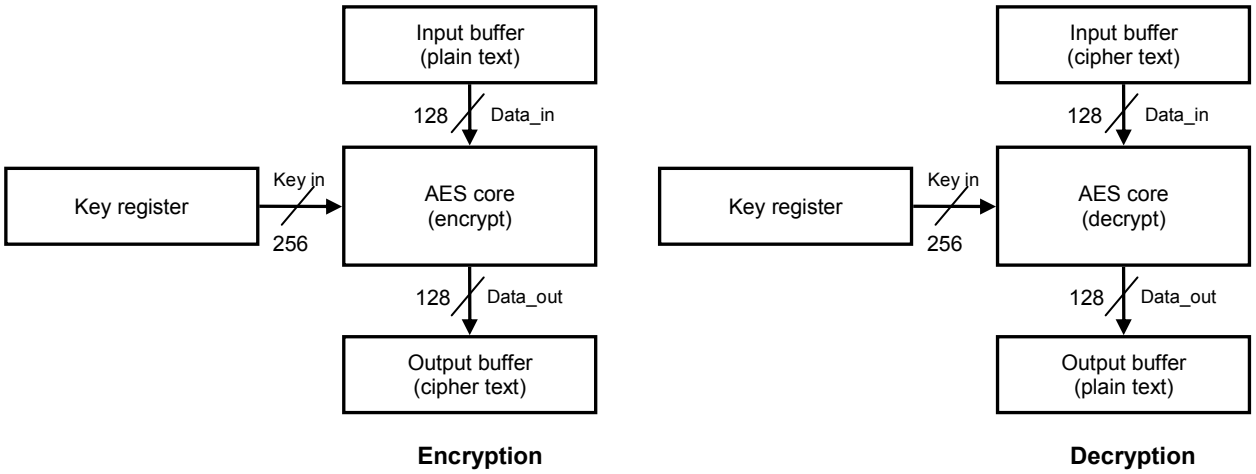
#### 13.2.3.1 Supported Modes of Operation

***ECB Feedback Mode***

Figure 13-2 on page 993 shows the basic ECB feedback mode of operation, where the input data is passed directly to the basic cryptographic core and the output is passed directly to the output buffer.

For decryption, the cryptographic core operates in reverse: the decryption data path is used for data processing, whereas encryption uses the encryption data path.

**Figure 13-2. AES - ECB Feedback Mode**

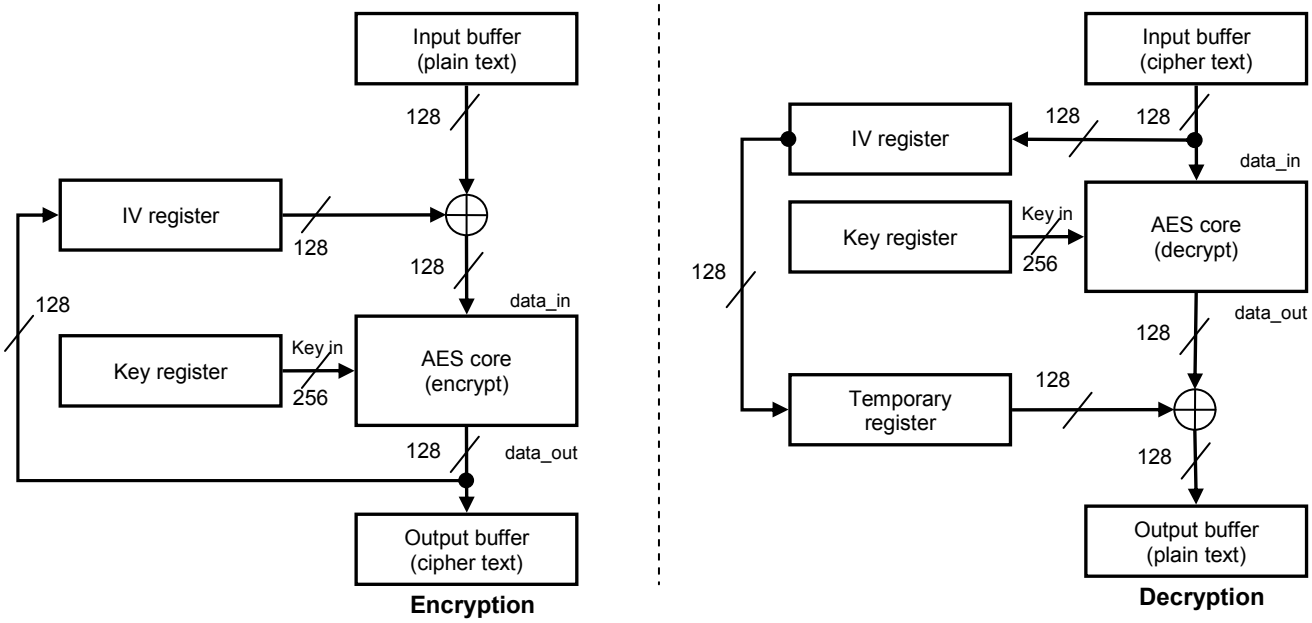



***CBC Feedback Mode***

Figure 13-3 on page 994 shows the CBC feedback mode of operation, where the input data is XORed with the IV before it is passed to the basic cryptographic core. The output of the cryptographic core passes directly to the output buffer and becomes the next IV.

The operation is reversed for decryption, resulting in an XOR at the output of the cryptographic core. The input cipher text of the current operation is the IV for the next operation.

**Figure 13-3. AES - CBC Feedback Mode**

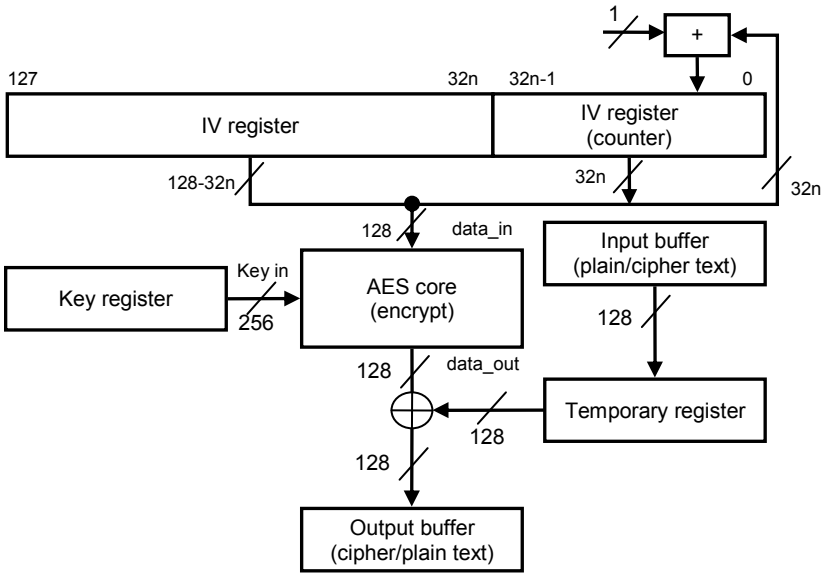



### *CTR and ICM Feedback Modes*

Figure 13-4 on page 994 shows the counter feedback (CTR/ICM) mode of operation. This operation encrypts the IV. The output of the cryptographic core (encrypted IV) is XORed with the data, thus creating the output result.

The IV is built out of two components: a fixed part and a counter part. The counter part is incremented with each block. The counter width is selectable per context and can be 16, 32, 64, 96, or 128 bits wide. In this mode, encryption and decryption use the same operation.

**Figure 13-4. AES Encryption With CTR/ICM Mode**

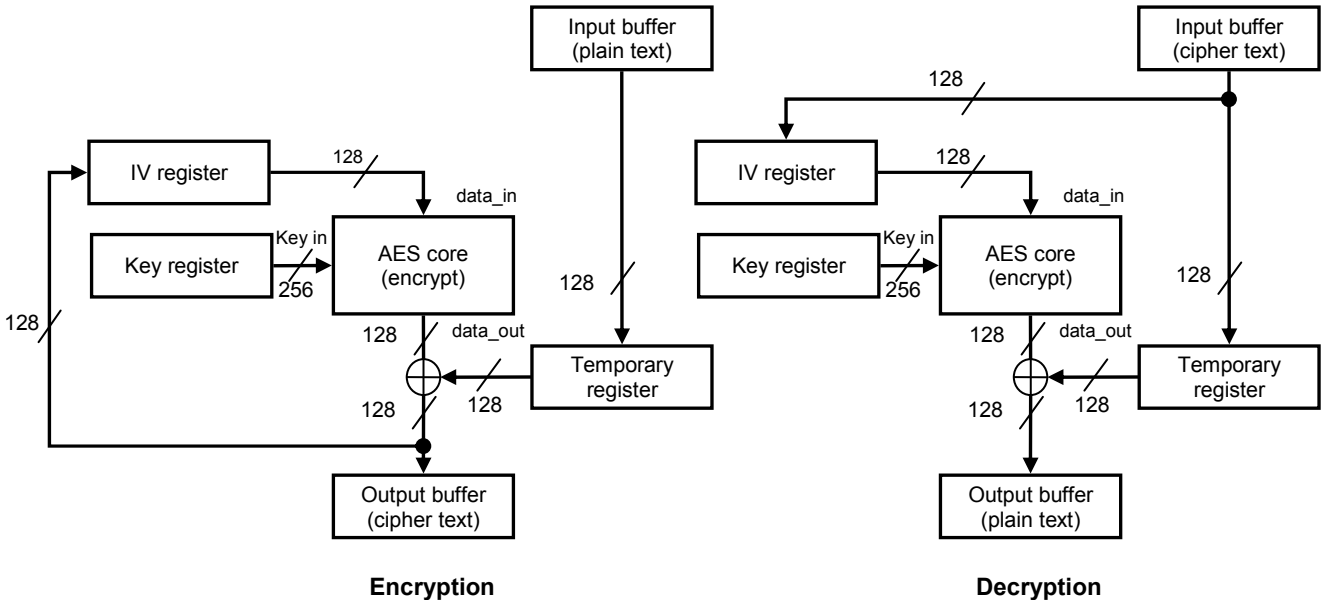


**Note:** The value for n can be 1, 2, 3, or 4 for CTR mode and is ½ for ICM mode.

### *CFB Mode*

Figure 13-5 on page 995 shows the full block (128 bits) CFB mode of operation for encryption and decryption. The input for the cryptographic core is the IV; the result is XORed with the data. The result is fed back through the IV register as the next input for the cryptographic core. The decryption operation is reversed, but the cryptographic core still performs encryption.

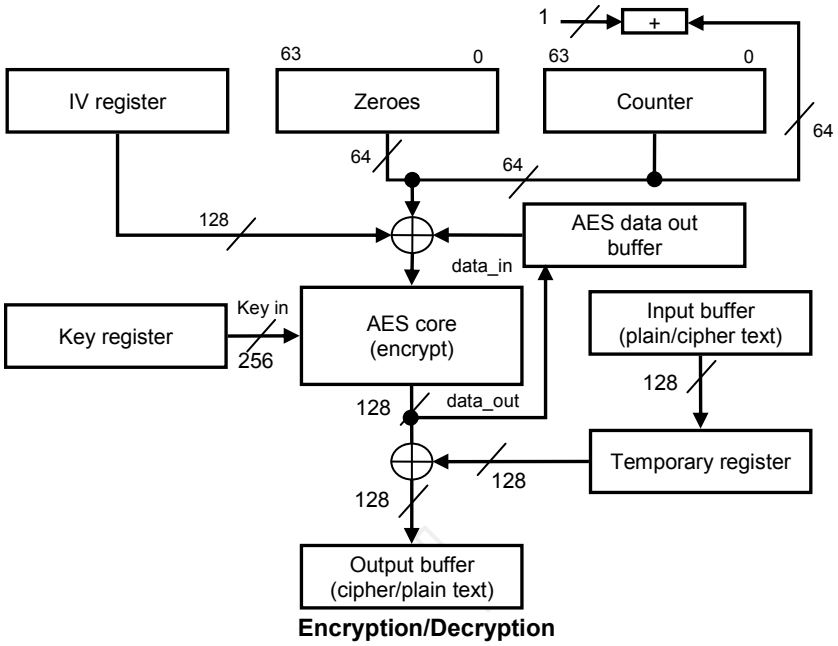
**Figure 13-5. AES - CFB Feedback Mode**




### *F8 Mode*

Figure 13-6 on page 996 shows the F8 feedback mode of operation for encryption and decryption. The input to the cryptographic core is the result of the XOR operation of the previous cryptographic core output, a constant IV, and a block counter. The output of the cryptographic core is XORed with the input to create the result. In this mode, encryption and decryption use the same operations.

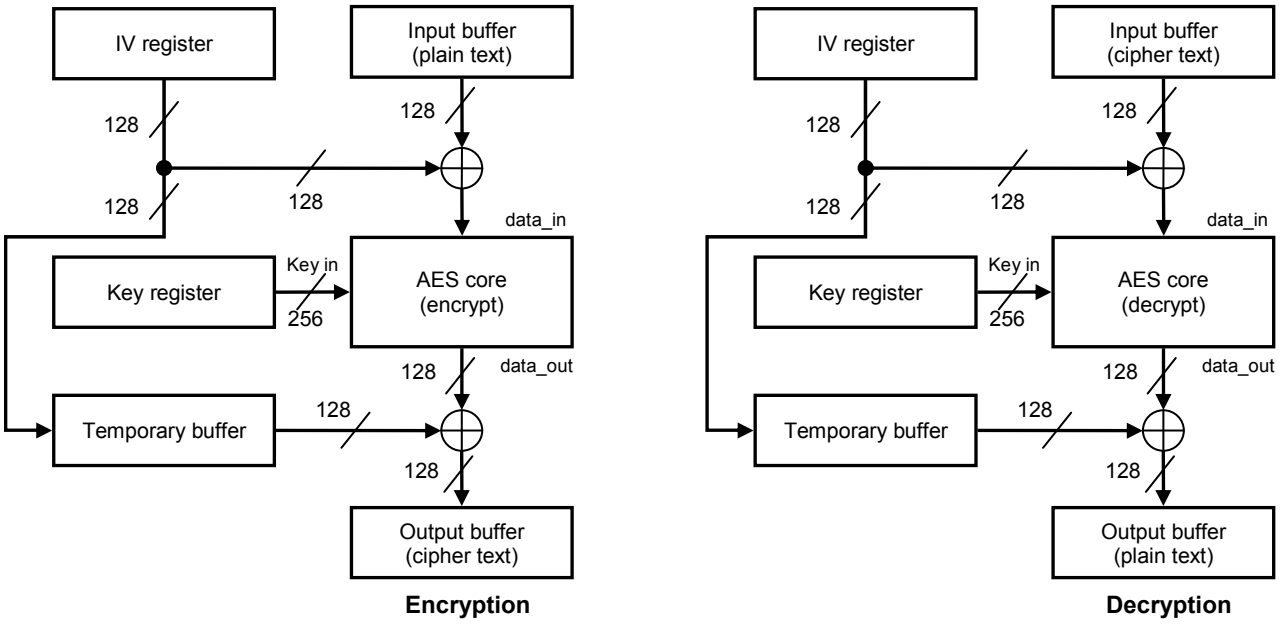
**Figure 13-6. AES - F8 Mode**




### *XTS Operation*

Figure 13-7 on page 996 shows the XTS mode of operation for encryption and decryption. The input to the cryptographic core is XORed with the IV; the output of the cryptographic core is XORed with the same IV. For decryption, the cryptographic core operates in reverse, but the XOR operations are the same.

**Figure 13-7. AES - XTS Operation**

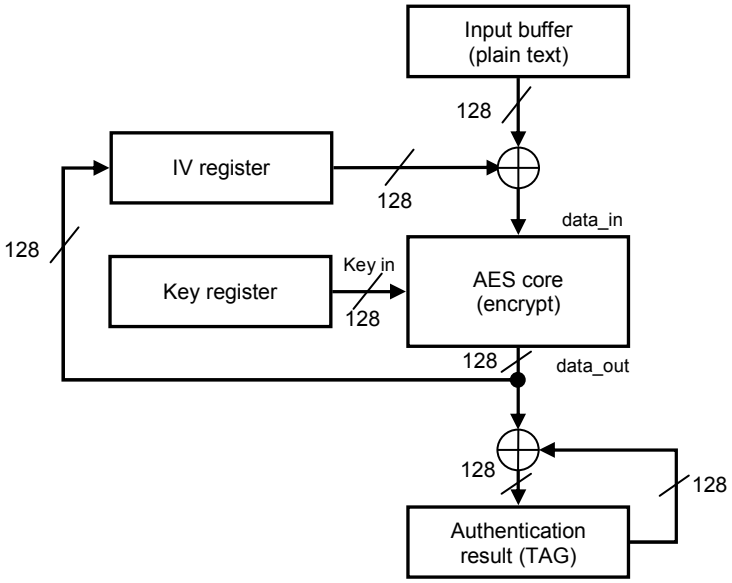


**Note:** The IV is created with an initial encryption, followed by an LFSR operation for each new block.

### *F9 Operation*

Figure 13-8 on page 997 shows the F9 authentication mode of operation, where the input to the cryptographic core is XORed with the IV, and the output is XORed with the previous result to create the next result. The cryptographic core output is fed back as IV for the next block. The result is the output of the last XOR operation of the cryptographic core output.

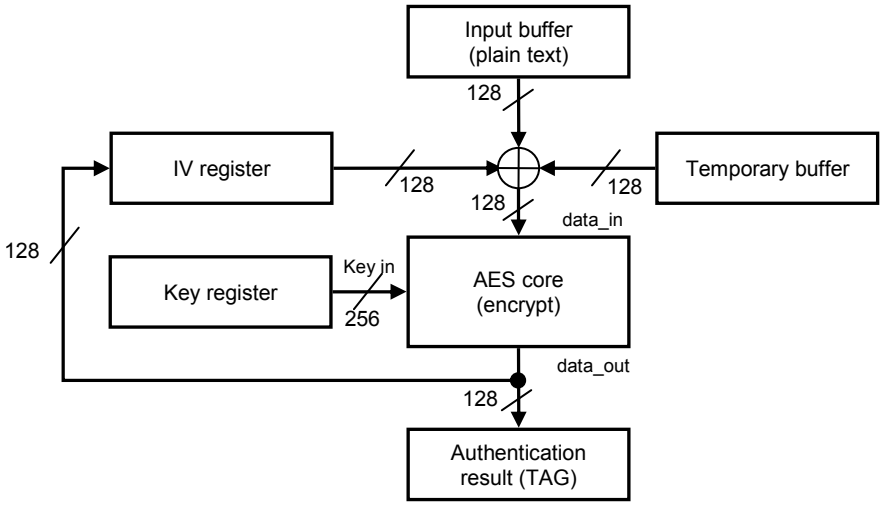
**Figure 13-8. AES - F9 Operation**




### *CBC-MAC Operation*

Figure 13-9 on page 998 shows the CBC-MAC authentication mode of operation, where the input to the cryptographic core is XORed with the IV. The cryptographic core output is then fed back as IV for the next block. The last data input block is XORed with an additional input value stored in the temporary buffer; this can be any precalculated value and is dependent on the alignment of the last input block. The result is the cryptographic core output of the last encryption operation.

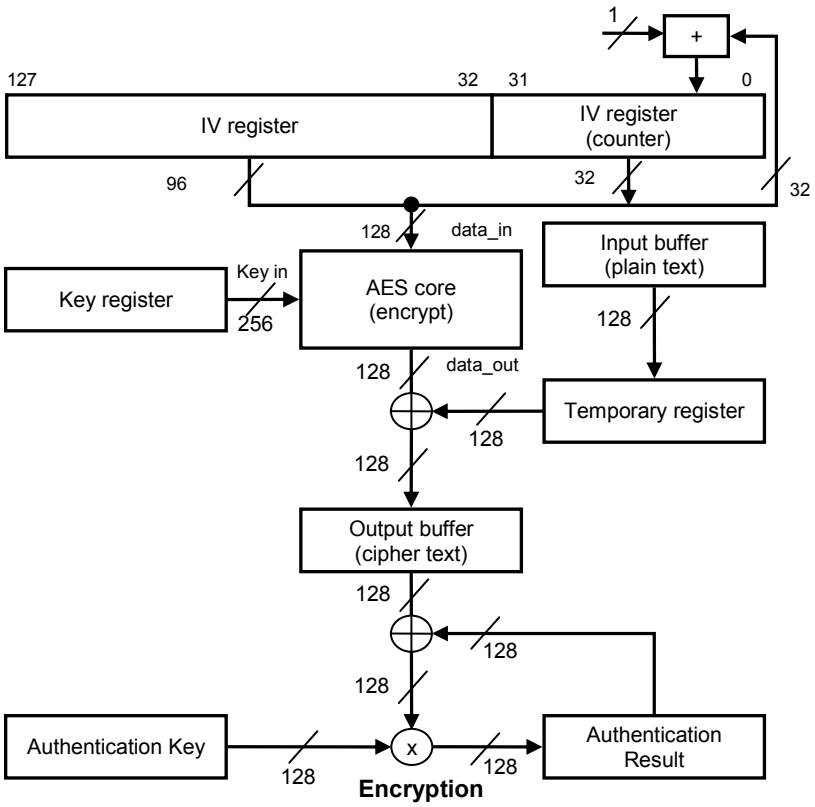
**Figure 13-9. AES - CBC-MAC Authentication Mode**




### *GCM Operation*

Figure 13-10 on page 999 shows one round of a GCM operation for encryption and decryption. A 32-bit counter is used as IV (as it is for CTR mode). The data is encrypted in the same way as in CTR mode, by XORing the cryptographic core output with the input. After the encryption/decryption, the ciphertext is XORed with the intermediate authentication result. The XORed result is used as input for the polynomial multiplication to create the next (intermediate) authentication result. For more information about the GCM protocol, see the section called “GCM Protocol Operation” on page 1000.

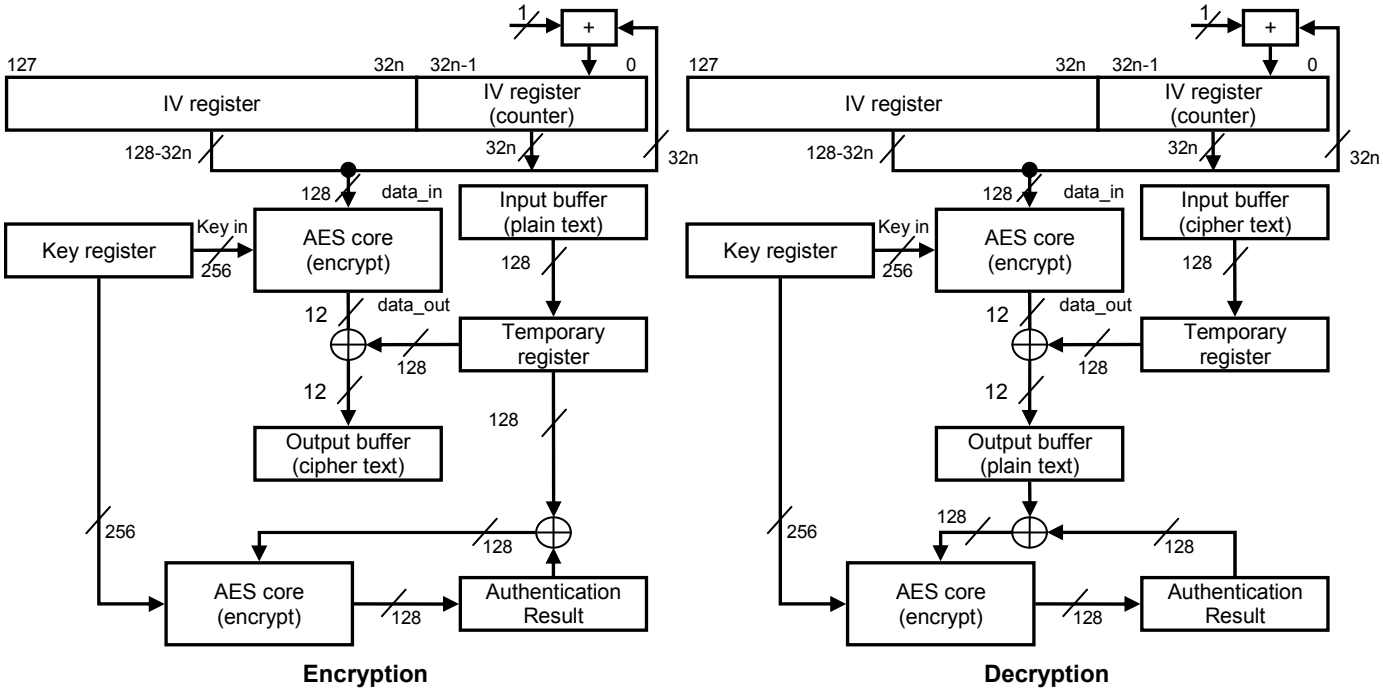
**Figure 13-10. AES - GCM Operation**




### *CCM Operation*

Figure 13-11 on page 1000 shows one round of a CCM (counter with CBC-MAC) operation for encryption and decryption. A 32-bit counter is used as IV (as it is for CTR mode). The data is encrypted in the same way as in CTR mode, by XORing the cryptographic core output with the input. Immediately after the encryption operation, the plaintext is XORed with the intermediate authentication result. The XOR result is used as input for a second encryption operation to calculate the next (intermediate) authentication result.

**Figure 13-11. AES - CCM Operation**




### 13.2.3.2 Extended and Combined Modes of Operations

This section describes the protocols (or autonomous precalculations) supported by the AES wide-bus engine.

#### *GCM Protocol Operation*

A GCM protocol operation is a combined operation consisting of encryption/decryption and authentication. A part of the input data stream can be authenticated only, while normally most of the input data is encrypted/decrypted and authenticated. The authentication-only data must always be in front of the data requiring encryption. Within GCM, the authentication-only data is called the additional authentication data (AAD). The AAD is fetched independently of other data.

The intermediate (temporary) result data is used as input to the remaining authentication operation. Because the authentication operation does not require the cryptographic core but only the polynomial multiplication, encryption/decryption and authentication can be performed in parallel. After encryption of the last data block, additional polynomial multiplication and encryption are required to authenticate a 128-bit length vector and finally encrypt the authentication result.

#### *CCM Protocol Operation*

The CCM protocol operation is a combined operation consisting of encryption/decryption and authentication. The authentication and encryption/decryption operations use the cryptographic core; these are executed sequentially on the AES core. A part of the data stream can require authentication only. The authentication-only data must always be in front of the data requiring encryption.

Authentication starts with the encryption of a predefined block B0. This block consists of flags, nonce, and message length. The next blocks contain the authentication data length concatenated with the authentication-only data. After processing the authentication-only data, the encryption/decryption operations are performed, each followed by the related authentication of the plaintext data block (which equals the input in the case of encryption, and the output in the case of

decryption). The final authentication result must be encrypted using the output of the encryption of the IV block A0. This block contains the IV (consisting of flags and nonce) concatenated with the counter, which is zero for A0.

### 13.2.4 AES Software Reset

To perform a software reset of the AES module, write a 1 to the SOFTRESET bit in the **AES System Configuration (AES_SYSCONFIG)** register. The RESETDONE bit in the **AES Secure System Status (AES_SYSSTATUS)** register indicates that the software reset is complete when its value is 1. When the software reset completes, the SOFTRESET bit in the **AES_SYSCONFIG** register is automatically reset. Software must ensure that the software reset completes before doing any operations.

The behavior of the software reset is the same as the hardware reset, except that the software reset bit resets this module without affecting the reset core domain of the entire device.

### 13.2.5 Power Management

To save power, the application can disable the clock to the AES module when not in use. The AES is clock gated by setting the AESCFG bit in the **Cryptographic Modules Clock Gating Request (CCMCGREQ)** register, CRC and Cryptographic Modules (CCM) offset 0x204. The AES in addition to the DES, SHA/MD5 and CRC can also be clock gated as a group by setting the D0 bit in the **CRC and Cryptographic Modules (DCGCCCM)** register, System Control Module offset 0x874.

### 13.2.6 Hardware Requests

The AES module can assert a µDMA request for context in, context out, input data, or output data read. The **AES uDMA Interrupt Mask (AES_DMAIM)** register can be set to generate interrupts during the following events:

- Context In µDMA request (AES0 Cin)
- Context Out µDMA request (AES0 Cout)
- Data In µDMA request (AES0 Din)
- Data Out µDMA request (AES0 Dout)

The AES Module can be programmed to assert an interrupt when the uDMA has completed its last transfer. Please refer to “AES µDMA Interrupt Register Descriptions (CCM Offset)” on page 1033 for more information.

If context and data transfers are to be handled through software, then the **AES Interrupt Enable (AES_IRQENABLE)**, offset 0x090, can be used to enable interrupt triggering when context out, context in, data in or data out is ready. The **AES Interrupt Status (AES_IRQSTATUS)**, offset 0x08C, indicates when an interrupt is triggered.

**Table 13-2. Interrupts and Events**

| Event | Description |
|---|---|
| **AES_IRQSTATUS**[3]: CONTEXT_OUT | Context output interrupt |
| **AES_IRQSTATUS**[2]: DATA_OUT | Data output interrupt |
| **AES_IRQSTATUS**[1]: DATA_IN | Data input interrupt |
| **AES_IRQSTATUS**[0]: CONTEXT_IN | Context input interrupt |

## 13.3 AES Performance Information

Table 13-3 on page 1002 lists the performance for all supported key sizes and modes of operations. It assumes that the engine is kept fully utilized (that is, the host is supplying input blocks and retrieving output blocks in such a way that the engine never has to wait for input), and that the previous output has been retrieved before the next output is ready.

**Table 13-3. AES Module Performance (Input/Output Block Size = 128)**

| Key Size | Mode of Operation | Cycles per Block[a] | Throughput (bits/cycle) |
|---|---|---|---|
| 128 | ECB-encrypt / decrypt | 32 | 4.00 |
| | CBC-decrypt | | |
| | CTR/ICM-encrypt / decrypt | | |
| | CFB128-decrypt | | |
| | CBC-encrypt | 33 | 3.88 |
| | OFB-encrypt / decrypt | | |
| | f8-encrypt / decrypt | | |
| | CFB128-encrypt | | |
| | XTS-encrypt / decrypt | | |
| | GCM-outbound / inbound | | |
| | CCM-encrypt / decrypt | 66 | 1.94 |
| | CBC-MAC[b] | 33 | 3.88 |
| | f9[b] | | |
| 192 | ECB-encrypt / decrypt | 38 | 3.37 |
| | CBC-decrypt | | |
| | CTR/ICM-encrypt / decrypt | | |
| | CFB128-decrypt | | |
| | CBC-encrypt | 39 | 3.28 |
| | OFB-encrypt / decrypt | | |
| | f8-encrypt / decrypt | | |
| | CFB128-encrypt | | |
| | XTS-encrypt / decrypt | | |
| | GCM-outbound / inbound | | |
| | CCM-encrypt / decrypt | 78 | 1.64 |
| | CBC-MAC[b] | 39 | 3.28 |
| | f9[b] | | |

**Table 13-3. AES Module Performance (Input/Output Block Size = 128) *(continued)***

| Key Size | Mode of Operation | Cycles per Block[a] | Throughput (bits/cycle) |
|---|---|---|---|
| 256 | ECB-encrypt / decrypt | 44 | 2.91 |
| | CBC-decrypt | | |
| | CTR/ICM-encrypt / decrypt | | |
| | CFB128-decrypt | | |
| | CBC-encrypt | 45 | 2.84 |
| | OFB-encrypt / decrypt | | |
| | f8-encrypt / decrypt | | |
| | CFB128-encrypt | | |
| | XTS-encrypt / decrypt | | |
| | GCM-outbound inbound | | |
| | CCM-encrypt / decrypt | 90 | 1.42 |
| | CBC-MAC[b] | 45 | 2.84 |
| | f9[b] | | |

a. Standard Block Size (128 bits)

b. Input Only

The "Cycles per block" numbers do not apply to the first block after selection of a new key and/or mode of operation, or the last block before switching to a new algorithm. The next table indicates the additional clock cycles; required for changing context data.

**Table 13-4. AES Module Packet Mode Switch Overhead**

| New Context | Cycles needed for first block/and to finish the last block | Total |
|---|---|---|
| mode is encrypt | 1 / 1 | 2 |
| mode is decrypt, key size is 128 | 32 / 1 | 33 |
| mode is decrypt, key size is 192 | 38 / 1 | 39 |
| mode is decrypt, key size is 256 | 44 / 1 | 45 |
| mode is XTS, key size is 128 | 34 / 1 | 35 |
| mode is XTS, key size is 192 | 40 / 1 | 41 |
| mode is XTS, key size is 256 | 46 / 1 | 47 |
| outbound GCM, AES key size is 128 | 33[a] / 52 | 85 |
| outbound GCM, AES key size is 192 | 39[a] / 52 | 91 |
| outbound GCM, AES key size is 256 | 45 [a]/ 52 | 97 |
| inbound GCM, AES key size is 128 | 33[a] / 27 | 60 |
| inbound GCM, AES key size is 192 | 39[a] / 27 | 66 |
| inbound GCM, AES key size is 256 | 45[a] / 27 | 72 |
| outbound CCM, AES key size is 128 | 33[b] / 33 | 66 |
| outbound CCM, AES key size is 192 | 39 [b]/ 39 | 78 |
| outbound CCM, AES key size is 256 | 45[b] / 45 | 90 |
| inbound CCM, AES key size is 128 | 33 [b]/ 33 | 66 |
| inbound CCM, AES key size is 192 | 39[b] / 39 | 78 |

**Table 13-4. AES Module Packet Mode Switch Overhead *(continued)***

| New Context | Cycles needed for first block/and to finish the last block | Total |
|---|---|---|
| inbound CCM, AES key size is 256 | 45[b] / 45 | 90 |

a. Numbers for regular GCM mode (H is precalculated and Y0-encrypted need to be calculated internally using the new IV). If H needs to be calculated by the core (complete GCM mode), this number needs to be doubled. If Y0-encrypted is not calculated (forced to zero, such that the hash result is not encrypted) this number is zero.

b. Numbers for regular CCM mode. Dependent on the AAD length it is possible that one additional encryption needs to be done to finalize the AAD authentication; if the additional operation is required, this number needs to be doubled.

# 13.4 AES Module Programming Guide

## 13.4.1 AES Low - Level Programming Models

This section describes the low-level hardware programming sequences for configuring and using the AES Module.

### 13.4.1.1 Global Initialization

The following list describes the requirements for initializing the AES and surrounding modules when AES is used for the first time after a device reset.

1. When reset has completed, enable the AES by setting the R0 bit in the **CRC and Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM)**, System Control offset 0x674. When the R0 bit is set in the **CRC and Cryptographic Modules Peripheral Ready (PRCCM)**, System Control offset 0xA74 register, the AES is powered and ready to be configured.

2. Configure the AES µDMA channels for Context In, Context Out, Data In, and/or Data Out by programming the appropriate encoding value in the **DMA Channel Map Select n (DMACHMAPn)** register in the µDMA module, offset 0x510. For more information on how to program channel assignments as well as enabling burst and the configured channels, refer to “Micro Direct Memory Access (µDMA)” on page 708.

3. Execute a software reset by setting the SOFTRESET bit in the **AES_SYSCONFIG** register. When reset is complete, the RESETDONE bit reads as 1 in the **AES_SYSSTATUS** register.

4. If the AES channels are configured in the µDMA, enable the required AES DMA requests by programming bits [9:5] of the **AES_SYSCONFIG** register, in addition to the completion interrupts in the **AES DMA Interrupt Mask (AES_DMAIM)** register, CCM offset 0x020.

5. Specify the size of the keys by programming the KEY_SIZE bit field in the **AES_CTRL** register.

6. Load **AES Key 1 (AES_KEY1_n)** register.

7. Load **AES Key 2 (AES_KEY2_n)** register if it is used by the configuration mode. Refer to Table 13-6 on page 1012 for information regarding which configuration modes require a load to this register.

8. Configure the AES for the appropriate encryption/decryption mode (see the section called “Subsequence: Initialize CCM AES Core Mode” on page 1005 through the section called “Subsequence: Initialize CBC AES Core Mode” on page 1006).

9. Select encryption or decryption by programming the DIRECTION bit in the **AES Control (AES_CTRL)** register, offset 0x050.

#### 13.4.1.2 Initialization Subsequence

The following sections list the initialization subsequences for the available encryption/decryption modes:

***Subsequence: Initialize CCM AES Core Mode***

The CCM mode initialization is as follows:

1. Define the width of the length field and the length of the authentication field by programming the CCM_L and CCM_M bit fields in the **AES_CTRL** register at offset 0x050.
2. Enable counter mode by setting the CTR bit in the **AES_CTRL** register.
3. Load the authentication data length in the AUTH field of the **AES Authentication Data Length (AES_AUTH_LENGTH)** register at offset 0x05C.
4. Select the IV counter by programming the CTR_WIDTH field in the **AES_CTRL** register.
5. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize GCM AES Core Mode***

The following steps to enable GCM mode is as follows:

1. Enable counter mode by setting the CTR bit in the **AES_CTRL** register.
2. Load the authentication data length in the AUTH field of the **AES Authentication Data Length (AES_AUTH_LENGTH)** register at offset 0x05C.
3. Select the IV counter by programming the CTR_WIDTH field in the **AES_CTRL** register.
4. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize CBC-MAC AES Core Mode***

To initialize CBC-MAC Mode:

1. Enable CBC-MAC Mode by setting the CBCMAC bit in the **AES_CTRL** register.
2. Select encryption mode by setting the DIRECTION bit in the **AES_CTRL** register at offset 0x050
3. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize F9 AES Core Mode***

The steps for configuring the AES for F9 mode is as follows:

1. Enable F9 Mode by setting the F9 bit in the **AES_CTRL** register.
2. Set the key size to 128 bits by programming the KEY_SIZE field to 0x1 in the **AES_CTRL** register.
3. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize F8 AES Core Mode***

The steps for configuring the AES for F8 mode is as follows:

1. Enable F8 Mode by setting the F8 bit in the **AES_CTRL** register.
2. Select the counter width by programming the CTR_WIDTH field in the **AES_CTRL** register.
3. Set the key size to 128 bits by setting the KEY_SIZE field to 0x1 in the **AES_CTRL** register.
4. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize XTS AES Core Mode***

The steps for XTS mode configuration are as follows:

1. Enable XTS Mode by configuring the XTS field in the **AES_CTRL** register.
2. If the XTS field in the **AES_CTRL** register indicates that the AAD length is required, load the AAD length in the **AES_AUTH_LENGTH** register.
3. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize CFB AES Core Mode***

To initialize the AES code for CFB mode:

1. Enable CFB Mode by setting the CFB bit in the **AES_CTRL** register.
2. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize ICM AES Core Mode***

To initialize the AES code for ICM mode:

1. Enable ICM Mode by setting the ICM bit in the **AES_CTRL** register.
2. Configure for a 16-bit counter by programming the CTR_WIDTH to 0x0 in the **AES_CTRL** register.
3. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize CTR AES Core Mode***

To initialize CTR mode:

1. Enable CTR Mode by setting the CTR bit in the **AES_CTRL** register.
2. Select counter width by programming the CTR_WIDTH in the **AES_CTRL** register.
3. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

***Subsequence: Initialize CBC AES Core Mode***

To configure CBC mode:

1. Enable CBC Mode by setting the MODE bit in the **AES_CTRL** register.
2. Load the **AES Initialization Vector Input n (AES_IV_IN_n)** registers at offset 0x040 to 0x04C.

### 13.4.1.3 Operational Modes Configuration

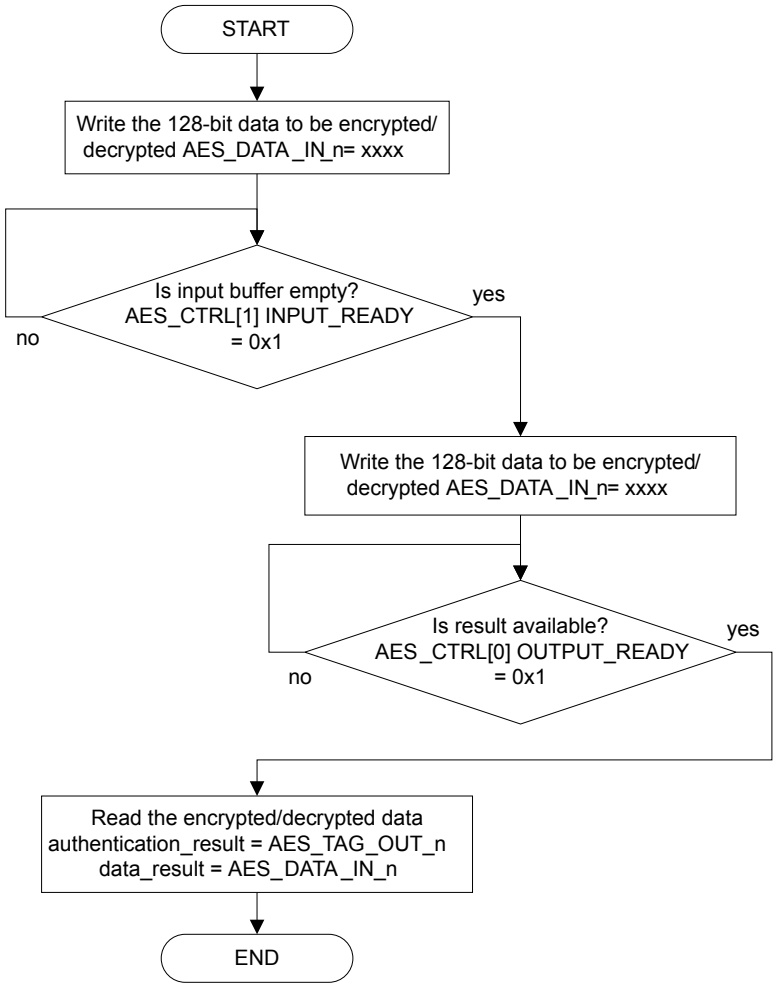
#### *AES Polling Mode*

#### *Main Sequence: AES Polling Mode*

Figure 13-12 on page 1007 shows AES polling mode. The registers used in AES polling mode are:

- **AES Data RW Plaintext/Ciphertext 0 (AES_DATA_IN_0)** registers, offset 0x060 to 0x06C
- **AES Control (AES_CTRL)** register, offset 0x050
- **AES Hash Tag Out 0 (AES_TAG_OUT_0)**, offset 0x070

**Figure 13-12. AES Polling Mode**




#### *AES Interrupt Mode*

The application can use software interrupts to control the flow of Context In, Context Out, Data In, and Data Out requests. To enable these interrupts:

1. Once the device has been initialized, following the initialization sequences described in “Global Initialization” on page 1004 and “Initialization Subsequence” on page 1005, the application can enable the AES module interrupts through the **AES Interrupt Enable (AES_IRQENABLE)** register, offset 0x090. If all four interrupts must be enabled, the application can write 0x0000.000F to the **AES_IRQENABLE** register.

2. Load the input buffers, **AES_DATA_IN_n**, with data.

**Note:** If the application uses Interrupt Mode, an interrupt is generated for each block of processed data. To support larger data flow, AES µDMA Mode should be used and the bits in the **AES_IRQENABLE** register should be cleared.

***AES DMA Mode***

When AES DMA Mode is enabled, the **AES_IRQENABLE** register should be cleared. To enable the µDMA to transfer data follow these steps:

1. Once the AES module has been initialized, enable the AES module µDMA channels by programming the **DMA Channel Map Select n (DMACHMAPn)** register in the µDMA module. Further µDMA configuration guidelines are available in the “Micro Direct Memory Access (µDMA)” on page 708.

2. Configure the dma_done interrupts by programming the **AES DMA Masked Interrupt Status (AES_DMAMIS)** register, at CCM offset 0x028.

3. Enable the µDMA channels in the AES by programming the µDMA enable bits in the **AES System Configuration (AES_SYSCONFIG)** register, offset 0x084.

4. The input buffer registers, **AES_DATA_IN_n**, at offsets 0x060 to 0x06C, are loaded.

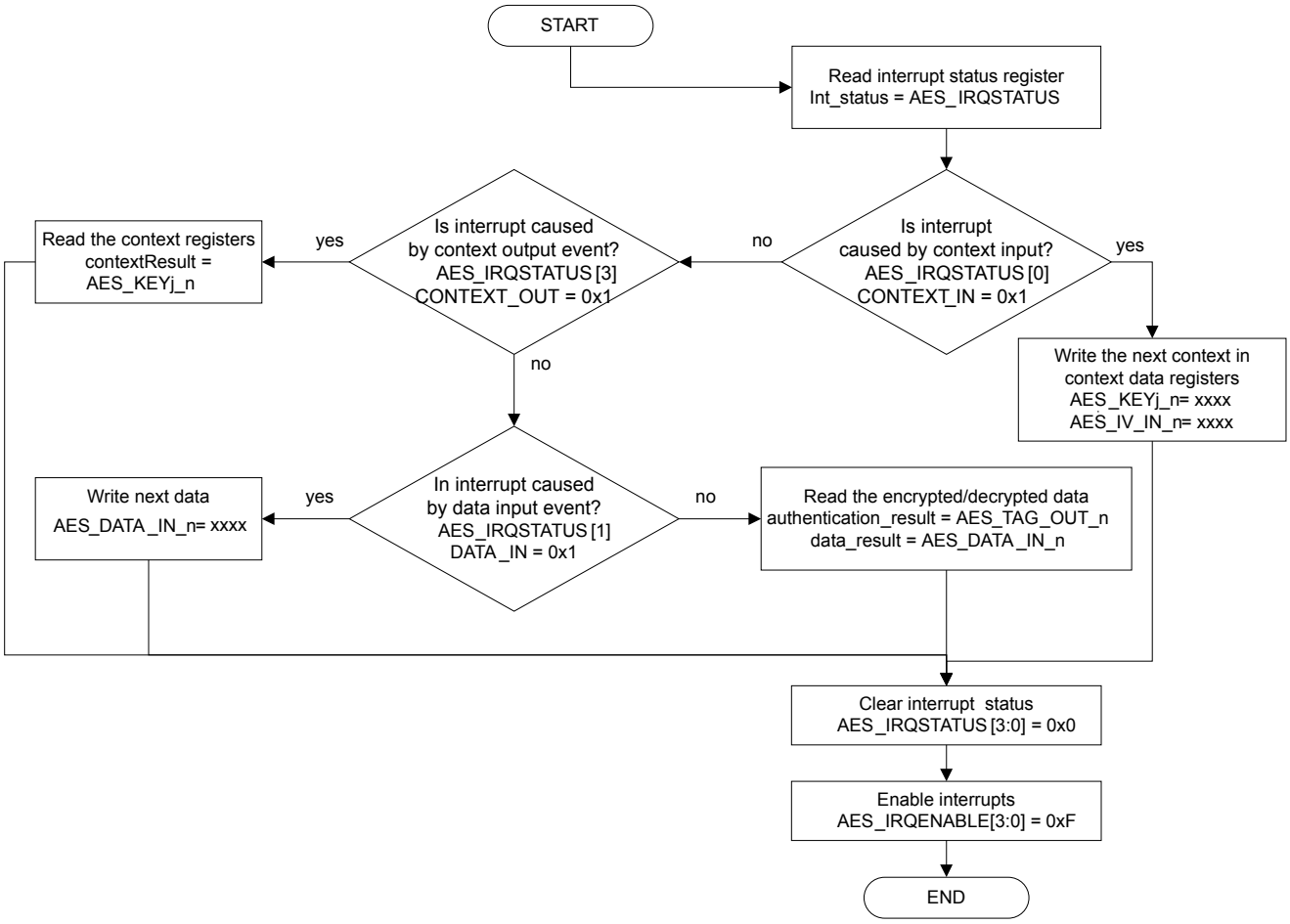
#### 13.4.1.4 AES Events Servicing

***Interrupt Servicing***

This section describes the event servicing of the module. Figure 13-13 on page 1009 shows the AES interrupt service. The registers used during event servicing are:

- **AES_IRQSTATUS**
- **AES_KEY1_n**
- **AES_KEY2_n**
- **AES_IV_IN_n**
- **AES_DATA_IN_n**
- **AES_TAG_OUT_n**
- **AES_IRQENABLE**

**Figure 13-13. AES Interrupt Service**




## 13.5 Register Map

Table 13-5 on page 1009 lists the AES registers. The AES Module comprises registers that exist at an offset relative to the AES Module base address and a small set of AES µDMA registers that exist at an offset relative to an Encryption Control Module base address. The AES Module register offsets are relative to the base address 0x4403.6000. The AES µDMA interrupt registers are offset are relative to the base address 0x4403.0000.

**Table 13-5. AES Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **AES Module Registers (AES Module Offset)** | | | | | |
| 0x000 | AES_KEY2_6 | RW | 0x0000.0000 | AES Key 2_6 | 1012 |
| 0x004 | AES_KEY2_7 | RW | 0x0000.0000 | AES Key 2_7 | 1012 |
| 0x008 | AES_KEY2_4 | RW | 0x0000.0000 | AES Key 2_4 | 1012 |

**Table 13-5. AES Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x00C | AES_KEY2_5 | RW | 0x0000.0000 | AES Key 2_5 | 1012 |
| 0x010 | AES_KEY2_2 | RW | 0x0000.0000 | AES Key 2_2 | 1012 |
| 0x014 | AES_KEY2_3 | RW | 0x0000.0000 | AES Key 2_3 | 1012 |
| 0x018 | AES_KEY2_0 | RW | 0x0000.0000 | AES Key 2_0 | 1012 |
| 0x01C | AES_KEY2_1 | RW | 0x0000.0000 | AES Key 2_1 | 1012 |
| 0x020 | AES_KEY1_6 | RW | 0x0000.0000 | AES Key 1_6 | 1012 |
| 0x024 | AES_KEY1_7 | RW | 0x0000.0000 | AES Key 1_7 | 1012 |
| 0x028 | AES_KEY1_4 | RW | 0x0000.0000 | AES Key 1_4 | 1012 |
| 0x02C | AES_KEY1_5 | RW | 0x0000.0000 | AES Key 1_5 | 1012 |
| 0x030 | AES_KEY1_2 | RW | 0x0000.0000 | AES Key 1_2 | 1012 |
| 0x034 | AES_KEY1_3 | RW | 0x0000.0000 | AES Key 1_3 | 1012 |
| 0x038 | AES_KEY1_0 | RW | 0x0000.0000 | AES Key 1_0 | 1012 |
| 0x03C | AES_KEY1_1 | RW | 0x0000.0000 | AES Key 1_1 | 1012 |
| 0x040 | AES_IV_IN_0 | RW | 0x0000.0000 | AES Initialization Vector Input 0 | 1014 |
| 0x044 | AES_IV_IN_1 | RW | 0x0000.0000 | AES Initialization Vector Input 1 | 1014 |
| 0x048 | AES_IV_IN_2 | RW | 0x0000.0000 | AES Initialization Vector Input 2 | 1014 |
| 0x04C | AES_IV_IN_3 | RW | 0x0000.0000 | AES Initialization Vector Input 3 | 1014 |
| 0x050 | AES_CTRL | RW | 0x8000.0000 | AES Control | 1015 |
| 0x054 | AES_C_LENGTH_0 | RW | 0x0000.0000 | AES Crypto Data Length 0 | 1020 |
| 0x058 | AES_C_LENGTH_1 | RW | 0x0000.0000 | AES Crypto Data Length 1 | 1020 |
| 0x05C | AES_AUTH_LENGTH | RW | 0x0000.0000 | AES Authentication Data Length | 1021 |
| 0x060 | AES_DATA_IN_0 | RW | 0x0000.0000 | AES Data RW Plaintext/Ciphertext 0 | 1022 |
| 0x064 | AES_DATA_IN_1 | RW | 0x0000.0000 | AES Data RW Plaintext/Ciphertext 1 | 1022 |
| 0x068 | AES_DATA_IN_2 | RW | 0x0000.0000 | AES Data RW Plaintext/Ciphertext 2 | 1022 |
| 0x06C | AES_DATA_IN_3 | RW | 0x0000.0000 | AES Data RW Plaintext/Ciphertext 3 | 1022 |
| 0x070 | AES_TAG_OUT_0 | RW | 0x0000.0000 | AES Hash Tag Out 0 | 1023 |
| 0x074 | AES_TAG_OUT_1 | RW | 0x0000.0000 | AES Hash Tag Out 1 | 1023 |
| 0x078 | AES_TAG_OUT_2 | RW | 0x0000.0000 | AES Hash Tag Out 2 | 1023 |
| 0x07C | AES_TAG_OUT_3 | RW | 0x0000.0000 | AES Hash Tag Out 3 | 1023 |
| 0x080 | AES_REVISION | RO | 0x00000041 | AES IP Revision Identifier | 1024 |
| 0x084 | AES_SYSCONFIG | RW | 0x0000.0001 | AES System Configuration | 1025 |
| 0x088 | AES_SYSSTATUS | RO | 0x0000.0001 | AES System Status | 1028 |

**Table 13-5. AES Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x08C | AES_IRQSTATUS | RO | 0x0000.0000 | AES Interrupt Status | 1029 |
| 0x090 | AES_IRQENABLE | RW | 0x0000.0000 | AES Interrupt Enable | 1031 |
| 0x094 | AES_DIRTYBITS | RW1C | 0x0000.0000 | AES Dirty Bits | 1033 |
| **AES µDMA Interrupt Registers (CRC and Cryptographic Modules (CCM) Offset)** | | | | | |
| 0x020 | AES_DMAIM | RW | 0x0000.0000 | AES DMA Interrupt Mask | 1034 |
| 0x024 | AES_DMARIS | RO | 0x0000.0000 | AES DMA Raw Interrupt Status | 1036 |
| 0x028 | AES_DMAMIS | RO | 0x0000.0000 | AES DMA Masked Interrupt Status | 1038 |
| 0x02C | AES_DMAIC | W1C | 0x0000.0000 | AES DMA Interrupt Clear | 1039 |

## 13.6 AES Register Descriptions

This section lists and describes the AES registers, in numerical order by address offset.

### Register 1: AES Key 2_6 (AES_KEY2_6), offset 0x000

### Register 2: AES Key 2_7 (AES_KEY2_7), offset 0x004

### Register 3: AES Key 2_4 (AES_KEY2_4), offset 0x008

### Register 4: AES Key 2_5 (AES_KEY2_5), offset 0x00C

### Register 5: AES Key 2_2 (AES_KEY2_2), offset 0x010

### Register 6: AES Key 2_3 (AES_KEY2_3), offset 0x014

### Register 7: AES Key 2_0 (AES_KEY2_0), offset 0x018

### Register 8: AES Key 2_1 (AES_KEY2_1), offset 0x01C

### Register 9: AES Key 1_6 (AES_KEY1_6), offset 0x020

### Register 10: AES Key 1_7 (AES_KEY1_7), offset 0x024

### Register 11: AES Key 1_4 (AES_KEY1_4), offset 0x028

### Register 12: AES Key 1_5 (AES_KEY1_5), offset 0x02C

### Register 13: AES Key 1_2 (AES_KEY1_2), offset 0x030

### Register 14: AES Key 1_3 (AES_KEY1_3), offset 0x034

### Register 15: AES Key 1_0 (AES_KEY1_0), offset 0x038

### Register 16: AES Key 1_1 (AES_KEY1_1), offset 0x03C

This register contains the 32-bit key data for the AES module. This value can be expanded to 256-bits depending on the mode of operation. Table 13-6 on page 1012 describes the type of key that is held in register.

**Note:** Reads return zeros.

**Table 13-6. AES Key Register Descriptions**

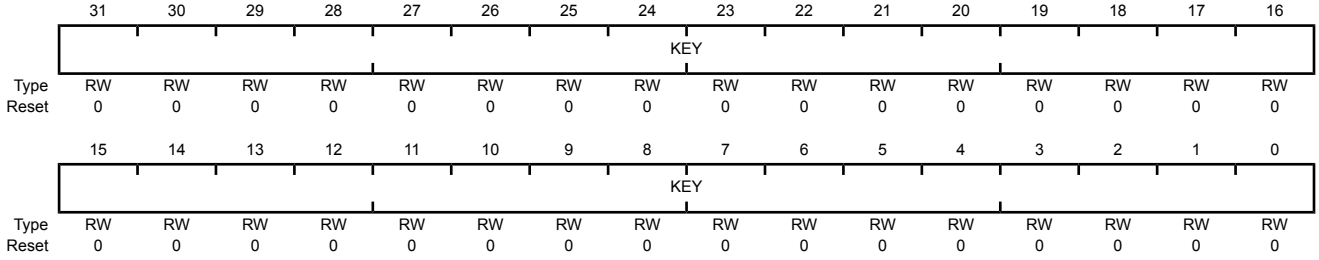
| Register Name | Address Offset | Description |
|---|---|---|
| **AES_KEY2_6** | 0x000 | Secure XTS second key / CBC-MAC third key |
| **AES_KEY2_7** | 0x004 | Secure XTS second key (MSW for 256-bit key) / CBC-MAC third key (MSW) |
| **AES_KEY2_4** | 0x008 | Secure XTS / CCM second key / CBC-MAC third key (LSW) |
| **AES_KEY2_5** | 0x00C | Secure XTS second key (MSW for 192-bit key) / CBC-MAC third key |
| **AES_KEY2_2** | 0x010 | Secure XTS / CCM / CBC-MAC second key / Hash Key input |
| **AES_KEY2_3** | 0x014 | Secure XTS second key (MSW for 128-bit key) + CCM/CBC-MAC second key (MSW) / Hash Key input (MSW) |
| **AES_KEY2_0** | 0x018 | Secure XTS / CCM / CBC-MAC second key (LSW) / Hash Key input (LSW) |
| **AES_KEY2_1** | 0x01C | Secure XTS / CCM / CBC-MAC second key / Hash Key input |
| **AES_KEY1_6** | 0x020 | Secure Key (LSW for 256-bit key) |
| **AES_KEY1_7** | 0x024 | Secure Key (MSW for 256-bit key) |
| **AES_KEY1_4** | 0x028 | Secure Key (LSW for 192-bit key) |
| **AES_KEY1_5** | 0x02C | Secure Key (MSW for 192-bit key) |

**Table 13-6. AES Key Register Descriptions *(continued)***

| Register Name | Address Offset | Description |
|---|---|---|
| **AES_KEY1_2** | 0x030 | Secure Key |
| **AES_KEY1_3** | 0x034 | Secure Key (MSW for 128-bit key) |
| **AES_KEY1_0** | 0x038 | Secure Key (LSW for 128-bit key) |
| **AES_KEY1_1** | 0x03C | Secure Key |

AES Key (AES_KEYn_n)

Base 0x4403.6000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | KEY | RW | 0x00000000 | Key Data<br>This register contains the 32-bit key data for the AES module. |

### Register 17: AES Initialization Vector Input 0 (AES_IV_IN_0), offset 0x040

### Register 18: AES Initialization Vector Input 1 (AES_IV_IN_1), offset 0x044

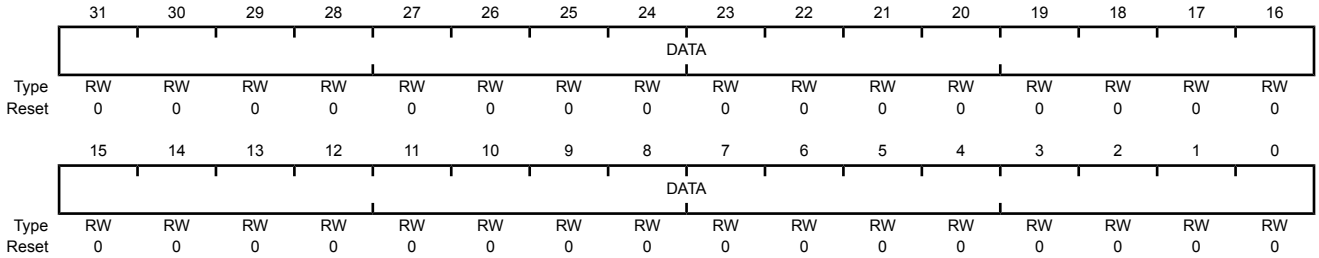
### Register 19: AES Initialization Vector Input 2 (AES_IV_IN_2), offset 0x048

### Register 20: AES Initialization Vector Input 3 (AES_IV_IN_3), offset 0x04C

This register contains the initialization vector input.

AES Initialization Vector Input (AES_IV_IN_n)

Base 0x4403.6000
Offset 0x040
Type RW, reset 0x0000.0000



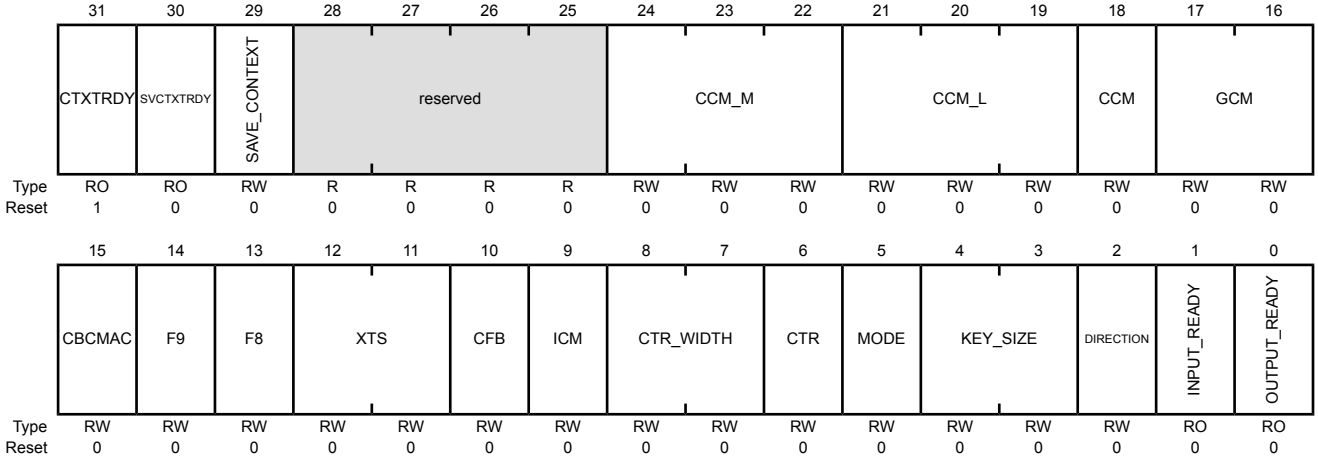

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RW | 0x0000.0000 | Initialization Vector Input<br>This field contains the initialization input vector. The least significant word (LSW) is represented in register **AES_IV_IN_0** and the most significant word is stored in **AES_IV_IN_3** |

## Register 21: AES Control (AES_CTRL), offset 0x050

This register determines the mode of operation of the AES Engine.

AES Control (AES_CTRL)

Base 0x4403.6000
Offset 0x050
Type RW, reset 0x8000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CTXTRDY | RO | 1 | Context Data Registers Ready |

| Value | Description |
|---|---|
| 0 | The context data registers are not ready to be overwritten. |
| 1 | The context data registers can be overwritten and the host is permitted to write the next context. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 30 | SVCTXTRDY | RO | 0 | AES TAG/IV Block(s) Ready |

**Note:** This bit is only asserted if the SAVE_CONTEXT bit is set to 1. This bit is mutual exclusive with the CTXTRDY bit.

| Value | Description |
|---|---|
| 0 | AES authentication TAG and/or IV block(s) is/are not available. |
| 1 | Indicates the AES authentication TAG and /or IV block(s) is/are available for the host to retrieve. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 29 | SAVE_CONTEXT | RW | 0 | TAG or Result IV Save |

If this bit is set, the CONTEXT_OUT interrupt bit is set in the **AES_IRQSTATUS** register if the operation is finished and related signals are enabled.

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Indicates an authentication TAG of result IV needs to be stored as a result context. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 28:25 | reserved | R | 0 | 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. |
| 24:22 | CCM_M | RW | 0 | Counter with CBC-MAC (CCM)<br>Defines M which indicates the length of the authentication field for CCM operations; the authentication field length equals two times the sum of CCM-M plus one.<br>**Note:** The AES Engine always returns a 128-bit authentication field, of which the M least significant bytes are valid. All values are supported. |
| 21:19 | CCM_L | RW | 0 | L Value<br>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. Supported values for L are:<br>Value: Description<br>0x0: width = 0<br>0x1: width = 2<br>0x2: reserved<br>0x3: width = 4<br>0x4 - 0x6: reserved<br>0x7: width = 8 |
| 18 | CCM | RW | 0 | AES-CCM Mode Enable<br>Value: Description<br>0: AES-CCM mode is not enabled.<br>1: AES-CCM mode enabled. This is a combined mode, using AES for both authentication and encryption. No additional mode selection is required. |
| 17:16 | GCM | RW | 0 | AES-GCM Mode Enable<br>This is a combined mode, using the Galois field-multiplier GF($2^{128}$) for authentication and AES-CTR mode for encryption; the bits specify the GCM mode.<br>Value: Description<br>0x0: No operation<br>0x1: GHASH with H loaded and Y0-encrypted forced to zero<br>0x2: GHASH with H loaded and Y0-encrypted calculated internally<br>0x3: Autonomous GHASH (both H and Y0-encrypted calculated internally) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | CBCMAC | RW | 0 | AES-CBC MAC Enable<br>The DIRECTION bit must be set to 1 for this mode.<br><br>Value Description<br>0 AES-CBC MAC mode is not enabled.<br>1 AES-CBC MAC mode enabled. |
| 14 | F9 | RW | 0 | AES f9 Mode Enable<br>The AES key size must be set to 128-bit for this mode.<br><br>Value Description<br>0 f9 mode is not enabled<br>1 f9 mode is enabled. |
| 13 | F8 | RW | 0 | AES f8 Mode Enable<br>The KEY_SIZE must be set to 128-bit for this mode.<br><br>Value Description<br>0 AES f8 mode is not enabled.<br>1 AES f8 mode is enabled. |
| 12:11 | XTS | RW | 0 | AES-XTS Operation Enabled<br>The bits specify the XTS mode.<br><br>Value Description<br>0x0 No operation<br>0x1 Previous/intermediate tweak value and j loaded (value is loaded via IV, j is loaded via the AAD length register)<br>0x2 Key2, n and j are loaded (n is loaded via IV, j is loaded via the AAD length register)<br>0x3 Key2 and n are loaded; j=0 (n is loaded via IV) |
| 10 | CFB | RW | 0 | Full block AES cipher feedback mode (CFB128) Enable<br><br>Value Description<br>0 AES-CFB mode is not enabled.<br>1 AES-CFB mode is enabled. |
| 9 | ICM | RW | 0 | AES Integer Counter Mode (ICM) Enable<br>This is a counter mode with a 16-bit wide counter.<br><br>Value Description<br>0 AES-ICM mode is not enabled.<br>1 AES-ICM mode is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8:7 | CTR_WIDTH | RW | 0 | AES-CTR Mode Counter Width<br>Value Description<br>0x0 Counter is 32 bits<br>0x1 Counter is 64 bits<br>0x2 Counter is 96 bits<br>0x3 Counter is 128 bits |
| 6 | CTR | RW | 0 | Counter Mode<br>This bit must also be set for GCM and CCM mode, when encryption/decryption is required.<br>Value Description<br>0 Counter mode is not enabled.<br>1 Counter mode is enabled. |
| 5 | MODE | RW | 0 | ECB/CBC Mode<br>Value Description<br>0 ECB mode<br>1 CBC mode |
| 4:3 | KEY_SIZE | RW | 0 | Key Size<br>Value Description<br>0x0 reserved<br>0x1 Key is 128 bits<br>0x2 Key is 192 bits<br>0x3 Key is 256 bits |
| 2 | DIRECTION | RW | 0 | Encryption/Decryption Selection<br>If set to DIRECTION=1, an encrypt operation is performed. If set to 0, a decrypt operation is performed.<br>Value Description<br>0 Decryption is selected.<br>1 Encryption is selected. |
| 1 | INPUT_READY | RO | 0 | Input Ready Status<br>Value Description<br>0 Input buffer is not empty.<br>1 Indicates that the 16-byte input buffer is empty, and the host is permitted to write the next block of data. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | OUTPUT_READY | RO | 0 | Output Ready Status |

| Value | Description |
|---|---|
| 0 | No AES output block is available. |
| 1 | An AES output block is available for the host to retrieve. |

## Register 22: AES Crypto Data Length 0 (AES_C_LENGTH_0), offset 0x054

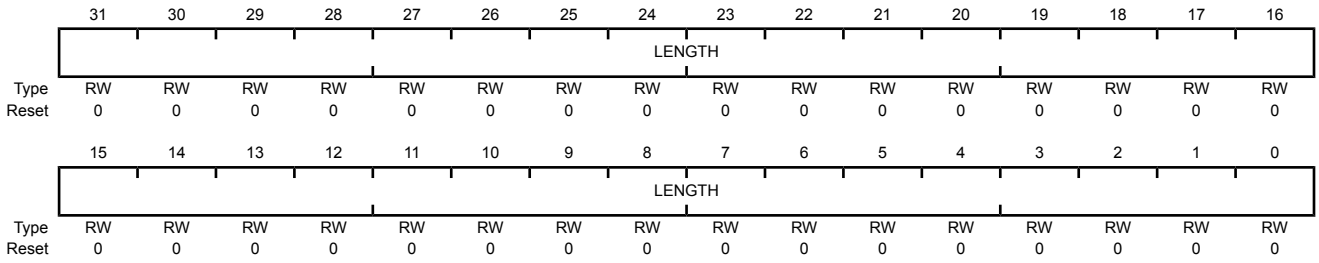
## Register 23: AES Crypto Data Length 1 (AES_C_LENGTH_1), offset 0x058

**AES Crypto Data Length n (AES_C_LENGTH_n)** registers (LSW and MSW) store the cryptographic data length in bytes for all modes. The **AES_C_LENGTH_0** register stores the most significant word and the **AES_C_LENGTH_1** register stores the least significant word. Once processing with this context is started, this length decrements to zero. Data lengths up to ($2^{61}$- 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 $2^{32}$ - 2, resulting in a maximum number of bytes of $2^{36}$ - 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 that for the combined modes, this length does not include the authentication only data; the authentication length is specified in the **AES_AUTH_LENGTH** register below. All modes must have a length greater than 0. For the combined modes, it is permissible to have one of the lengths equal to zero. For the basic encryption modes (ECB/CBC/CTR/ICM/CFB128) it is permissible to program zero to the length field; in that case the length is assumed infinite. All data must be byte (8-bit) aligned; bit aligned data streams are not supported by the AES Engine.

**Note:** For a Host read operation, these registers return all zeroes.

AES Crypto Data Length (AES_C_LENGTH_n)

Base 0x4403.6000
Offset 0x054
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LENGTH | RW | 0x00000000 | Data Length<br>This field stores the cryptographic data length in bytes for all modes. |

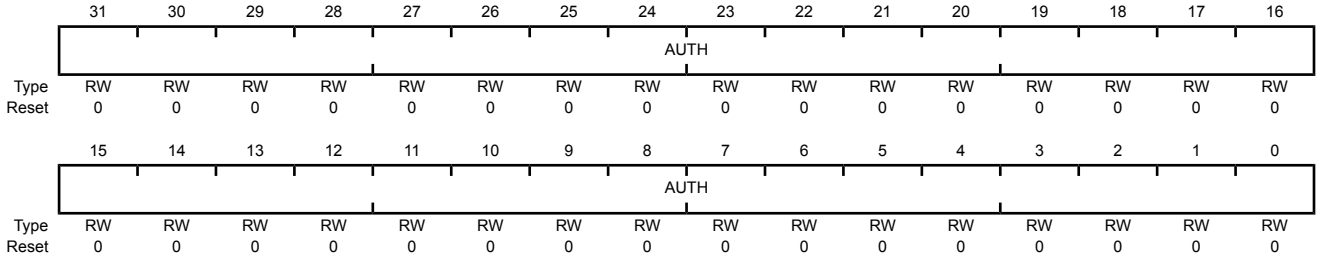
## Register 24: AES Authentication Data Length (AES_AUTH_LENGTH), offset 0x05C

The **AES Authentication Data Length (AES_AUTH_LENGTH)** register stores the authentication data length in bytes for combined modes only (GCM or CCM). Supported AUTH lengths for CCM are from 0 to ($2^{16}$ - 28) 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 zero. A write to this register triggers the engine to start using this context for GCM and CCM. For XTS, this register is optionally used to load j. Loading of j is only required if j ≠ 0. j is a 28-bit value and must be written to bits [31-4] of this register. j represents the sequential number of the 128-bit block inside the data unit. For the first block in a unit, this value is zero. It is not required to provide a j for each new data block within a unit. Note that it is possible to start with a j unequal to zero.

**Note:** For a Host read operation, these registers return all-zeroes..

AES Authentication Data Length (AES_AUTH_LENGTH)

Base 0x4403.6000
Offset 0x05C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | AUTH | RW | 0x00000000 | Authentication Data Length<br>This field stores the authentication data length in bytes for combined modes (GCM or CCM). |

### Register 25: AES Data RW Plaintext/Ciphertext 0 (AES_DATA_IN_0), offset 0x060

### Register 26: AES Data RW Plaintext/Ciphertext 1 (AES_DATA_IN_1), offset 0x064

### Register 27: AES Data RW Plaintext/Ciphertext 2 (AES_DATA_IN_2), offset 0x068

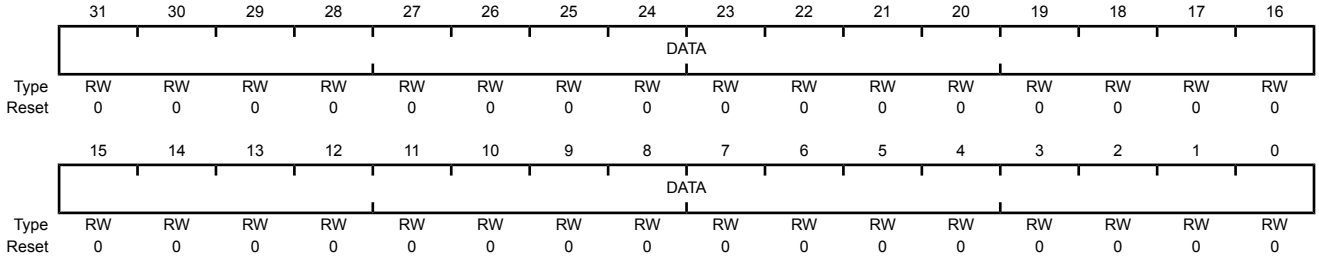
### Register 28: AES Data RW Plaintext/Ciphertext 3 (AES_DATA_IN_3), offset 0x06C

The **AES Data RW Plaintext/Ciphertext n (AES_DATA_IN_n)** registers are used to read and write plaintext/ciphertext. The **AES_DATA_IN_0** register contains the most significant word; the **AES_DATA_IN_3** register contains least significant word.

**Note:** The **AES_DATA_IN_0** register acts as a FIFO and shifts data into the other **AES_DATA_IN_n** registers.

AES Data RW Plaintext/Ciphertext (AES_DATA_IN_n)

Base 0x4403.6000
Offset 0x060
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RW | 0x00000000 | Secure Data RW Plaintext/Ciphertext<br>This field holds the plaintext/ciphertext data. |

### Register 29: AES Hash Tag Out 0 (AES_TAG_OUT_0), offset 0x070

### Register 30: AES Hash Tag Out 1 (AES_TAG_OUT_1), offset 0x074

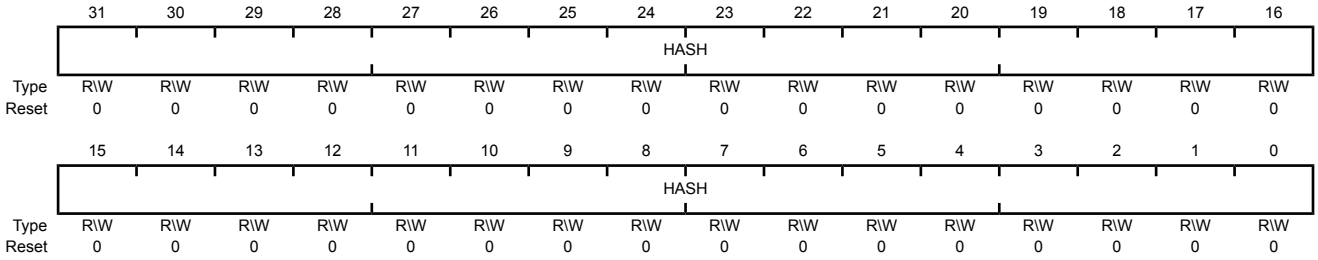
### Register 31: AES Hash Tag Out 2 (AES_TAG_OUT_2), offset 0x078

### Register 32: AES Hash Tag Out 3 (AES_TAG_OUT_3), offset 0x07C

This register displays the Hash result. The **AES_TAG_OUT_0** register is the most significant word of the Hash and the **AES_TAG_OUT_3** register is the least significant word.

AES Hash Tag Out (AES_TAG_OUT_n)

Base 0x4403.6000
Offset 0x070
Type RW, reset 0x0000.0000



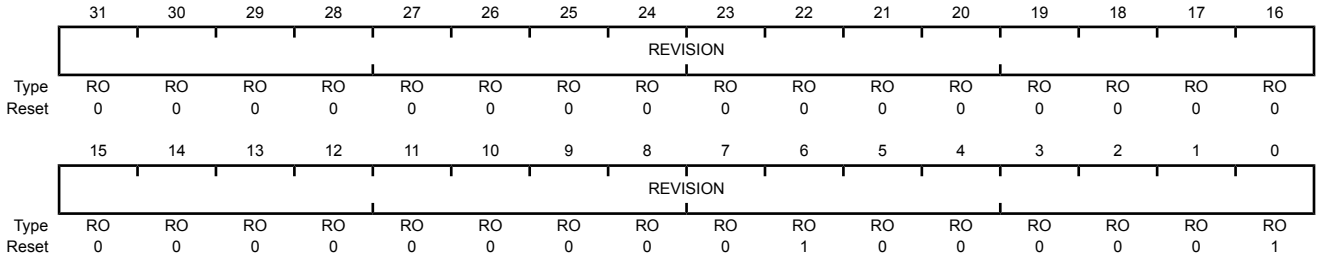

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | HASH | R\W | 0x00000000 | Hash Result<br>This field holds the hash result. |

## Register 33: AES IP Revision Identifier (AES_REVISION), offset 0x080

This register contains the IP revision number of the AES.

AES IP Revision Identifier (AES_REVISION)

Base 0x4403.6000
Offset 0x080
Type RO, reset 0x00000041



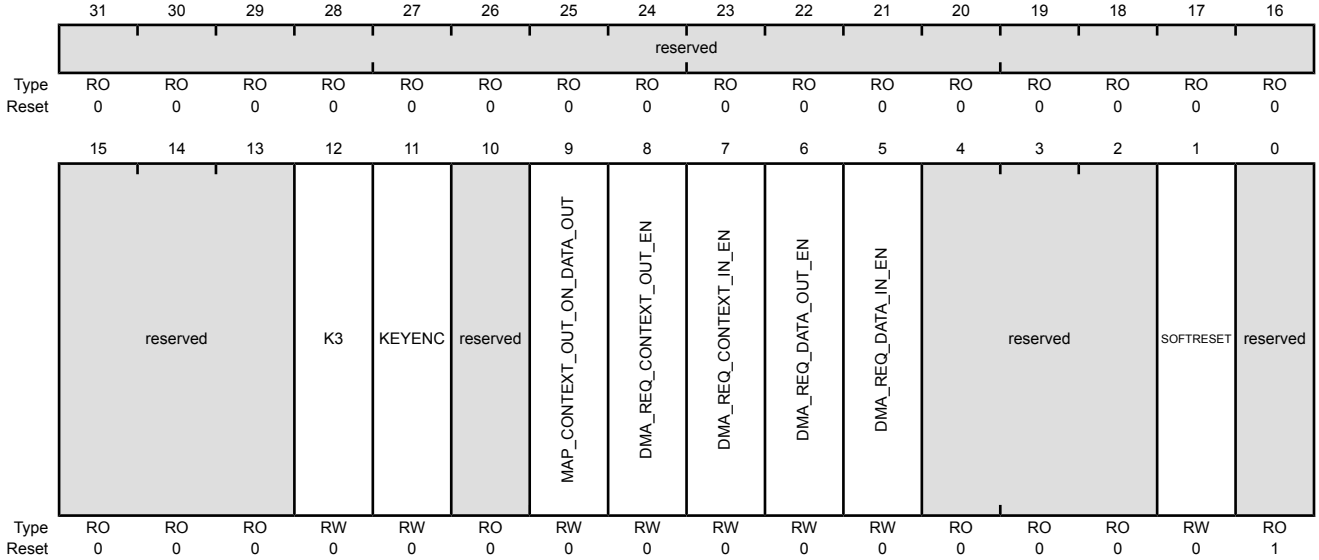

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | REVISION | RO | 0x00000041 | Revision number |

## Register 34: AES System Configuration (AES_SYSCONFIG), offset 0x084

This register controls the IDLE and reset logic.

**Note:** After one operation has completed, the **AES_SYSCONFIG** register must be cleared and re-configured for the next operation to ensure proper DMA and data operation functionality.

AES System Configuration (AES_SYSCONFIG)
Base 0x4403.6000
Offset 0x084
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:13 | reserved | RO | 0 | 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. |
| 12 | K3 | RW | 0 | K3 Select<br>Value Description<br>0 A regular cryptographic operation is performed.<br>1 The K3 key is used as the key for the selected cryptographic operation. The key size should be 128-bits.<br>This bit may only be set to one if the KEYENC bit of this register is cleared to zero. |
| 11 | KEYENC | RW | 0 | Key Encoding<br>Value Description<br>0 A regular cryptographic operation is performed<br>1 The KEK key (see KEKMODE bit) is XOR-ed with a predefined constant value before it is used as a key for the selected cryptographic operation. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | reserved | RO | 0 | 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. |
| 9 | MAP_CONTEXT_OUT_ON_DATA_OUT | RW | 0 | Map Context Out on Data Out Enable<br>Value Description<br>0 Original context out bit values are used.<br>1 The DMA_REQ_CONTEXT_OUT_EN bit in this register and the CONTEXT_OUT bit in the **AES_IRQENABLE** register are mapped on the corresponding DATA_OUT request bits. In this case, the original CONTEXT_OUT bit values are ignored. |
| 8 | DMA_REQ_CONTEXT_OUT_EN | RW | 0 | DMA Request Context Out Enable<br>If set to '1', the DMA context output request is enabled (for context data out, for example, TAG for authentication modes). The dma_done indication bits in **AES_DMARIS** register, at CCM module offset 0x024, identify to the application when the DMA transfer is complete.<br>Value Description<br>0 DMA disabled for context output request.<br>1 DMA enabled for context output request. |
| 7 | DMA_REQ_CONTEXT_IN_EN | RW | 0 | DMA Request Context In Enable<br>The dma_done indication bits in **AES_DMARIS** register, at CCM offset 0x024, identify to the application when the DMA transfer is complete.<br>Value Description<br>0 DMA disabled for context input request.<br>1 DMA enabled for context input request. |
| 6 | DMA_REQ_DATA_OUT_EN | RW | 0 | DMA Request Data Out Enable<br>The dma_done indication bits in **AES_DMARIS** register, at CCM offset 0x024, identify to the application when the DMA transfer is complete.<br>Value Description<br>0 DMA disabled for data output request.<br>1 DMA enabled for data output request. |
| 5 | DMA_REQ_DATA_IN_EN | RW | 0 | DMA Request Data In Enable<br>The dma_done indication bits in **AES_DMARIS** register, at CCM module offset 0x024, identify to the application when the DMA transfer is complete.<br>Value Description<br>0 DMA disabled for data input request.<br>1 DMA enabled for data input request. |

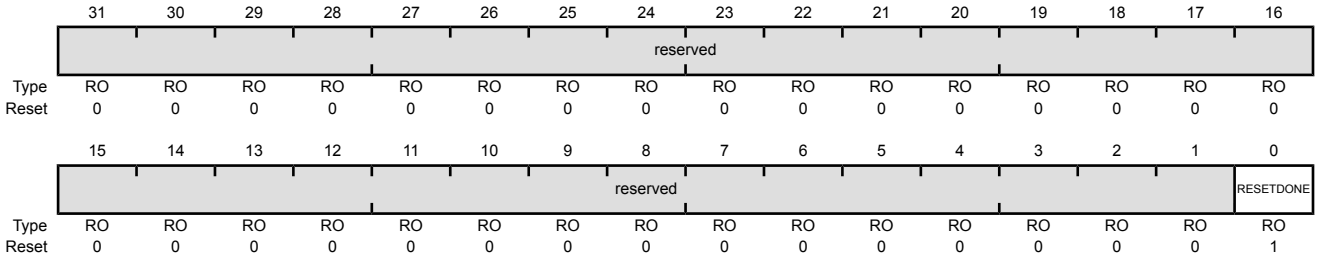
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4:2 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | SOFTRESET | RW | 0 | Soft reset<br>Value Description<br>0 No operation<br>1 Start soft reset sequence |
| 0 | reserved | RO | 1 | 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. |

## Register 35: AES System Status (AES_SYSSTATUS), offset 0x088

This register indicates if reset has completed.

AES System Status (AES_SYSSTATUS)

Base 0x4403.6000
Offset 0x088
Type RO, reset 0x0000.0001



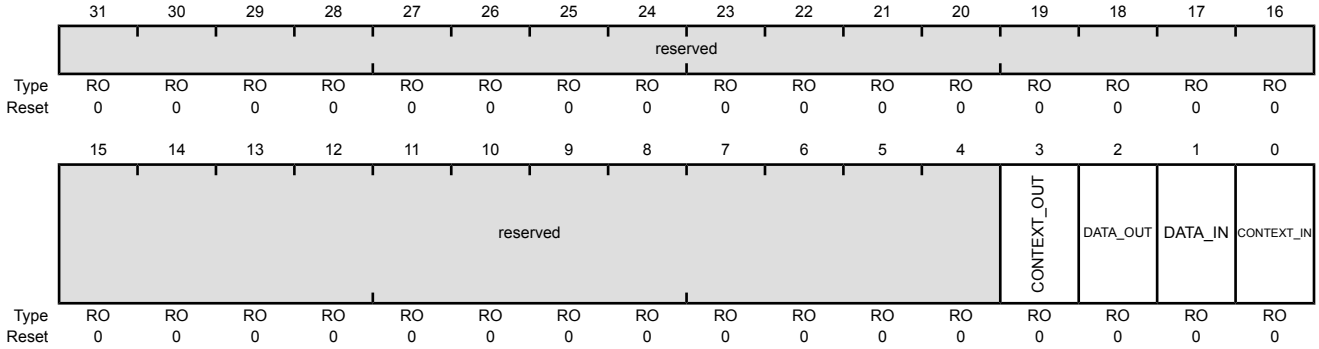

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | RESETDONE | RO | 1 | Reset Done<br>Value Description<br>0 Reset is not complete.<br>1 Reset is has completed. |

## Register 36: AES Interrupt Status (AES_IRQSTATUS), offset 0x08C

This register indicates the interrupt status. If one of the interrupt bits is set, the interrupt output is asserted.

AES Interrupt Status (AES_IRQSTATUS)

Base 0x4403.6000
Offset 0x08C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | CONTEXT_OUT | RO | 0 | Context Output Interrupt Status<br><br>Value Description<br>0 Authentication tag (and IV) interrupt(s) is/are not active.<br>1 Authentication tag (and IV) interrupt(s) is/are active and the interrupt output has been triggered. |
| 2 | DATA_OUT | RO | 0 | Data Out Interrupt Status<br><br>Value Description<br>0 The data out interrupt is not active.<br>1 The data out interrupt is active and the interrupt output has been triggered. |
| 1 | DATA_IN | RO | 0 | Data In Interrupt Status<br><br>Value Description<br>0 The data in interrupt is not active.<br>1 The data in interrupt is active and the interrupt output has been triggered. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CONTEXT_IN | RO | 0 | Context In Interrupt Status |

| Value | Description |
|---|---|
| 0 | The context in interrupt is not active. |
| 1 | The context in interrupt is active and the interrupt output has been triggered. |

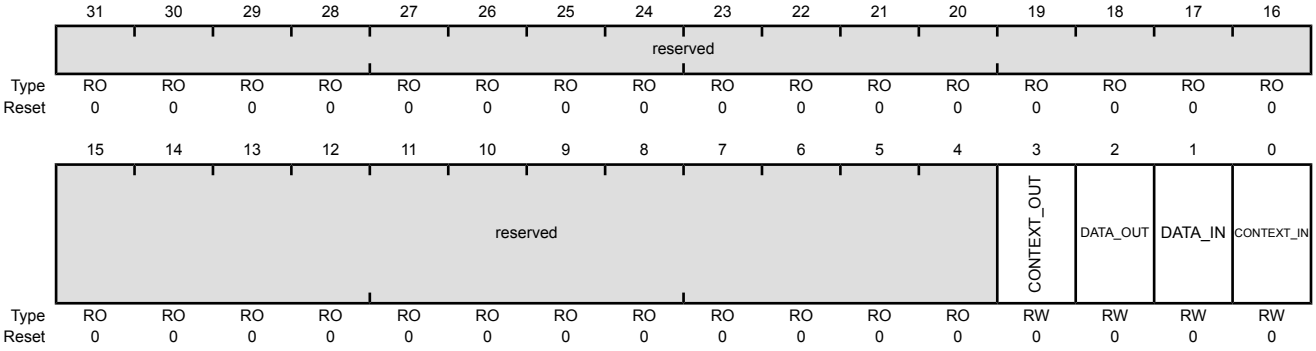
## Register 37: AES Interrupt Enable (AES_IRQENABLE), offset 0x090

This register contains an enable bit for each unique interrupt generated by the module. It matches the layout of **AES_IRQSTATUS** register. An interrupt is enabled when the bit in this register is set to 1. An interrupt that is enabled is propagated to the NVIC controller. All AES software interrupts need to be enabled explicitly by writing this register.

**Note:** If the application uses Interrupt Mode, an interrupt is generated for each block of processed data. To support larger data flow, AES µDMA Mode should be used and the bits in the **AES_IRQENABLE** register should be cleared.

AES Interrupt Enable (AES_IRQENABLE)

Base 0x4403.6000
Offset 0x090
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | CONTEXT_OUT | RW | 0 | Context Out Interrupt Enable<br>Value Description<br>0 Authentication tag (and IV) interrupt(s) is/are disabled.<br>1 Authentication tag (and IV) interrupt(s) is/are enabled. |
| 2 | DATA_OUT | RW | 0 | Data Out Interrupt Enable<br>Value Description<br>0 The data out interrupt is disabled.<br>1 The data out interrupt is enabled. |
| 1 | DATA_IN | RW | 0 | Data In Interrupt Enable<br>Value Description<br>0 The data in interrupt is disabled.<br>1 The data in interrupt is enabled. |

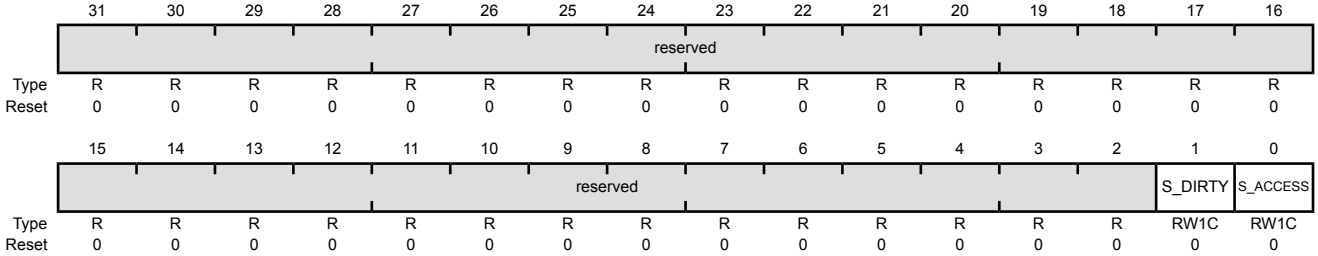
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CONTEXT_IN | RW | 0 | Context In Interrupt Enable |

| Value | Description |
|---|---|
| 0 | The context in interrupt is disabled. |
| 1 | The context in interrupt is enabled. |

### Register 38: AES Dirty Bits (AES_DIRTYBITS), offset 0x094

This register can be used to identify if AES registers have been read or written to.

AES Dirty Bits (AES_DIRTYBITS)
Base 0x4403.6000
Offset 0x094
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | R | 0 | 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. |
| 1 | S_DIRTY | RW1C | 0 | AES Dirty Bit<br>This bit must be written to a 1 to clear.<br>Value Description<br>0 No AES registers have been written.<br>1 Indicates when any of the **AES_x** registers have been written (except for the **AES_DIRTYBITS** register). |
| 0 | S_ACCESS | RW1C | 0 | AES Access Bit<br>This bit must be written to a 1 to clear.<br>Value Description<br>0 No AES registers have been read.<br>1 Indicates when any of the **AES_x** registers have been read (except for the **AES_DIRTYBITS** register). |

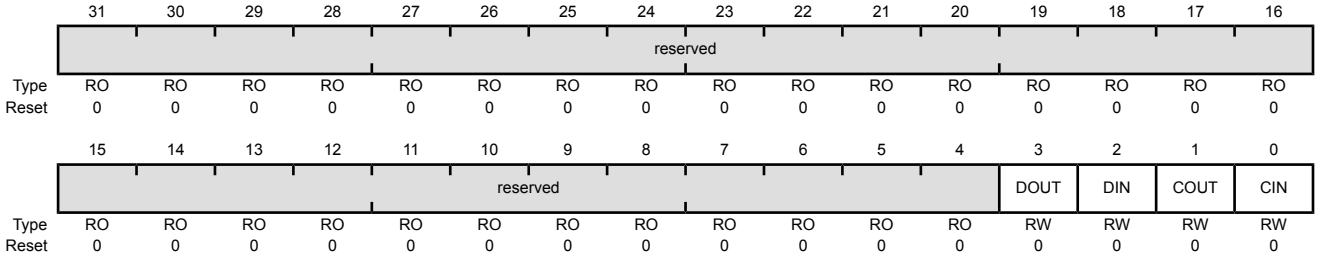
## 13.7 AES µDMA Interrupt Register Descriptions (CCM Offset)

This section lists and describes the AES µDMA registers, in numerical order by address offset. Registers in this section are relative to the base address of 0x4403.0000.

## Register 39: AES DMA Interrupt Mask (AES_DMAIM), offset 0x020

The **AES DMA Interrupt Mask (AES_DMAIM)** register controls interrupt behavior and is used to program which interrupts are suppressed.

AES DMA Interrupt Mask (AES_DMAIM)
Base 0x4403.0000
Offset 0x020
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | DOUT | RW | 0 | Data Out DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA writes the last word of the process result.<br>Value Description<br>0 The DOUT interrupt is suppressed and not sent to the interrupt controller.<br>1 The DOUT interrupt is sent to the interrupt controller. |
| 2 | DIN | RW | 0 | Data In DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA writes the last word of input data to the internal FIFO of the engine.<br>Value Description<br>0 The DIN interrupt is suppressed and not sent to the interrupt controller.<br>1 The DIN interrupt is sent to the interrupt controller. |
| 1 | COUT | RW | 0 | Context Out DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA completes the output context read from the internal register.<br>Value Description<br>0 The COUT interrupt is suppressed and not sent to the interrupt controller.<br>1 The COUT interrupt is sent to the interrupt controller. |

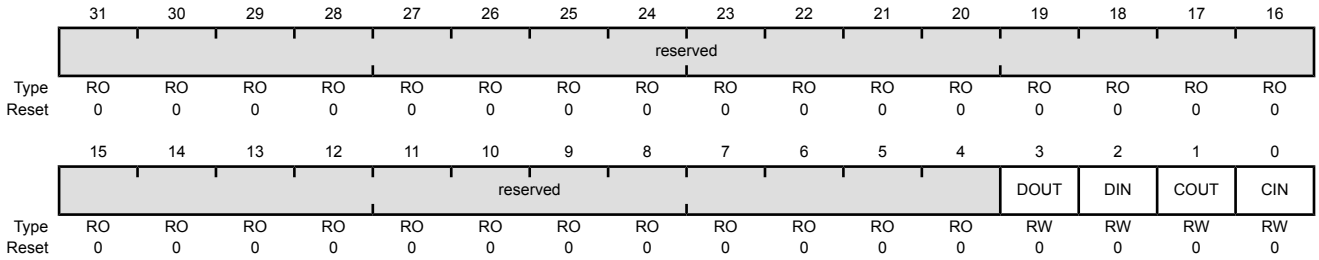
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CIN | RW | 0 | Context In DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA completes a context write to the internal register.<br><br>Value Description<br>0 The CIN interrupt is suppressed and not sent to the interrupt controller.<br>1 The CIN interrupt is sent to the interrupt controller. |

## Register 40: AES DMA Raw Interrupt Status (AES_DMARIS), offset 0x024

The **AES DMA Raw Interrupt Status (AES_DMARIS)** register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set to '1.'

AES DMA Raw Interrupt Status (AES_DMARIS)

Base 0x4403.0000
Offset 0x024
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | DOUT | RW | 0 | Data Out DMA Done Raw Interrupt Status<br><br>Value Description<br>0 No Interrupt.<br>1 The µDMA has written the last word of the process result and an interrupt has been triggered and is pending. |
| 2 | DIN | RW | 0 | Data In DMA Done Raw Interrupt Status<br><br>Value Description<br>0 No Interrupt.<br>1 The µDMA has written the last word of input data to the internal FIFO of the engine and an interrupt has been triggered and is pending. |
| 1 | COUT | RW | 0 | Context Out DMA Done Raw Interrupt Status<br><br>Value Description<br>0 No Interrupt.<br>1 The µDMA has completed the output context read from the internal register and an interrupt has been triggered and is pending. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CIN | RW | 0 | Context In DMA Done Raw Interrupt Status |

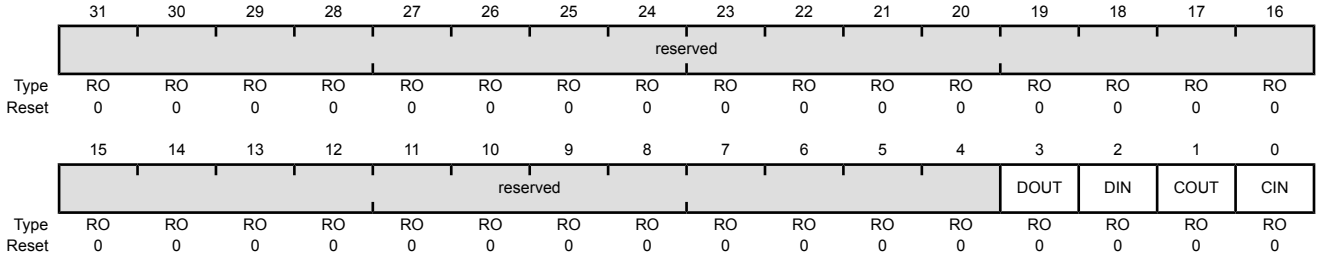
| Value | Description |
|---|---|
| 0 | No interrupt. |
| 1 | The µDMA has completed a context write to the internal register and an interrupt has been triggered and is pending. |

## Register 41: AES DMA Masked Interrupt Status (AES_DMAMIS), offset 0x028

The **AES DMA Masked Interrupt Status (AES_DMAMIS)** register displays the raw interrupts that are unmasked in the **AES DMA Raw Interrupt Status (AES_DMARIS)** register.

AES DMA Masked Interrupt Status (AES_DMAMIS)

Base 0x4403.0000
Offset 0x028
Type RO, reset 0x0000.0000



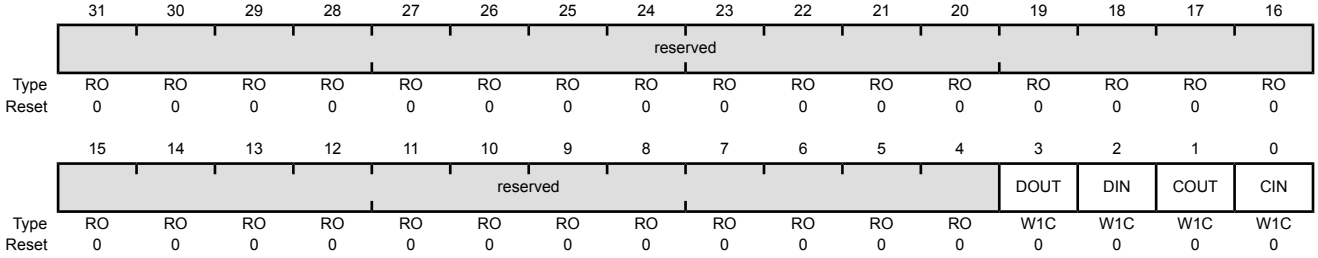

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | DOUT | RO | 0 | Data Out DMA Done Masked Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A DOUT interrupt has occurred. |
| 2 | DIN | RO | 0 | Data In DMA Done Masked Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A DIN interrupt has occurred. |
| 1 | COUT | RO | 0 | Context Out DMA Done Masked Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A COUT interrupt has occurred. |
| 0 | CIN | RO | 0 | Context In DMA Done Raw Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A CIN interrupt has occurred. |

## Register 42: AES DMA Interrupt Clear (AES_DMAIC), offset 0x02C

The **AES DMA Interrupt Clear (AES_DMAIC)** register is used to clear the **AES_DMARIS** and **AES_DMAMIS** registers by writing a 1 to each register bit.

**Note:** This registers always reads as zero.

AES DMA Interrupt Clear (AES_DMAIC)
Base 0x4403.0000
Offset 0x02C
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | DOUT | W1C | 0 | Data Out DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the `DOUT` bit in the **AES_DMARIS** and **AES_DMAMIS** register. |
| 2 | DIN | W1C | 0 | Data In DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the `DIN` bit in the **AES_DMARIS** and **AES_DMAMIS** register. |
| 1 | COUT | W1C | 0 | Context Out DMA Done Masked Interrupt Status<br>Writing a 1 to this bit clears the `COUT` bit in the **AES_DMARIS** and **AES_DMAMIS** register. |
| 0 | CIN | W1C | 0 | Context In DMA Done Raw Interrupt Status<br>Writing a 1 to this bit clears the `CIN` bit in the **AES_DMARIS** and **AES_DMAMIS** register. |

# 14 Data Encryption Standard Accelerator (DES)

The DES module provides hardware accelerated data encryption and decryption functions. The module runs either the single DES or the triple DES (3DES) algorithm in compliance with the FIPS 46-3 standard and supports electronic codebook (ECB), cipher block chaining (CBC), and cipher feedback (CFB) modes of operation. It does not support the output feedback (OFB) mode of operation in hardware.

The purpose of the DES algorithm is to encrypt (encipher) or decrypt (decipher) binary coded information. Encrypting data converts it to an unintelligible form called cipher text. Decrypting cipher text converts the data back to its original form called plaintext. DES is a symmetrical algorithm in that the encryption and decryption keys are identical. Each triple DES encrypt/decrypt operation is a compound of DES encrypt and decrypt operations.

The DES accelerator includes the following main features:

- DES/3DES encryption and decryption.
- Feedback modes: ECB, CBC, CFB
- Host interrupt or µDMA driven modes of operation. µDMA support for data and context in/result out
- Fully synchronous design
- Internal wide-bus interface

## 14.1 DES Functional Description

The DES module is an efficient implementation of a DES block cipher. Block ciphers, as opposed to stream ciphers, operate on blocks of plain text and cipher text. The DES block size is 8-byte. The DES key consists of 64 binary digits, but only 56 bits are actually used directly by the algorithm. The other 8 bits are used for error detection.

The 64-bit block of input data to be enciphered is initially permuted, then passed through 16 iterations of a calculation that uses a cipher function and finally permutated to the inverse of the initial permutation. At each of the 16 iterations, a 48-bit key computed from the 64-bit input key is applied to one of the 32-bit sub-blocks of the 64-bit input block using the cipher function. The 48-bit key value changes for each iteration. The result of the cipher function is a 32-bit sub-block, which is concatenated with the second 32-bit input sub-block. The resulting 64-bit output block of each iteration feeds back as the input of the next iteration. To decipher, it is only necessary to apply the same algorithm to the enciphered message block, taking care that each iteration of the computation will use the same 48-bit key which was used during enciphering.

The triple DES is the DES used three times in a row (also known as DES-EDE). It uses three keys key1, key2 , and key3 , so that key length is 168 bits effective: a 64-bit block plaintext is encrypted with key1 , decrypted with key2, and encrypted with key3, and a 64-bit ciphertext is decrypted with key1 , encrypted with key2 , and decrypted with key3 .

There are three keying options defined in ANSI X9.52 for DES-EDE:

- The three keys key1, key2, and key3 are independent.
- key1 and key2 are independent, but key1 = key3
- key1 = key2 = key3

The first option provides highest level of security; the last option is compatible with single DES. See Table 14-1 on page 1041 for key use.

**Table 14-1. Key Repartition**

| Mode | Key1_L | Key1_H | Key2_L | Key2_H | Key3_L | Key3_H |
|---|---|---|---|---|---|---|
| 64-bit (DES) | √ | √ | X | X | X | X |
| 192-bit (3DES) | √ | √ | √ | √ | √ | √ |

ECB, CBC, and CFB modes can be used with DES and 3DES modes.

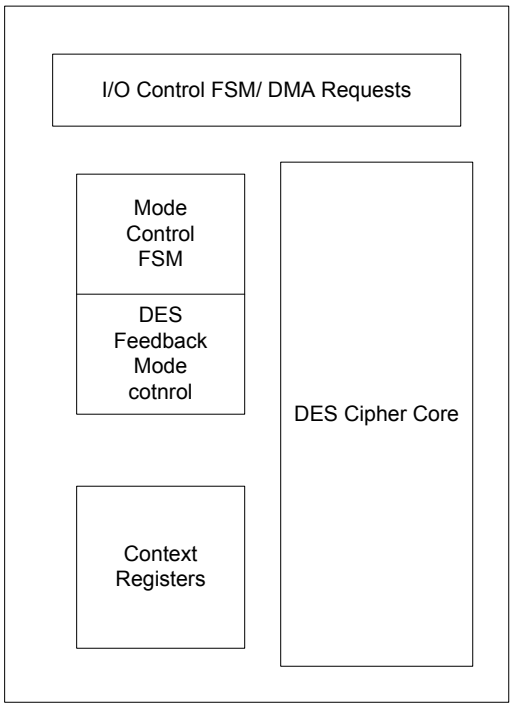
## 14.2 DES Block Diagram

The module architecture consists of primary blocks, as shown in Figure 14-1 on page 1041. The DES module includes a register interface, and a µDMA and interrupt interface.

Depending of the availability of context and data, the DES engine is automatically triggered to process the data. The DES engine is directly connected to the context and data registers such that it can immediately start processing when all data is available.

Packets (blocks of 64 bits) must be parsed into blocks and sequentially fed into the DES, which can buffer the block currently being processed as well as an additional block that may be queued in advance.

**Figure 14-1. DES Block Diagram**




### 14.2.1 µDMA Control

The µDMA and interrupt request logic is controlled by the DES Engine. The DES Engine can have multiple µDMA request signals active in parallel.

There are three DMA channels available:

- DMA context in - Request a new context (DES0 Cin)
- DMA data in - Request input data (DES0 Din)
- DMA data out - Request output data read (DES0 Dout)

### 14.2.2 Interrupt Control

There is one interrupt for the DES that is sent to the interrupt controller. This interrupt is an OR of the enabled interrupt bits in the **DES Interrupt Status (DES_IRQSTATUS)** register. These bits are enabled through the **DES Interrupt Enable (DES_IRQENABLE)** register. The following events can generate an interrupt bit to be set in the **DES_IRQSTATUS** register. These are:

- New context input required
- Data input required
- Data output ready

### 14.2.3 Register Interface

The Register Interface block performs all address decoding and control; however, not all registers are available in this block. The context and data input registers are in the DES engine.

### 14.2.4 DES Engine

The DES buffered engine consists of the following major functional blocks:

- Cipher core: the DES algorithm
- Mode control FSM: manages the data flow to and from the DES buffered engine and starts each encrypt/decrypt operation
- DES feedback mode block: the logic that implements the various feedback modes supported by the DES buffered engine

#### 14.2.4.1 Mode Control FSM

The Mode control FSM manages the data flow to and from the DES engine. This block also sends a start pulse to the encrypt/decrypt core and triggers the core to use the new mode keys when a new context is needed. This module also controls the 3DES operation, such that the DES core module is triggered three times (with different keys) before the result data becomes available.

#### 14.2.4.2 DES Feedback Mode Block

The DES feedback mode block buffers the input and output blocks and contains all logic to implement the 3DES and various feedback modes. See the FIPS-81 document for details on the ECB, CBC, and CFB modes of operation.

By itself, the DES cipher core outputs data compliant for ECB encryption. However, most applications use DES with feedback. Feedback provides additional security by randomizing repeated patterns in the plain text, which could otherwise be exploited to attack the cipher text. The DES buffered engine supports ECB, CBC, and CFB modes of operations for the DES and 3DES algorithm.

3DES mode performs the DES algorithm three times on a single block and uses a different key for each invocation of the DES algorithm, greatly increasing security of the cipher text but at the cost

of a 3x reduction in throughput. The DES buffered engine implements a 3DES logic wrapper around the 2-round DES core, enabling seamless 3DES encryption.

#### 14.2.4.3 DES Cipher Core

The DES cipher core implements the DES algorithm as specified in the FIPS 46-3. The core operates on the input block and performs the required substitution, shift, and mix operations. The core also applies the correct key-scheduling.

Inherently, considerable parallelism is possible with the DES algorithm. This is exploited in two ways. For high performance, the 64 bits composing the data block are processed concurrently (4-round implementation). For low gate-count, resources are shared on both the main data and key paths (1-round implementation).

A fundamental component of the DES algorithm is the substitution box (S-Box). The S-Box provides a unique 4-bit output for each 6-bit input. The S-Box design is a primary factor for both performance and gate count. The DES Cipher Core has a standard lookup table S-Box that allows room for the synthesizer to optimize on timing or gate count.

## 14.3 Software Reset

Table 14-2 on page 1043 lists the resets used in the DES module.

**Table 14-2. DES Reset Description**

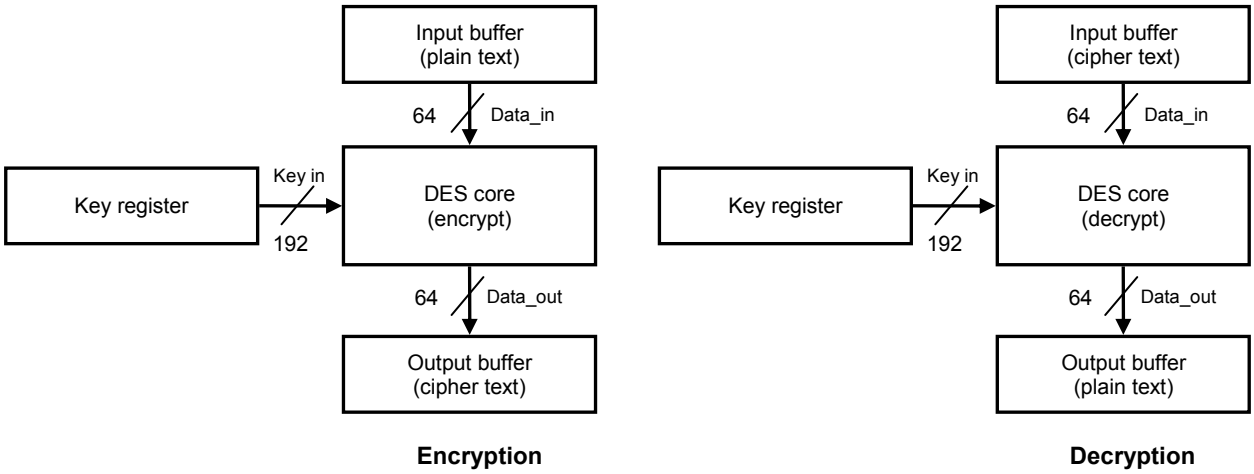
| Type | Name | Source | Description |
|---|---|---|---|
| Hardware | DES_RST | PRCM | Global reset |
| Software | DES_SYSCONFIG [1] SOFTRESET | Internal | Starts the soft reset sequence. When the DES_SYSSTATUS/DES_P_SYSSTATUS[0] RESETDONE bit goes to 1, the soft reset sequence is finished. |

## 14.4 DES Supported Modes of Operation

### 14.4.1 ECB Feedback Mode

Figure 14-2 on page 1044 shows the basic ECB feedback mode of operation, where the input data is passed directly to the basic cryptographic core and the output of the cryptographic core is passed directly to the output buffer. For decryption the DES core operates in reverse, this means the decrypt key sequence is used for the data processing, where encryption uses the encrypt key sequence.

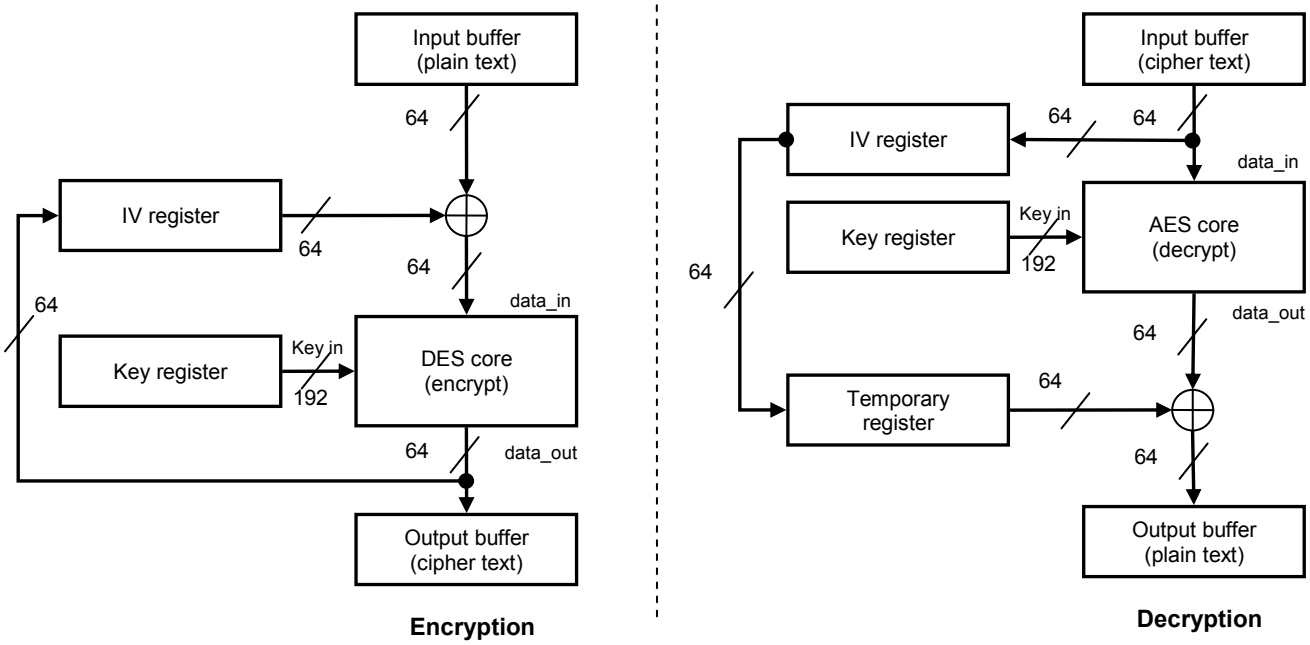
**Figure 14-2. DES - ECB Feedback Mode**




### 14.4.1.1 CBC Feedback Mode

Figure 14-3 on page 1044 shows the CBC feedback mode of operation, where the input data is XORed with the initialization vector (IV) before it is passed to the basic crypto core. The output of the crypto core is passed directly to the output buffer. For decryption the operation is reversed, resulting in an XOR at the output of the crypto core.

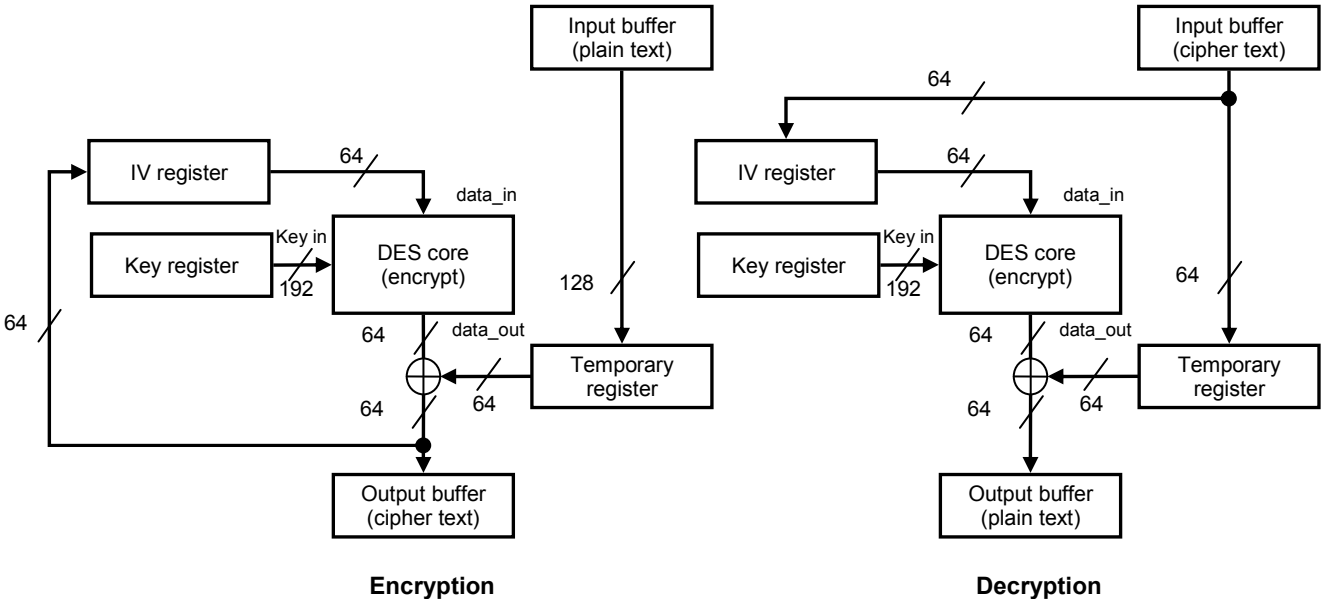
**Figure 14-3. DES3DES - CBC Feedback Mode**




### 14.4.1.2 CFB Feedback Mode

Figure 14-4 on page 1045 shows the CFB mode of operation for encryption and decryption. The input for the crypto core is the IV; the result is XORed with the data. The result is fed back via the IV register, as the next input for the crypto core. The decrypt operation is reversed, but the crypto core is still performing an encryption.

**Figure 14-4. DES3DES-CFB Feedback Mode**




# 14.5 DES Module Programming Guide -Low Level Programming Models

## 14.5.1 Surrounding Modules Global Initialization

### 14.5.1.1 Main Sequence - DES Global Initialization

This procedure initializes the DES after a POR.

**Table 14-3. DES Global Initialization**

| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Execute software reset. | DES_SYSCONFIG[1] SOFTRESET | 0x1 |
| Wait for reset to complete | DES_SYSSTATUS[0] RESETDONE | = 0x1 |
| Select force idle mode | DES_SYSCONFIG[3:2] SIDLE | 0x0 |
| Select the algorithm type (DES or 3DES) | See the programming models | |
| Select the operating mode (ECB, CBC or CFB). | DES_CTRL[5:4] MODE | - |
| IF: ECB operating mode not selected. | DES_CTRL[5:4] MODE | ≠ 0x0 |
| Load the initialization vector LSW. | DES_IV_L[31:0] IV_L | - |
| Load the initialization vector MSW. | DES_IV_H[31:0] IV_H | - |
| Define the cryptographic data length. | DES_LENGTH[31:0] LENGTH | - |
| ENDIF | | |
| Select encryption or decryption. | DES_CTRL[2] DIRECTION | - |

### 14.5.1.2 Subsequence - Configure the DES Algorithm Type

This subsequence details the DES algorithm type settings.

**Table 14-4. DES Algorithm Type Configuration**

| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Load key 1 LSW. | DES_KEY1_L[31:0] KEY1_L | - |
| Load key 1 MSW. | DES_KEY1_H[31:0] KEY1_H | - |
| Select DES algorithm. | DES_CTRL[3] TDES | 0x0 |

#### 14.5.1.3 Subsequence - Configure the 3DES Algorithm Type

This subsequence details the 3DES algorithm type settings.

**Table 14-5. 3DES Algorithm Type Configuration**

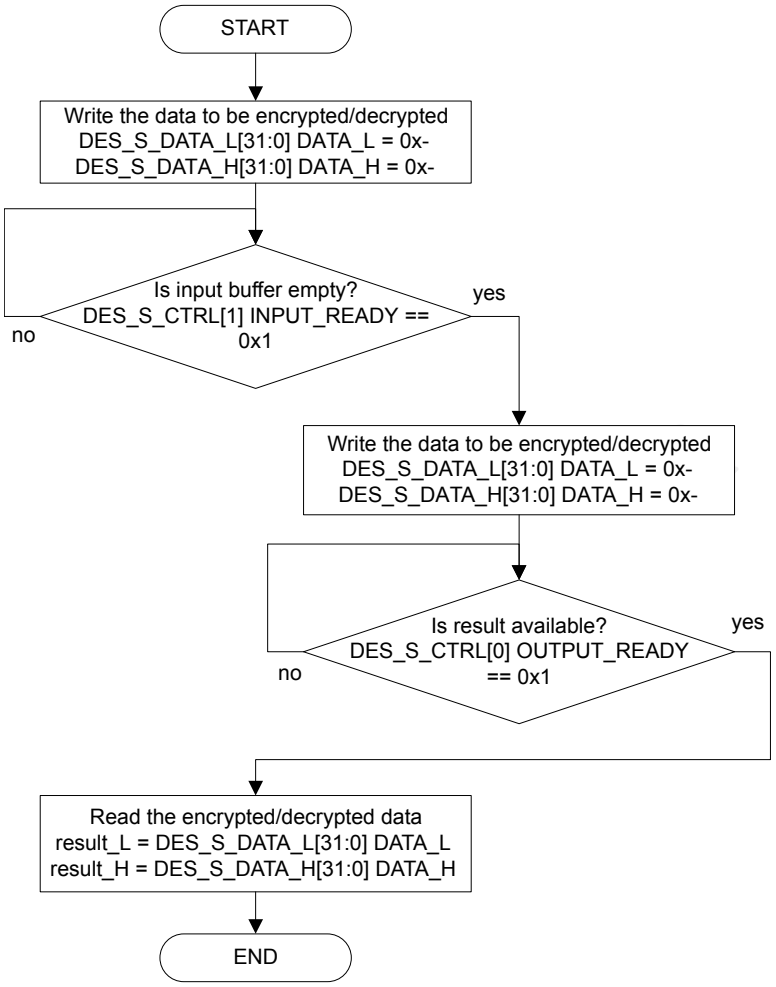
| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Load key 1 LSW. | DES_KEY1_L[31:0] KEY1_L | - |
| Load key 1 MSW. | DES_KEY1_H[31:0] KEY1_H | - |
| Load key 2 LSW. | DES_KEY2_L[31:0] KEY2_L | - |
| Load key 2 MSW. | DES_KEY2_H[31:0] KEY2_H | - |
| Load key 3 LSW. | DES_KEY3_L[31:0] KEY3_L | - |
| Load key 3 MSW. | DES_KEY3_H[31:0] KEY3_H | - |
| Select 3DES algorithm. | DES_CTRL[3] TDES | 0x1 |

### 14.5.2 Operational Modes Configuration

#### 14.5.2.1 Main Sequence - DES Polling Mode

Figure 14-5 on page 1047 shows DES polling mode. The registers used in DES polling mode are: DES_DATA_L, DES_DATA_H, and DES_CTRL.

**Figure 14-5. DES Polling Mode**




### 14.5.2.2 DES Interrupt Mode

Table 14-6 on page 1047 lists the DES interrupt mode steps.

**Table 14-6. DES Interrupt Mode**

| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Enable DES module interrupts. | DES_IRQENABLE[2:0] | 0x7 |
| Load the input buffer data LSW register. | DES_DATA_L[31:0] DATA_L | - |
| Load the input buffer data HSW register. | DES_DATA_H[31:0] DATA_H | - |

### 14.5.2.3 DES Interrupt DMA Mode

Table 14-7 on page 1047 lists the DES DMA mode steps.

**Table 14-7. DES DMA Mode**

| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Enable DES module DMA requests. | DES_SYSCONFIG[7:5] | 0x7 |
| Load the input buffer data LSW register. | DES_DATA_L[31:0] DATA_L | - |

**Table 14-7. DES DMA Mode *(continued)***

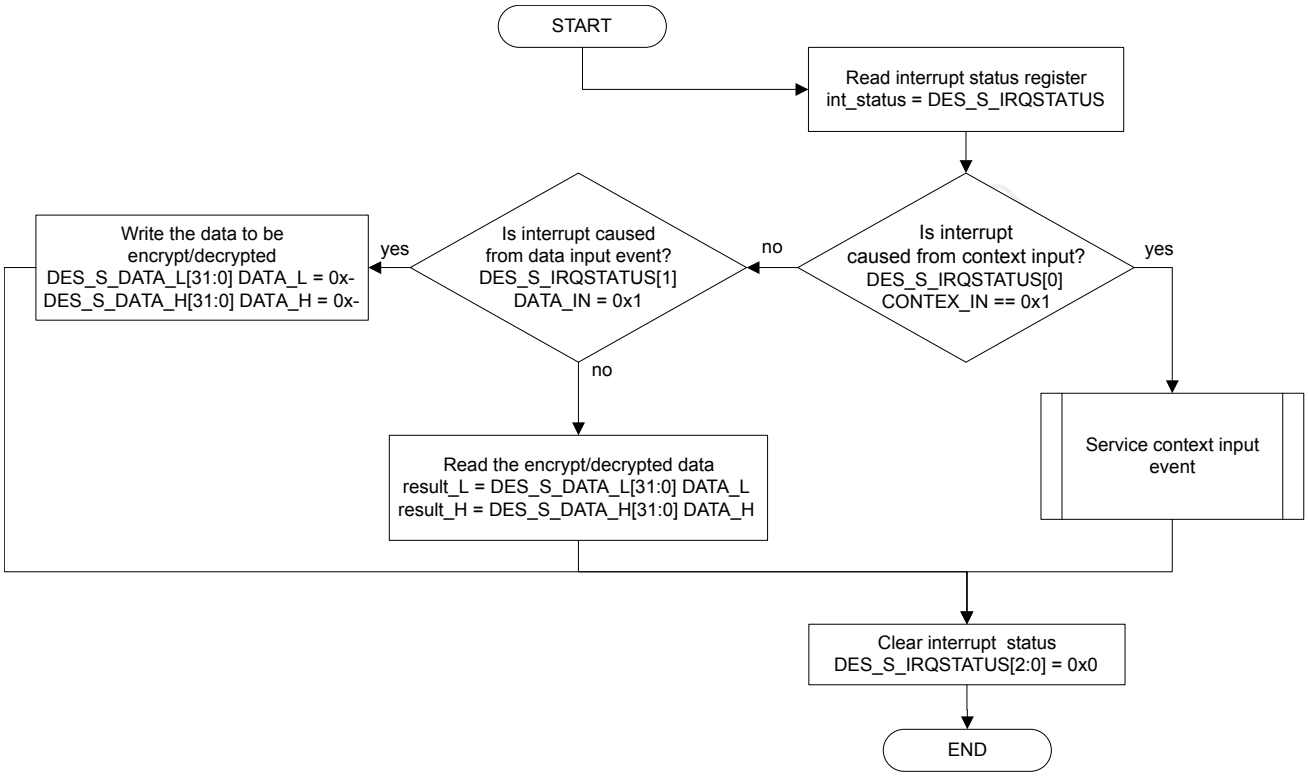
| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Load the input buffer data HSW register. | DES_DATA_H[31:0] DATA_H | - |

## 14.5.3 DES Events Servicing

### 14.5.3.1 Interrupt Servicing

This section describes the event servicing of the module. Figure 14-6 on page 1048 shows the DES interrupt service. The registers used during event servicing are: DES_IRQSTATUS, DES_DATA_L, and DES_DATA_H.

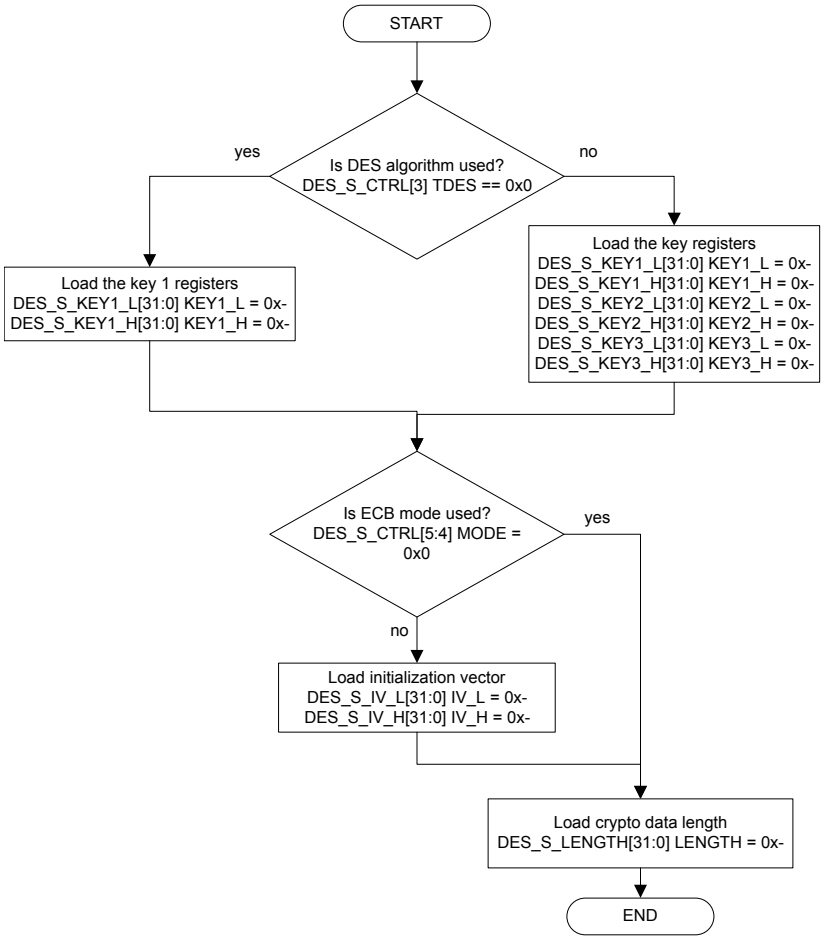
**Figure 14-6. DES Interrupt Service**




### 14.5.3.2 Context Input Event Servicing

This section describes the context input event servicing of the module, as shown in Figure 14-7 on page 1049. The registers used during event servicing are: DES_CTRL, DES_SYSCONFIG, DES_KEY1_L, DES_KEY1_H, DES_KEY2_L, DES_KEY2_H, DES_KEY3_L, DES_KEY3_H, DES_IV_L, DES_IV_H, and DES_LENGTH.

**Figure 14-7. DES Context Input Event Service**




## 14.6 Register Map

Table 14-8 on page 1049 lists the DES registers. The DES Module comprises registers that exist at an offset relative to the DES Module base address and a small set of DES µDMA registers that exist at an offset relative to an Encryption Control Module base address. The DES Module register offsets are relative to the base address 0x4403.8000 . The Encryption Control register offsets are relative to the base address 0x4403.0000.

**Note:** The DES registers are limited to 32-bit data accesses; 8- and 16-bit accesses are not allowed and can corrupt register contents.

**Table 14-8. DES Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **DES Module Registers (DES Module Offset)** | | | | | |
| 0x000 | DES_KEY3_L | RW | 0x0000.0000 | DES Key 3 LSW for 192-Bit Key | 1051 |
| 0x004 | DES_KEY3_H | RW | 0x0000.0000 | DES Key 3 MSW for 192-Bit Key | 1051 |
| 0x008 | DES_KEY2_L | RW | 0x0000.0000 | DES Key 2 LSW for 128-Bit Key | 1051 |

**Table 14-8. DES Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x00C | DES_KEY2_H | RW | 0x0000.0000 | DES Key 2 MSW for 128-Bit Key | 1051 |
| 0x010 | DES_KEY1_L | RW | 0x0000.0000 | DES Key 1 LSW for 64-Bit Key | 1051 |
| 0x014 | DES_KEY1_H | RW | 0x0000.0000 | DES Key 1 MSW for 64-Bit Key | 1051 |
| 0x018 | DES_IV_L | RW | 0x0000.0000 | DES Initialization Vector | 1052 |
| 0x01C | DES_IV_H | RW | 0x0000.0000 | DES Initialization Vector | 1053 |
| 0x020 | DES_CTRL | RW | 0x8000.0000 | DES Control | 1054 |
| 0x024 | DES_LENGTH | RW | 0x0000.0000 | DES Cryptographic Data Length | 1055 |
| 0x028 | DES_DATA_L | RW | 0x0000.0000 | DES LSW Data RW | 1056 |
| 0x02C | DES_DATA_H | RW | 0x0000.0000 | DES MSW Data RW | 1057 |
| 0x030 | DES_REVISION | RO | 0x0000.0021 | DES Revision Number | 1058 |
| 0x034 | DES_SYSCONFIG | RW | 0x0000.0001 | DES System Configuration | 1059 |
| 0x038 | DES_SYSSTATUS | RO | 0x0000.0001 | DES System Status | 1061 |
| 0x03C | DES_IRQSTATUS | RO | 0x0000.0000 | DES Interrupt Status | 1062 |
| 0x040 | DES_IRQENABLE | RW | 0x0000.0000 | DES Interrupt Enable | 1063 |
| 0x044 | DES_DIRTYBITS | RW1C | 0x0000.0000 | DES Dirty Bits | 1064 |
| **DES µDMA Interrupt Registers (CRC and Cryptographic Modules (CCM) Offset)** | | | | | |
| 0x030 | DES_DMAIM | RW | 0x0000.0000 | DES DMA Interrupt Mask | 1065 |
| 0x034 | DES_DMARIS | RO | 0x0000.0000 | DES DMA Raw Interrupt Status | 1066 |
| 0x038 | DES_DMAMIS | RO | 0x0000.0000 | DES DMA Masked Interrupt Status | 1067 |
| 0x03C | DES_DMAIC | W1C | 0x0000.0000 | DES DMA Interrupt Clear | 1068 |

## 14.7 DES Register Description

This section lists and describes the DES registers, in numerical order by address offset.

### Register 1: DES Key 3 LSW for 192-Bit Key (DES_KEY3_L), offset 0x000

### Register 2: DES Key 3 MSW for 192-Bit Key (DES_KEY3_H), offset 0x004

### Register 3: DES Key 2 LSW for 128-Bit Key (DES_KEY2_L), offset 0x008

### Register 4: DES Key 2 MSW for 128-Bit Key (DES_KEY2_H), offset 0x00C

### Register 5: DES Key 1 LSW for 64-Bit Key (DES_KEY1_L), offset 0x010

### Register 6: DES Key 1 MSW for 64-Bit Key (DES_KEY1_H), offset 0x014

The function of each of these registers is described in more detail in Table 14-9 on page 1051.

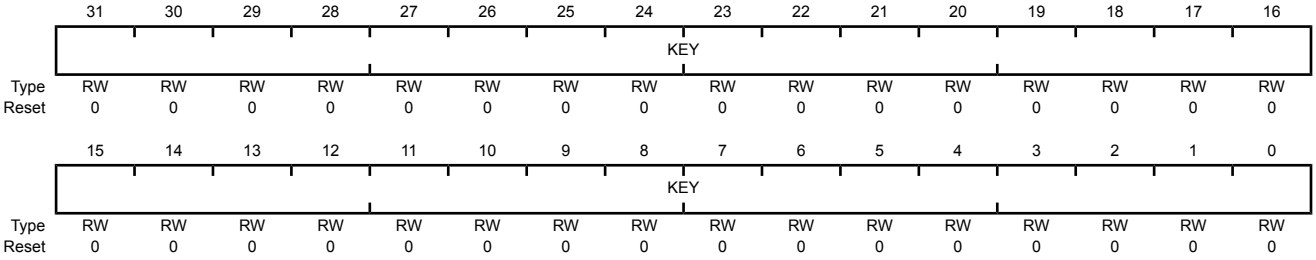
**Note:** A read of this register returns all zeros.

**Table 14-9. DES Key Register Mapping**

| Register Name | Address Offset | Description |
|---|---|---|
| DES_KEY3_L | 0x00000000 | KEY3 (LSW) for 192-bit key |
| DES_KEY3_H | 0x00000004 | KEY3 (MSW) for 192-bit key |
| DES_KEY2_L | 0x00000008 | KEY2 (LSW) for 192-bit key |
| DES_KEY2_H | 0x0000000C | KEY2 (MSW) for 192-bit key |
| DES_KEY1_L | 0x00000010 | KEY1 (LSW) for 64-bit key/192-bit key |
| DES_KEY1_H | 0x00000014 | KEY1 (MSW) for 64-bit key/192-bit key |

DES Key for 192-Bit Key (DES_KEYn_n)

Base 0x4403.8000
Offset 0x000
Type RW, reset 0x0000.0000



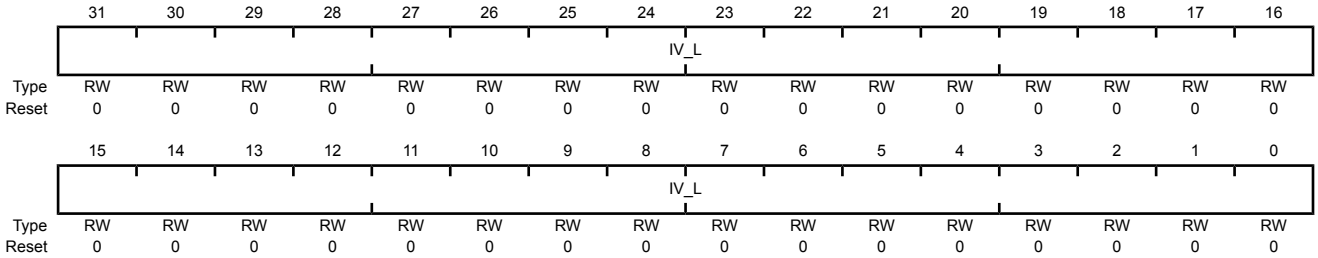

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | KEY | RW | 0x00000000 | Key Data |

## Register 7: DES Initialization Vector (DES_IV_L), offset 0x018

Least significant word of the initialization vector.

DES Initialization Vector (DES_IV_L)

Base 0x4403.8000
Offset 0x018
Type RW, reset 0x0000.0000



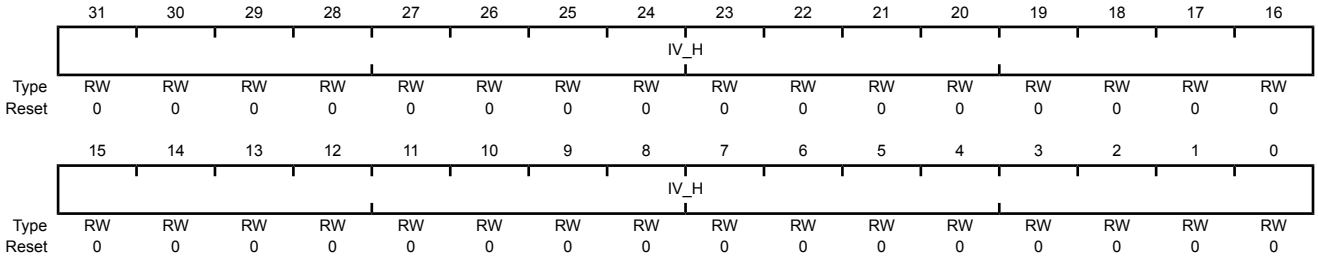

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | IV_L | RW | 0x00000000 | Initialization vector for CBC, CFB modes (LSW) |

## Register 8: DES Initialization Vector (DES_IV_H), offset 0x01C

Most significant word of the initialization vector.

DES Initialization Vector (DES_IV_H)

Base 0x4403.8000
Offset 0x01C
Type RW, reset 0x0000.0000

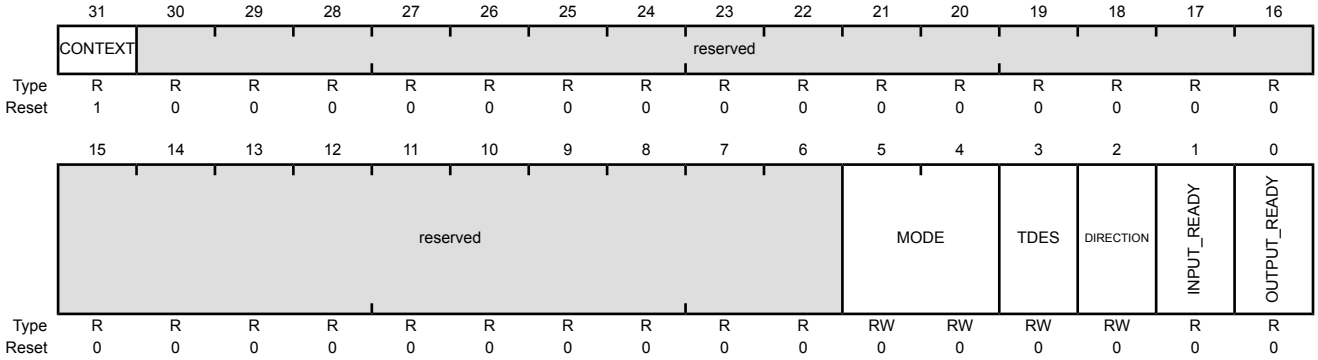



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | IV_H | RW | 0x00000000 | Initialization vector for CBC, CFB modes (MSW) |

## Register 9: DES Control (DES_CTRL), offset 0x020

DES Control (DES_CTRL)

Base 0x4403.8000
Offset 0x020
Type RW, reset 0x8000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CONTEXT | R | 1 | 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. |
| 30:6 | reserved | R | 0 | 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 | MODE | RW | 0 | Select CBC, ECB or CFB mode0x0: ECB mode0x1: CBC mode0x2: CFB mode0x3: reserved |
| 3 | TDES | RW | 0 | Select DES or triple DES encryption/decryption.0x0: DES mode0x1:TDESmode |
| 2 | DIRECTION | RW | 0 | Select encryption/decryption 0x0: decryption is selected0x1: Encryption is selected |
| 1 | INPUT_READY | R | 0 | When 1, ready to encrypt/decrypt data |
| 0 | OUTPUT_READY | R | 0 | When 1, Data decrypted/encrypted ready |

## Register 10: DES Cryptographic Data Length (DES_LENGTH), offset 0x024

Indicates the cryptographic data length in bytes for all modes. Once processing is started with this context, this length decrements to zero. Data lengths up to (232 - 1) bytes are allowed. A write to this register triggers the engine to start using this context.

**Note:** A read of this register returns all zeros.

DES Cryptographic Data Length (DES_LENGTH)

Base 0x4403.8000
Offset 0x024
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LENGTH | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LENGTH | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

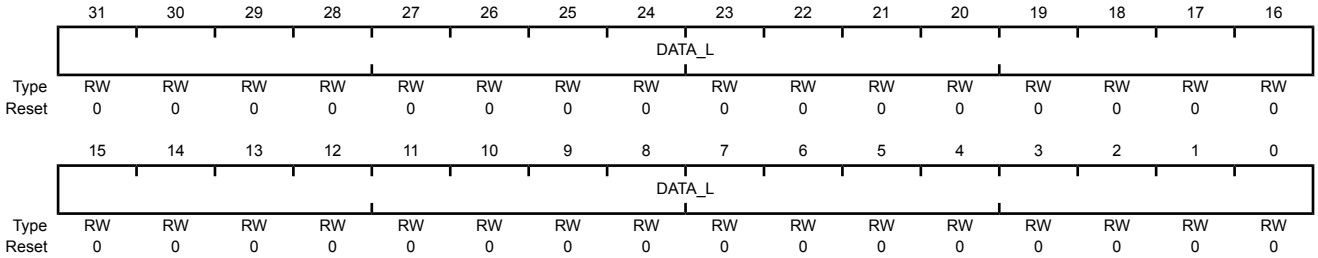
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LENGTH | RW | 0x00000000 | Cryptographic data length in bytes for all modes |

## Register 11: DES LSW Data RW (DES_DATA_L), offset 0x028

Data register (LSW) to read/write encrypted/decrypted data.

DES LSW Data RW (DES_DATA_L)

Base 0x4403.8000
Offset 0x028
Type RW, reset 0x0000.0000



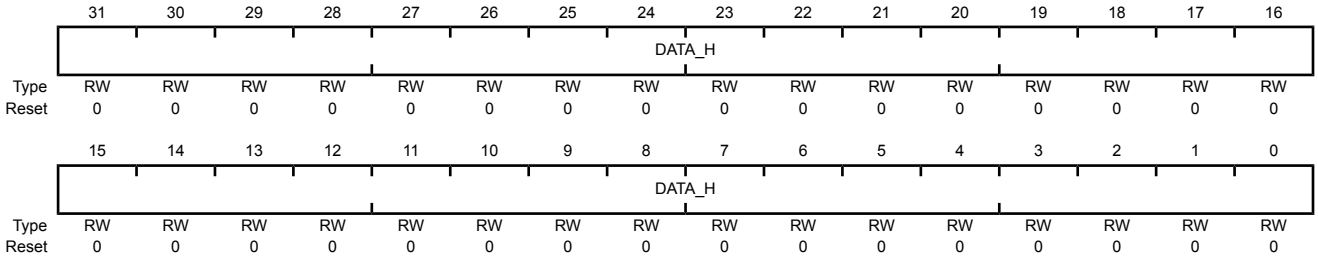

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA_L | RW | 0x00000000 | Data for encryption/decryption, LSW |

## Register 12: DES MSW Data RW (DES_DATA_H), offset 0x02C

Data register (MSW) to read/write encrypted/decrypted data.

DES MSW Data RW (DES_DATA_H)

Base 0x4403.8000
Offset 0x02C
Type RW, reset 0x0000.0000



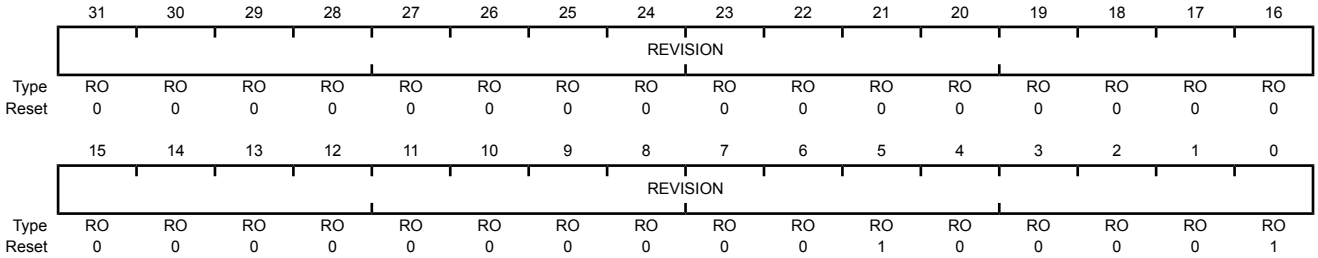

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA_H | RW | 0x00000000 | Data for encryption/decryption, MSW |

## Register 13: DES Revision Number (DES_REVISION), offset 0x030

Revision Number

DES Revision Number (DES_REVISION)

Base 0x4403.8000
Offset 0x030
Type RO, reset 0x0000.0021




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | REVISION | RO | 0x0000.0021 | Revision number |

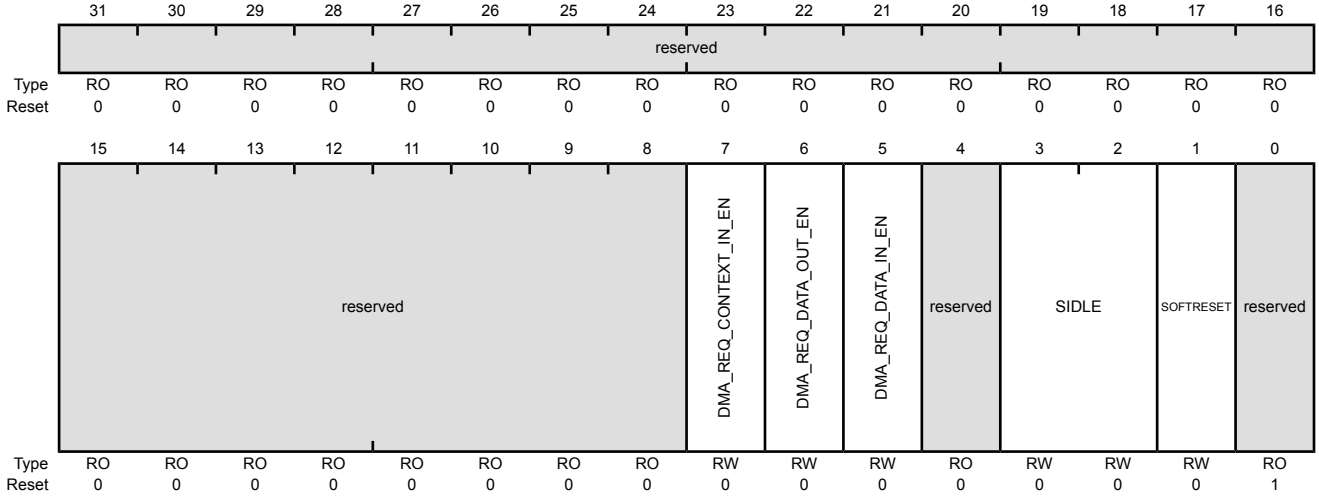
## Register 14: DES System Configuration (DES_SYSCONFIG), offset 0x034

System Configuration of DES module

**Note:** After one operation has completed, the **DES_SYSCONFIG** register must be cleared and re-configured for the next operation to ensure proper DMA and data operation functionality.

DES System Configuration (DES_SYSCONFIG)

Base 0x4403.8000
Offset 0x034
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | DMA_REQ_CONTEXT_IN_EN | RW | 0 | DMA Request Context In Enable<br>Value Description<br>0 DMA disabled<br>1 DMA enabled |
| 6 | DMA_REQ_DATA_OUT_EN | RW | 0 | DMA Request Data Out Enable<br>Value Description<br>0 DMA disabled<br>1 DMA enabled |
| 5 | DMA_REQ_DATA_IN_EN | RW | 0 | DMA Request Data In Enable<br>Value Description<br>0 DMA disabled<br>1 DMA enabled |

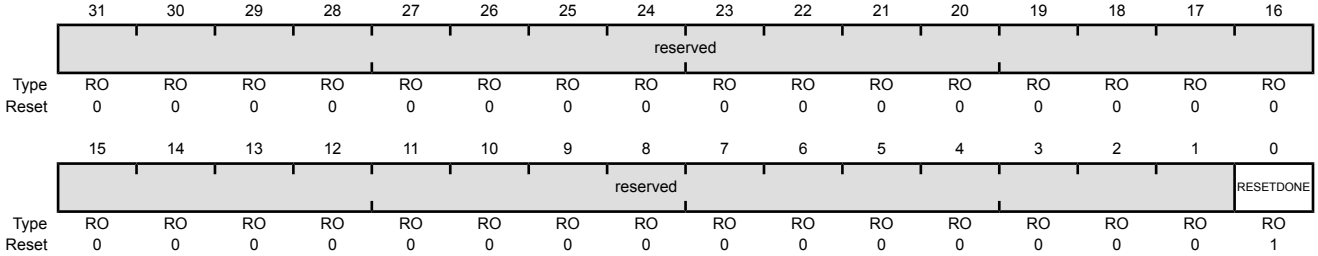
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3:2 | SIDLE | RW | 0x0 | Sidle mode<br><br>Value Description<br>0x0 Force-idle mode<br>0x1-0x3 reserved |
| 1 | SOFTRESET | RW | 0 | Soft reset<br><br>Value Description<br>0 No operation<br>1 Start soft reset sequence |
| 0 | reserved | RO | 1 | 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. |

## Register 15: DES System Status (DES_SYSSTATUS), offset 0x038

System Status Register

DES System Status (DES_SYSSTATUS)

Base 0x4403.8000
Offset 0x038
Type RO, reset 0x0000.0001



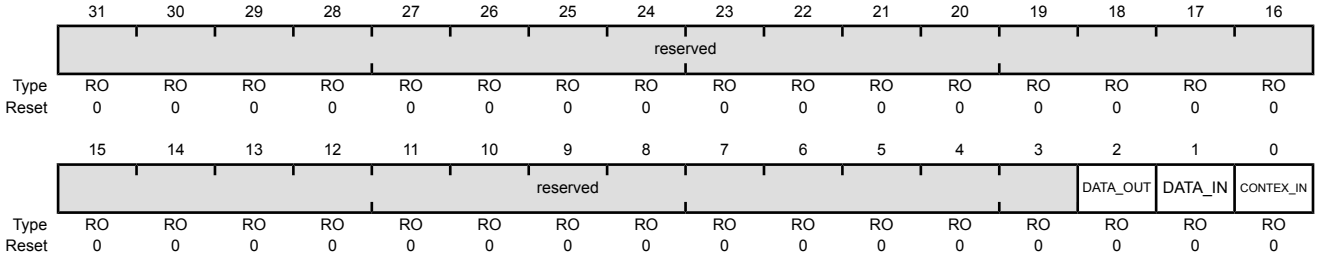

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | RESETDONE | RO | 1 | Reset Done<br>Value Description<br>0 Reset is not complete<br>1 Reset done |

## Register 16: DES Interrupt Status (DES_IRQSTATUS), offset 0x03C

This register indicates the interrupt status. If one of the interrupt bits is set the interrupt output will be asserted.

DES Interrupt Status (DES_IRQSTATUS)

Base 0x4403.8000
Offset 0x03C
Type RO, reset 0x0000.0000



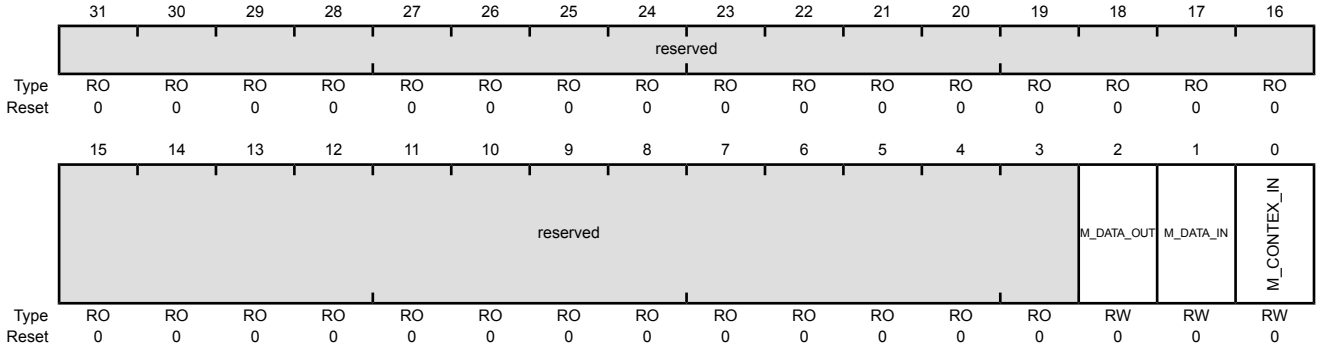

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DATA_OUT | RO | 0 | This bit indicates data output interrupt is active and triggers the interrupt output. |
| 1 | DATA_IN | RO | 0 | This bit indicates data input interrupt is active and triggers the interrupt output. |
| 0 | CONTEX_IN | RO | 0 | This bit indicates context interrupt is active and triggers the interrupt output. |

## Register 17: DES Interrupt Enable (DES_IRQENABLE), offset 0x040

This register contains an enable bit for each unique interrupt generated by the module. It matches the layout of **DES_IRQSTATUS** register. An interrupt is enabled when the bit in this register is set to 1.

DES Interrupt Enable (DES_IRQENABLE)

Base 0x4403.8000
Offset 0x040
Type RW, reset 0x0000.0000

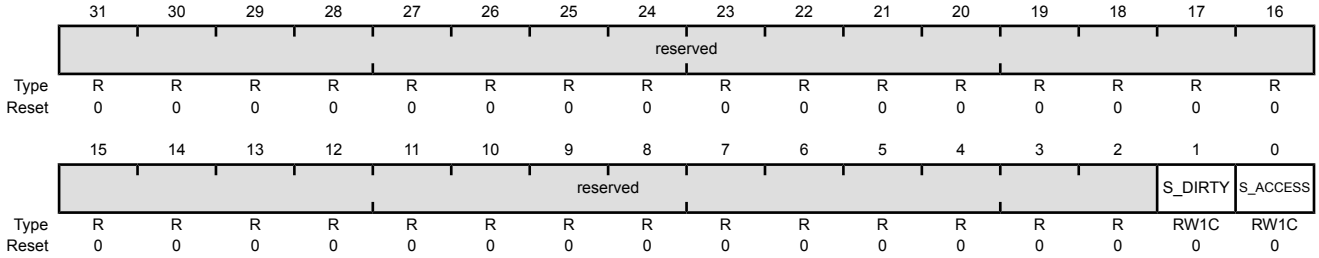



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | M_DATA_OUT | RW | 0 | If this bit is set to 1 the data output interrupt is enabled. |
| 1 | M_DATA_IN | RW | 0 | If this bit is set to 1 the data input interrupt is enabled. |
| 0 | M_CONTEX_IN | RW | 0 | If this bit is set to 1 the context interrupt is enabled. |

### Register 18: DES Dirty Bits (DES_DIRTYBITS), offset 0x044

DES Dirty Bits (DES_DIRTYBITS)

Base 0x4403.8000
Offset 0x044
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | R | 0 | 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. |
| 1 | S_DIRTY | RW1C | 0 | This bit is set to 1 by the module if any of the DES_* registers is written. Except DES_DIRTYBITS and DES_LOCKDOWN. |
| 0 | S_ACCESS | RW1C | 0 | This bit is set to 1 by the module if any of the DES_* registers is read. Except DES_DIRTYBITS and DES_LOCKDOWN. |

## 14.8 DES µDMA Interrupt Register Descriptions (CCM Offset)

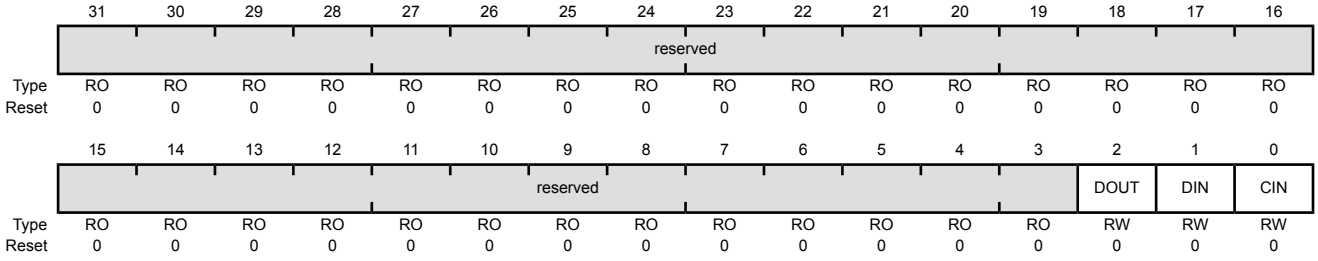
This section lists and describes the DES µDMA registers, in numerical order by address offset. Registers in this section are relative to the base address of 0x4403.0000.

## Register 19: DES DMA Interrupt Mask (DES_DMAIM), offset 0x030

The **DES DMA Interrupt Mask** register control interrupt behavior and are used to program which interrupts are suppressed.

DES DMA Interrupt Mask (DES_DMAIM)

Base 0x4403.0000
Offset 0x030
Type RW, reset 0x0000.0000



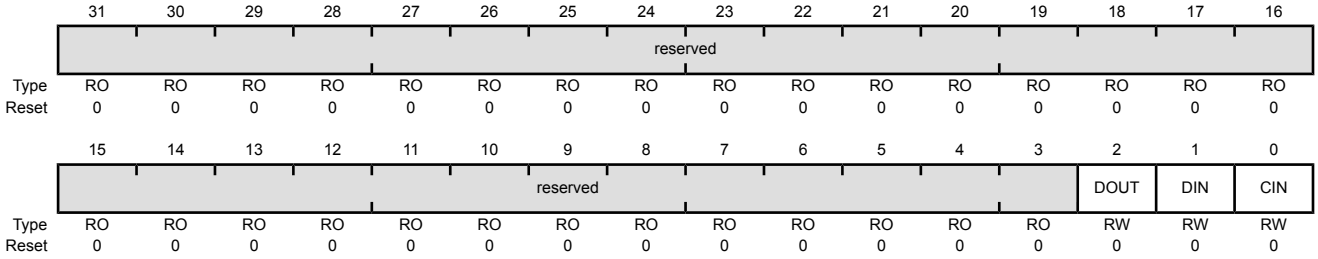

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DOUT | RW | 0 | Data Out DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA writes the last word of the process result.<br><br>Value Description<br>0 The DOUT interrupt is suppressed and not sent to the interrupt controller.<br>1 The DOUT interrupt is sent to the interrupt controller. |
| 1 | DIN | RW | 0 | Data In DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA writes the last word of input data to the internal FIFO of the engine.<br><br>Value Description<br>0 The DIN interrupt is suppressed and not sent to the interrupt controller.<br>1 The DIN interrupt is sent to the interrupt controller. |
| 0 | CIN | RW | 0 | Context In DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA completes a context write to the internal register.<br><br>Value Description<br>0 The CIN interrupt is suppressed and not sent to the interrupt controller.<br>1 The CIN interrupt is sent to the interrupt controller. |

## Register 20: DES DMA Raw Interrupt Status (DES_DMARIS), offset 0x034

The **DES DMA Raw Interrupt Status** register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set to 1.

DES DMA Raw Interrupt Status (DES_DMARIS)

Base 0x4403.0000
Offset 0x034
Type RO, reset 0x0000.0000



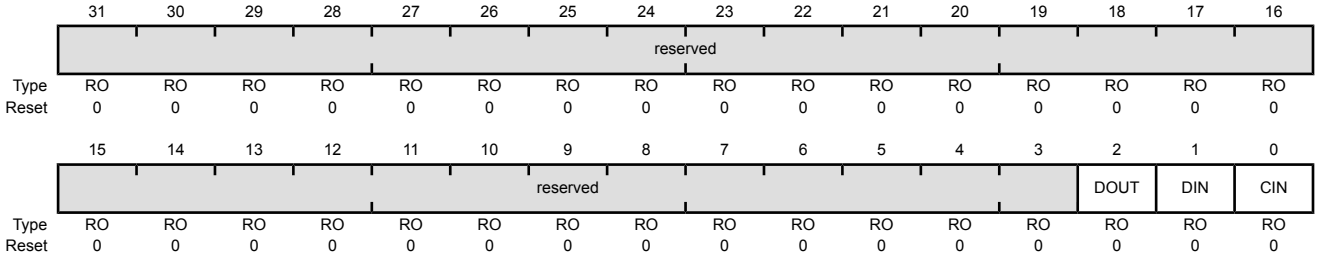

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DOUT | RW | 0 | Data Out DMA Done Raw Interrupt Status<br><br>Value Description<br>0 No Interrupt.<br>1 The µDMA has written the last word of the process result and an interrupt has been triggered and is pending. |
| 1 | DIN | RW | 0 | Data In DMA Done Raw Interrupt Status<br><br>Value Description<br>0 No Interrupt.<br>1 The µDMA has written the last word of input data to the internal FIFO of the engine and an interrupt has been triggered and is pending. |
| 0 | CIN | RW | 0 | Context In DMA Done Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The µDMA has completed a context write to the internal register and an interrupt has been triggered and is pending. |

## Register 21: DES DMA Masked Interrupt Status (DES_DMAMIS), offset 0x038

The **DES DMA Masked Interrupt Status** register displays the raw interrupts that are unmasked in the **DES DMA Raw Interrupt Status (DES_DMARIS)** register.

DES DMA Masked Interrupt Status (DES_DMAMIS)

Base 0x4403.0000
Offset 0x038
Type RO, reset 0x0000.0000



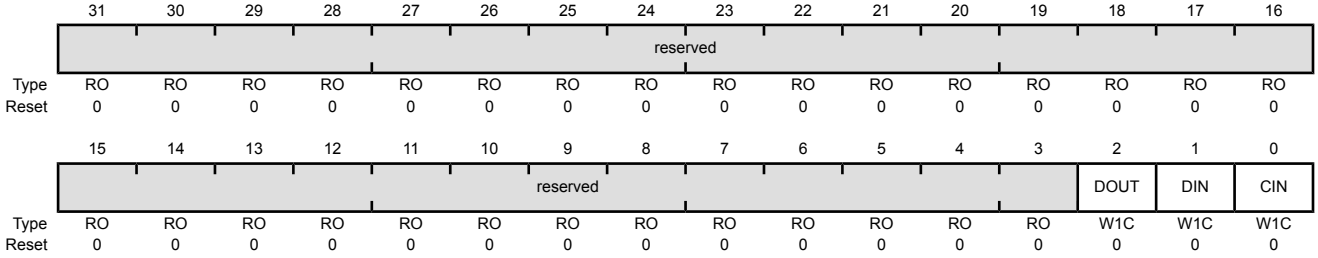

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DOUT | RO | 0 | Data Out DMA Done Masked Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A DOUT interrupt has occurred. |
| 1 | DIN | RO | 0 | Data In DMA Done Masked Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A DIN interrupt has occurred. |
| 0 | CIN | RO | 0 | Context In DMA Done Raw Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A CIN interrupt has occurred. |

## Register 22: DES DMA Interrupt Clear (DES_DMAIC), offset 0x03C

The **DES DMA Interrupt Clear** register is used to clear the **DES_DMARIS** and **DES_DMAMIS** registers by writing a 1 to the corresponding register bit.

**Note:** This registers always reads as zero.

DES DMA Interrupt Clear (DES_DMAIC)
Base 0x4403.0000
Offset 0x03C
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | 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. |
| 2 | DOUT | W1C | 0 | Data Out DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the `DOUT` bit in the **DES_DMARIS** and **DES_DMAMIS** register. |
| 1 | DIN | W1C | 0 | Data In DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the `DIN` bit in the **DES_DMARIS** and **DES_DMAMIS** register. |
| 0 | CIN | W1C | 0 | Context In DMA Done Raw Interrupt Status<br>Writing a 1 to this bit clears the `CIN` bit in the **DES_DMARIS** and **DES_DMAMIS** register. |

# 15 SHA/MD5 Accelerator

The SHA/MD5 module provides hardware-accelerated hash functions and can run:

- MD5 message digest algorithm developed by Ron Rivest in 1991
- SHA-1 algorithm compliant with the FIPS 180-3 standard
- SHA-2 (SHA-224 and SHA-256) algorithm compliant with the FIPS 180-3 standard
- Hash message authentication code (HMAC) operation

The algorithms produce a condensed representation of a message or a data file, called digest or signature, which can then be used to verify the message integrity.

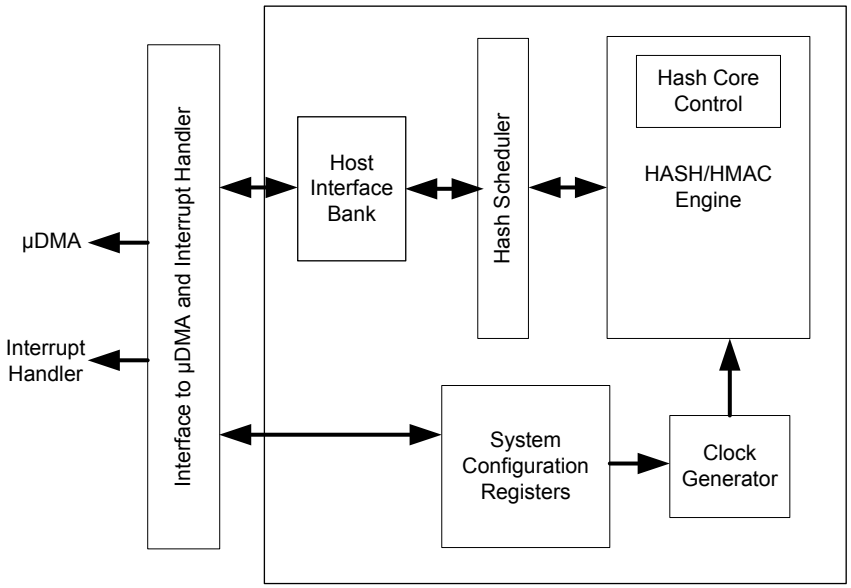
- Hashing of 0 to $2^{33}$ - 2 bytes of data (of which $2^{32}$ - 1 bytes are in one pass) using the MD5, SHA-1, SHA-224, or SHA-256 hash algorithm (byte granularity only, no support for bit granularity)
- Automatic HMAC key preprocessing for HMAC keys up to 64 bytes
- Host-assisted HMAC key preprocessing for HMAC keys larger than 64 bytes
- HMAC from precomputes (inner/outer digest) for improved performance on small blocks
- Supports µDMA operation for data and context in/result out transfers
- Supports interrupt to read the digest (signature)

## 15.1 SHA/MD5 Functional Description

### 15.1.1 SHA/MD5 Block Diagram

Figure 15-1 on page 1070 shows the module architecture, which consists of four primary blocks: the Hash/HMAC engine, the configuration registers and the interface to µDMA and the interrupt handler.

**Figure 15-1. SHA/MD5 Module Block Diagram**




### 15.1.1.1 Configuration Registers

The configuration registers contain the following global control and status registers for the SHA/MD5 Module:

- System control register that controls the mode of operation (**SHA_SYSCONFIG** register)
- µDMA interrupt control registers (**SHA_DMAIM**, **SHA_DMARIS**, **SHA_DMAMIS**, and **SHA_DMAIC** registers, which reside in the Encryption Control Base address space)
- Interrupt status register (**SHA_IRQSTATUS** register)
- Enable register (**SHA_IRQENABLE** register)

### 15.1.1.2 Hash/HMAC Engine

The Hash/HMAC engine performs the SHA-1, SHA-2, or MD5 hash computation. When loaded with a data block, and optionally an intermediate digest, it independently performs the hash computation (64 or 80 rounds, depending on the algorithm) on that data block.

It can also start from the specified initial digest values instead of a loaded intermediate. Furthermore, it can perform the IPAD and OPAD XORs for MAC operations. The hash core does not perform any hash padding; this is performed in the Host Interface Block, where the data input registers are located. A loaded data block must always be a full 64 bytes (512 bits) long.

### 15.1.1.3 Hash Core Control

When the hash core is idle or done, a new hash operation can be started. Any additional information needed by the hash core (mode of operation, data to process, input digest if not starting from algorithm constants or continuing) must be provided by programming the SHA registers before the core can accept the operation.

#### 15.1.1.4 Host Interface Bank

The Host Interface Bank can access the hash core for hashing individual 64-byte hash blocks. The Host Interface Bank contains registers such as the data FIFO (**SHA Data n Input (SHA_DATA_n_IN)** registers), the **SHA Inner Digest x (SHA_IDIGEST_X)** registers, and several control and status registers.

The Host Interface Block contains all relevant control logic for performing hash and HMAC computations on large (that is, larger than one hash block) blocks of data, including hash padding, final hash, and outer hash. It provides the necessary flow control to the SHA µDMA and interrupt interface.

### 15.1.2 Power Management

To save power, the application can disable the clock to the SHA/MD5 module when not in use. The SHA/MD5 is clock gated by setting the SHACFG bit in the **Cryptographic Modules Clock Gating Request (CCMCGREQ)** register, CCM offset 0x204. The SHA in addition to the AES, DES, and Enhanced CRC can also be clock gated as a group by setting the D0 bit in the **CRC and Cryptographic Modules Deep-Sleep Mode Clock Gating Control (DCGCCCM)** register, System Control Module offset 0x874.

### 15.1.3 Reset Management

To perform a software reset of the SHA module, write a 1 to the SOFTRESET bit in the **SHA System Configuration (SHA_SYSCONFIG)** register. The RESETDONE bit in the **SHA System Status (SHA_SYSSTATUS)** register indicates that the software reset is complete when its value is 1. When the software reset completes, the SOFTRESET bit in the **SHA_SYSCONFIG** register is automatically reset. Software must ensure that the software reset completes before doing any operations.

The behavior of the software reset is the same as the hardware reset, except that the software reset bit resets this module without affecting the reset core domain of the entire device.

### 15.1.4 µDMA and Interrupt Requests

The SHA/MD5 module can operate in µDMA mode where the module can assert a µDMA request for context in, context out, or data input. The µDMA signals that can be generated are:

- Context In µDMA request (SHA/MD5 0 Cin): Request for Key, Digest, Mode and LENGTH information
- Context Out µDMA request (SHA/MD5 0 Cout): Request for read from HMAC
- Data In µDMA request (SHA/MD5 0 Din): Request input data in multiples of 16 bytes

The SHA/MD5 Module be programmed to assert an interrupt when the µDMA has completed its last transfer by programming the **SHA DMA Interrupt Mask (SHA_DMAIM)**, at the CRC and Cryptographic Modules (CCM) offset 0x010. The **SHA DMA Raw Interrupt Status (SHA_DMARIS)** register, at CCM offset 0x014, indicates when the µDMA has completed and can be cleared by the **SHA DMA Interrupt Clear (SHA_DMAIC)** register at CCM offset 0x01C.

**Note:** The SHA module can only be accessed through privileged mode. If the µDMA is used for SHA transfers, then the µDMA's **DMA Channel Control (DMACHCTL)** register also needs to be programmed to allow for privileged accesses.

If context and data transfers are to be handled through software in Interrupt Mode, then the **SHA Interrupt Enable (SHA_IRQENABLE)**, offset 0x11C, can be used to enable interrupt triggering

when context out, context in, data in or data out is ready. The **SHA Interrupt Status (SHA_IRQSTATUS)**, offset 0x118, indicates when an interrupt is triggered.

**Note:** If the application uses Interrupt Mode, an interrupt is generated for each block of processed data. To support larger data flow, SHA µDMA Mode should be used and the bits in the **SHA_IRQENABLE** register should be cleared.

**Table 15-1. Interrupts and Events**

| Event | Description |
|---|---|
| **SHA_IRQSTATUS**[3]: CONTEXT_OUT | Context output interrupt |
| **SHA_IRQSTATUS**[1]: DATA_IN | Data input interrupt |
| **SHA_IRQSTATUS**[0]: CONTEXT_IN | Context input interrupt |

## 15.1.5 Operation Description

The SHA/MD5 Module can run the SHA-1, SHA-224, SHA-256, and MD5 algorithms, depending on the value of the ALGO bit field in the **SHA Mode (SHA_MODE)** register, offset 0x044, as listed in Table 15-2 on page 1072.

**Table 15-2. SHA/MD5 Module Algorithm Selection**

| ALGO Field Value in SHA_MODE Register | Description |
|---|---|
| 0x0 | MD5 algorithm selected |
| 0x1 | SHA-1 algorithm selected |
| 0x2 | SHA-224 algorithm selected |
| 0x3 | SHA-256 algorithm selected |

### 15.1.5.1 SHA Mode

***Starting a New Hash***

To start a new hash, follow these steps:

1. Set the ALGO bitfield in the **SHA_MODE** register, at offset 0x044, to 0x1, 0x2, or 0x3 to select SHA-1, SHA-224, or SHA-256, respectively.

2. Set the ALGO_CONSTANT bit in the **SHA_MODE** register to 1 to initialize all **SHA Inner/Outer Digest n** registers from **SHA_ODIGEST_A/SHA_IDIGEST_A** to **SHA_ODIGEST_H/SHA_IDIGEST_H** with their default values specified by the algorithm, and set the **SHA_DIGESTCOUNT** register to 0.

3. Set the CLOSE_HASH bit of the **SHA Mode (SHA_MODE)** register to let the SHA engine do the padding. If the Hash is computed in one shot, the length of the message can be any value up to 128 MB. To process an intermediate Hash digest, the CLOSE_HASH bit is set to 0, in which case the packets hashed must be 64 bytes; the last packet must be hashed with the CLOSE_HASH bit set to 1.

4. Specify the LENGTH field in the **SHA Length (SHA_LENGTH)** register of the hash data to process in bytes.

After the configuration is complete, the INPUT_READY status bit equals 1 in the **SHA Interrupt Status (SHA_IRQSTATUS)** register (regardless of whether or not the M_INPUT_READY bit in the **SHA_IRQENABLE** register is set. When this bit is set, it indicates the SHA engine can receive the

data to process. Data must be written to the 16 x 32-bit **SHA_DATA_n_IN** registers that provide storage for one 64-byte block of data. Unless the CLOSE_HASH bit is set, all of the **SHA_DATA_n_IN** input buffers must be filled. Data can be written by single write accesses to the 16 registers from a processor or by a DMA transfer.

For µDMA transfers, the DMA_EN bit must be set in the **SHA_SYSCONFIG** register and the appropriate mask bits must be set in the **SHA_DMAIM** register before starting the new hash. Note that if the µDMA is used for transfers, the **SHA_IRQENABLE** register should be clear so all interrupts are generated through the µDMA interrupt registers.

The µDMA must be configured to transfer 16 data words of 32 bits each time it is triggered by a µDMA request from the SHA/MD5 Module. The 16 data words written are sent to the 16 **SHA_DATA_n_IN** registers.

The module detects that a 64-byte block is available, and then moves the data to a working register space for processing and asserts the INPUT_READY bit in the **SHA_IRQSTATUS** register to 1. If the DMA_EN bit in the **SHA_SYSCONFIG** register has been set to 1, a new µDMA request triggers a new block transfer; otherwise, the processor polls the INPUT_READY bit and writes the 16 data words of 32 bits when it equals 1.

This operation is repeated until the length of the message to hash is reached. The OUPUT_READY bit in the **SHA_IRQSTATUS** register then indicates that the hash operation is complete. If the IT_EN bit in the **SHA_SYSCONFIG** register is set, an interrupt (active low) is also generated to indicate the hash completion.

The processor can then read the eight digest registers A through H that contain the hash and/or HMAC result. If the hash is an intermediate result of a larger hash, the digest count register must also be read and saved.

**Note:** The number of digest registers used depends on the algorithm selected for the SHA/MD5 Module (MD5, SHA-1, SHA-224, or SHA-256), as shown in Table 15-3 on page 1073 and Table 15-4 on page 1074

**Table 15-3. Outer Digest Registers**

| Register | Address | MD5 (Read/Write) | SHA-1 (Read/Write) | SHA-2 (Read/Write) | HMAC Key Processing (write) |
|---|---|---|---|---|---|
| SHA_ODIGEST_A | 0x000 | Outer digest [127:96] | Outer digest [159:128] | Outer digest [255:224] | HMAC Key [31:0] |
| SHA_ODIGEST_B | 0x004 | Outer digest [95:64] | Outer digest [127:96] | Outer digest [223:192] | HMAC key [63:32] |
| SHA_ODIGEST_C | 0x008 | Outer digest [63:32] | Outer digest [95:64] | Outer digest [191:160] | HMAC key [95:64] |
| SHA_ODIGEST_D | 0x00C | Outer digest [31:0] | Outer digest [63:32] | Outer digest [159:128] | HMAC key [127:96] |
| SHA_ODIGEST_E | 0x010 | | Outer digest [31:0] | Outer digest [127:96] | HMAC key [159:128] |
| SHA_ODIGEST_F | 0x014 | | | Outer digest [95:64] | HMAC key [191:160] |
| SHA_ODIGEST_G | 0x018 | | | Outer digest [63:32] | HMAC key [223:192] |
| SHA_ODIGEST_H | 0x01C | | | Outer digest [31:0] | HMAC key [255:224] |

**Table 15-4. Inner Digest Registers**

| Register | Address | MD5 (Read/Write) | SHA-1 (Read/Write) | SHA-2 (Read/Write) | SHA-256 (Read/Write) | HMAC Key Processing (write) |
|---|---|---|---|---|---|---|
| SHA_IDIGEST_A | 0x020 | Inner digest [127:96] | Inner digest [159:128] | Inner digest [223:192] | Inner digest [255:224] | HMAC key [287:256] |
| SHA_IDIGEST_B | 0x024 | Inner digest [95:64] | Inner digest [127:96] | Inner digest [191:160] | Inner digest [223:192] | HMAC key [319:288] |
| SHA_IDIGEST_C | 0x028 | Inner digest [63:32] | Inner digest [95:64] | Inner digest [159:128] | Inner digest [191:160] | HMAC key [351:320] |
| SHA_IDIGEST_D | 0x02C | Inner digest [31:0] | Inner digest [63:32] | Inner digest [127:96] | Inner digest [159:128] | HMAC key [383:352] |
| SHA_IDIGEST_E | 0x030 | | Inner digest[31:0] | Inner digest [95:64] | Inner digest [127:96] | HMAC key [415:384] |
| SHA_IDIGEST_F | 0x034 | | | Inner digest [63:32] | Inner digest [95:64] | HMAC key [447:416] |
| SHA_IDIGEST_G | 0x038 | | | Inner digest [31:0] | Inner digest [63:32] | HMAC key [479:448] |
| SHA_IDIGEST_H | 0x03C | | | | Inner digest[31:0] | HMAC key [511:480] |

**Note:** Inner digests are initial, intermediate, and result digests.

### *Outer Digest Registers*

The **SHA_ODIGEST_A** to **SHA_ODIGEST_H** registers are relevant only for HMAC operations; the contents are ignored for hash operations.

Before writing to the digest registers, the operation must be configured in the **SHA Mode (SHA_MODE)** register. For HMAC operations without key processing, the HMAC_KEY_PROC bit must be clear in the **SHA_MODE** register before starting operations. Once the algorithm has been programmed in the **SHA_MODE** register, only the relevant digest registers for the selected algorithm must be written:

- **SHA_ODIGEST_A** to **SHA_ODIGEST_D** registers for MD5
- **SHA_ODIGEST_A** to **SHA_ODIGEST_E** registers for SHA-1
- **SHA_ODIGEST_A** to **SHA_ODIGEST_H** registers for SHA-2 (224 to 256)

When HMAC key processing is enabled (HMAC_KEY_PROC=1), these registers must be written with the lower 256 bits of the HMAC key to be processed in little-endian format (first byte of key string in bits [7:0]).

**Note:** If the HMAC key is less than 512 bits, it must be properly padded with zeros: all 16 HMAC key registers must be written explicitly; the core does not pad. Additionally, if the HMAC key is larger than 512 bits, the host must perform a preprocessing step to reduce it to one 512-bit block. This involves hashing the large key and padding the hash result with zeros until it is 512 bits wide.

The computed outer digest can be read from these registers when the **SHA Interrupt Status (SHA_IRQSTATUS)** register when the OUTPUT_READY bit has been set indicating that the operation is done.

**Note:** If no HMAC key processing is performed, the value read is identical to the value written initially. The MD5 outer digest is available from registers **SHA_ODIGEST_A** to

**SHA_ODIGEST_D**, the SHA-1 outer digest from registers **SHA_ODIGEST_A** to **SHA_ODIGEST_E**, and the SHA-224 and SHA-256 outer digest from registers**SHA_ODIGEST_A** to **SHA_ODIGEST_H**.

**Note:** The HMAC key is not preserved. If another block must be authenticated using the same key, the key must be reloaded by the host. If the same key must be used many times, it is advisable to do a HMAC key processing-only pass to obtain the inner and outer digest precomputes and load these precomputes for subsequent passes (only the inner digest must be reloaded if the outer digest is not modified by the host), because this saves two hash blocks worth of computation time.

### *Inner Digest Registers*

The **SHA_IDIGEST_A** to **SHA_IDIGEST_H** registers are used for HMAC and hash operations.

The inner/initial digest for HMAC and hash continue operations (HMAC_KEY_PROC = 0 and ALGO_CONSTANT = 0) must be written to these registers before starting the operation by writing to the **SHA_MODE** register. Only the relevant digest registers for the selected algorithm must be written:

- **SHA_IDIGEST_A** to **SHA_IDIGEST_D** registers for MD5
- **SHA_IDIGEST_A** to **SHA_IDIGEST_E** registers for SHA-1
- **SHA_IDIGEST_A to SHA_IDIGEST_H** registers for SHA-2

When ALGO_CONSTANT = 1 in the **SHA_MODE** register, the **SHA Inner Digest n (SHA_IDIGEST_n)** registers do not need to be written by the application because they are overwritten with the appropriate algorithm constants.

When HMAC_KEY_PROC is 1, these registers must be written with the upper 256 bits of the HMAC key to be processed in little-endian format (first byte of key string in bits [7:0]).

**Note:** If the HMAC key is less than 512 bits, it must be properly padded with zeros: all 16 HMAC key registers must be written explicitly; the core does not pad. Additionally, if the HMAC key is larger than 512 bits, the host must perform a preprocessing step to reduce it to one 512-bit block. This involves hashing the large key and padding the hash result with zeros until it is 512 bits wide.

The order of the bytes within the digest is such that it can be fed back unmodified into the little-endian data input when preprocessing HMAC keys larger than 64 bytes, or it can typically be inserted unmodified into a little-endian data stream (for example, IPSEC packets), regardless of the selected algorithm.

**Note:** The HMAC key or inner digest is not preserved. If another block must be authenticated using the same key, the key or inner digest must be reloaded by the host. If the same key must be used many times, it is advisable to do a HMAC key processing-only pass to obtain the inner and outer digest precomputes and load these precomputes for subsequent passes (only the inner digest must be reloaded if the outer digest is not modified by the host), because this saves two hash blocks worth of computation time.

### *Closing a Hash*

The amount of data to hash is not necessarily a multiple of 64 bytes. The CLOSE_HASH bit in the **SHA_MODE** register is set to append padding so that the message size becomes a multiple of 64 bytes. Consequently, a minimum of 9 bytes must be added to the message. Nine bytes is the minimum number of bytes that contains the minimum 65-bit padding specified by FIPS 180-1.

If the size of the last block of data is less than or equal to 55 bytes, no additional 64-byte block is required. However, if the last block of data contains more than 55 bytes, an extra 64-byte block must be added to make the padding as specified by FIPS 180-1. This extra block is added automatically by the hardware; thus, the module is fed with a 64-byte block of data. However, appending a pad on the last block of data can result in the creation of an extra 64-byte block.

The one or two last blocks that contain the padding are processed in the same way as the other blocks. Hash completion is then indicated in the same way as for a new hash, and the hash result can be read in the digest registers. The **SHA_DIGESTCOUNT** register returns restored Digest Count + Length when it is read, and hashing completes.

Assuming a message of 129 bytes, Table 15-5 on page 1076 shows the SHA digest for three passes. Table 15-6 on page 1076 shows the SHA digest for one pass.

**Table 15-5. SHA Digest Processed in Three Passes**

| | Digest (A to E) | SHA_DIGESTCOUNT | SHA_MODE and SHA_LENGTH | SHA_DATA_n_IN |
|---|---|---|---|---|
| First pass | | | WRITE:<br>LENGTH=64<br>ALGO (dependent on the algorithm to apply)<br>ALGO_CONSTANT=1<br>CLOSE_HASH=0 | First 64 bytes of message |
| Second pass | Round 1 digest calculation | WRITE: 64 | WRITE:<br>LENGTH=64<br>ALGO (dependent on the algorithm to apply)<br>ALGO_CONSTANT=0<br>CLOSE_HASH=0 | Second 64 bytes of message |
| Third pass | Round 2 digest calculation | WRITE: 128 | Write:<br>LENGTH=1<br>ALGO (dependent on the algorithm to apply)<br>ALGO_CONSTANT=0<br>CLOSE_HASH=1 | Last byte of message |
| | Final digest | READ: 129 | | |

If the three passes are not performed in succession, the digest registers must be saved and restored for the next use of the SHA/MD5 engine. If the rounds are performed consecutively, there is no need to do anything with the digest registers.

**Table 15-6. SHA Digest Processed in One Pass**

| | Digest (A to E) | SHA_DIGEST_COUNT | SHA_MODE and SHA_LENGTH | SHA_DATA_n_IN |
|---|---|---|---|---|
| First pass | | | WRITE:<br>LENGTH=129<br>ALGO (dependent on the algorithm to apply)<br>ALGO_CONSTANT=1<br>CLOSE_HASH=1 | First 64 bytes of message |

**Table 15-6. SHA Digest Processed in One Pass *(continued)***

| | Digest (A to E) | SHA_DIGEST_COUNT | SHA_MODE and SHA_LENGTH | SHA_DATA_n_IN |
|---|---|---|---|---|
| | Round 1 digest calculation | | | Second 64 bytes of message |
| | Round 2 digest calculation | | | Last byte of message |
| | Final digest | Read: 129 | | |

### 15.1.5.2 MD5 Mode

***Starting a New Hash***

To start a new hash, perform the following steps:

1. Set the ALGO bit field in the **SHA_MODE** register to 0x0 to select the MD5 algorithm.
2. Set the ALGO_CONSTANT bit to 1 in the **SHA_MODE** register to initialize all digest registers from **SHA_ODIGEST_A/SHA_IDIGEST_A** to **SHA_ODIGEST_H/SHA_IDIGEST_H** with default values specified by the algorithm, and set the **SHA_DIGESTCOUNT** register to 0.
3. Specify the LENGTH field in the **SHA_LENGTH** register of the hash data to process in bytes.
4. Set the CLOSE_HASH bit in the **SHA_MODE** register to let the SHA/MD5 engine do the padding. If MD5 is computed in one shot, the length of the message can be any value up to . To process an intermediate MD5 digest, the CLOSE_HASH bit is set to 0, in which case packets to be hashed must be 64 bytes; the last packet must be hashed with the CLOSE_HASH bit set to 1.

After the configuration is complete, the hash engine can receive the data to process (the INPUT_READY bit is 1 in the **SHA_IRQSTATUS** register). Data must be written to the 16 x 32-bit **SHA_DATA_n_IN** registers that provide storage for one 64-byte block of data. Unless the CLOSE_HASH bit is set in the **SHA_MODE** register, the **SHA_DATA_n_IN** 64-byte input buffer must be filled. Data can be written by single write transactions to the 16 registers from a processor or by a µDMA transfer.

For a µDMA transfer, the SDAM_EN bit must be set in the **SHA_SYSCONFIG** register before starting the new hash and the µDMA channel for SHA/MD5 0 Data In Request must be configured. The µDMA must be configured to the appropriate hash transfer size. See “Micro Direct Memory Access (µDMA)” on page 708 for more information on programming the µDMA. A µDMA done is asserted after the last **SHA_DATA_n_IN** register is filled..

The module detects that a 64-byte block is available, and then moves the data to a working register space for processing and sets the INPUT_READY bit to 1 in the **SHA_IRQSTATUS** register. If the DMA_EN bit is set in the **SHA_SYSCONFIG** register, then a new µDMA request triggers a new block transfer; otherwise, the processor polls the INPUT_READY bit in the **SHA_IRQSTATUS** register and writes the 16 data words of 32 bits when it equals 1.

This operation repeats until the length of the message to hash is reached. The OUTPUT_READY bit in the **SHA_IRQSTATUS** register then indicates that the hash operation is complete. If the IT_EN bit in the **SHA_SYSCONFIG** register is set, an interrupt (active low) is also generated to indicate the hash completion.

#### *Closing a Hash*

The amount of data to hash is not necessarily a multiple of 64 bytes. In this case, the CLOSE_HASH bit in the SHA_MODE register must be set to append padding so that the message size becomes a multiple of 64 bytes. See the previous MD5 algorithm for more information on padding.

The module is fed with a 64-byte block of data, as long as enough data is available. However, a pad is appended on the last block of data. This can result in the creation of an extra 64-byte block.

The one or two last blocks that contain the padding are processed the same way as the other blocks. Hash completion is then indicated the same way as for a new hash, and the 128-bit result can be read in the digest registers. The **SHA_DIGESTCOUNT** register returns restored digest count and length when it is read, and hashing completes.

### 15.1.5.3 Generating an Software Interrupt

If the IT_EN bit is 1 in the **SHA_SYSCONFIG** register, an interrupt is generated at the completion of the hash by the following steps:

1. Receive last block of data (= 64 bytes). (The number of data bytes defined by the **SHA_LENGTH** register is received in the digest registers, from **SHA_ODIGEST_A/SHA_IDIGEST_A** to **SHA_ODIGEST_H/SHA_IDIGEST_H**.
2. If required, apply padding to the last block of data.
3. Hash the last block of data (80 cycles in SHA-1 mode and 64 cycles in MD5, SHA-224, and SHA-256 modes).
4. If required, add an extra 64-byte block of data to complete the padding.
5. Hash this extra block of data (80 cycles in SHA-1 mode and 64 cycles in MD5, SHA-224, and SHA-256 modes).
6. An interrupt is generated (active low).

## 15.1.6 SHA/MD5 Performance Information

The following table lists the performance for all supported key sizes and modes of operations. It assumed that the engine is kept fully utilized (that is, the host is supplying input block and retrieving output blocks in such a way, that the engine never has to wait for input) and that the previous output has been retrieved before the next output is ready.

Maximum throughput does not include per operation overhead cycles, as the impact on the throughput depends on the size of the block being processed, and would be negligible for large blocks anyway.

**Table 15-7. SHA/MD5 Performance**

| Operation | Algorithm | Cycles per operation | Cycles per block |
|---|---|---|---|
| Hash | MD5 | 0 / 65 | 65 |
| | SHA-1 | 0 / 81 | 81 |
| | SHA-224 | 0 / 65 | 65 |
| | SHA-256 | 0 / 65 | 65 |

**Table 15-7. SHA/MD5 Performance *(continued)***

| Operation | Algorithm | Cycles per operation | Cycles per block |
|---|---|---|---|
| HMAC from Key | MD5 | 196 / 261 | 65 |
| | SHA-1 | 244 / 325 | 81 |
| | SHA-224 | 196 / 261 | 65 |
| | SHA-256 | 196 / 261 | 65 |
| | MD5 | 66 / 131 | 65 |
| HMAC from precomputes | SHA-1 | 82 / 163 | 81 |
| | SHA-224 | 66 / 131 | 65 |
| | SHA-256 | 66 / 131 | 65 |

**Note:** An extra block needs to be processed if the length of the data block to be hashed module 64 is 0 or equal to 56.

## 15.1.7 SHA/MD5 Programming Guide

This section covers the hardware programming sequences for configuration and use of the SHA/MD5 Module.

### 15.1.7.1 Global Initialization

***Surrounding Modules Global Initialization***

This section identifies the requirements for initializing the surrounding modules when the SHAM is used for the first time after a device reset.

1. When reset has completed, enable the SHA/MD5 Module by setting the R0 bit in the **CRC And Cryptographic Modules Run Mode Clock Gating Control (RCGCCCM)** register, System Control offset 0x674. When the R0 bit is set in the **CRC and Cryptographic Modules (PRCCM)** register, System Control offset 0xA74 register, the SHA/MD5 Module is powered and ready to be configured.

2. Configure the SHA µDMA channels for Context In, Context Out, Data In, and/or Data Out by programming the appropriate encoding value in the **DMA Channel Map Select n (DMACHMAPn)** register in the µDMA module, offset 0x510. For more information on how to program channel assignments as well as enabling burst and the configured channels, refer to “Micro Direct Memory Access (µDMA)” on page 708.

3. Execute a software reset by setting the SOFTRESET bit in the **SHA_SYSCONFIG** register. When reset is complete, the RESETDONE bit reads as 1 in the **SHA_SYSSTATUS** register.

4. If the SHA channels are configured in the µDMA, enable the required SHA DMA requests by programming bits [9:5] of the **SHA_SYSCONFIG** register, in addition to the completion interrupts in the **SHA DMA Interrupt Mask (SHA_DMAIM)** register, CRC and Cryptographic Modules offset 0x020.

***Starting a New HMAC using the SHA-1 Hash Function and HMAC Key Processing***

The following procedure is used to begin a new HMAC operation, starting from initial digest values.

1. Load the key value in the **SHA_ODIGEST_A/SHA_IDIGEST_A** to **SHA_ODIGEST_H/SHA_IDIGEST_H** registers.

2. Pad the rest of the SHA _ODIGEST_x and SHA_IDIGEST registers with zeros.
3. Load the message in the **SHA_DATA_n_IN** FIFO registers.
4. Enable HMAC key processing by setting the HMAC_KEY_PROC bit in the **SHA_MODE** register.
5. Select the SHA-1 hash function by programming the ALGO bit in the **SHA_MODE** register to 0x1.
6. Select the already loaded key by programming the ALGO_CONSTANT bit in the SHA_MODE register to 0x0.
7. Set the CLOSE_HASH bit and the HMAC_OUTER_HASH bit in the **SHA_MODE** register so that appropriate padding is inserted and the outer hash is performed immediately after the inner hash has finished.
8. Program the **SHA_LENGTH** register with the block length. Writing this register triggers the HMAC engine to begin processing.

**Note:** If more than one pass is used during the process (SHA_MODE[4] CLOSE_HASH == 0x0), the block length value must be a 64-byte multiple. From this point, three operational modes are possible to continue with the processing: polling, interrupt, and DMA. For more information, see the Operational Modes Configuration section.

***Subsequence - Continuing a Prior HMAC Using the SHA-1 Hash Function***

The procedure in continues a prior HMAC calculation interrupted from a high priority task.

**Table 15-8. Continuing a Prior HMAC**

| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Load the initial digest for the used hash algorithm. | SHA_IDIGEST_i[31:0] DATA | - |
| Restore the digest counter with the value before the switch to the high-priority task. | SHA_DIGEST_COUNT[31:0] COUNT | - |
| Use the already loaded in the engine key. | SHA_MODE[5] HMAC_KEY_PROC | 0x0 |
| Do not use the constants of the selected hash algorithm. | SHA_MODE[3] ALGO_CONSTANT | 0x0 |
| Select the SHA-1 hash algorithm. | SHA_MODE[2:1] ALGO | 0x1 |
| IF: This is the last 64-byte data block from the input message? | User decision | |
| Close the hash; an appropriate padding is added. | SHA_MODE[4] CLOSE_HASH | 0x1 |
| ENDIF | | |
| Load the block length; this is the trigger to start processing. | SHA_LENGTH[31:0] LENGTH | - |

**Note:** This initial digest is the intermediate digest from the previous calculation before switching to the high priority task. The value is equal to context1 in .

**Note:** The value is equal to context2 in .

**Note:** The block length is equal to the context3 value in .

***Subsequence - Hashing a Key Bigger than 512 Bits with the SHA-1 Hash Function***

The procedure in creates a hash value from the key in only one pass.

**Table 15-9. SHA-1 Apply on the Key**

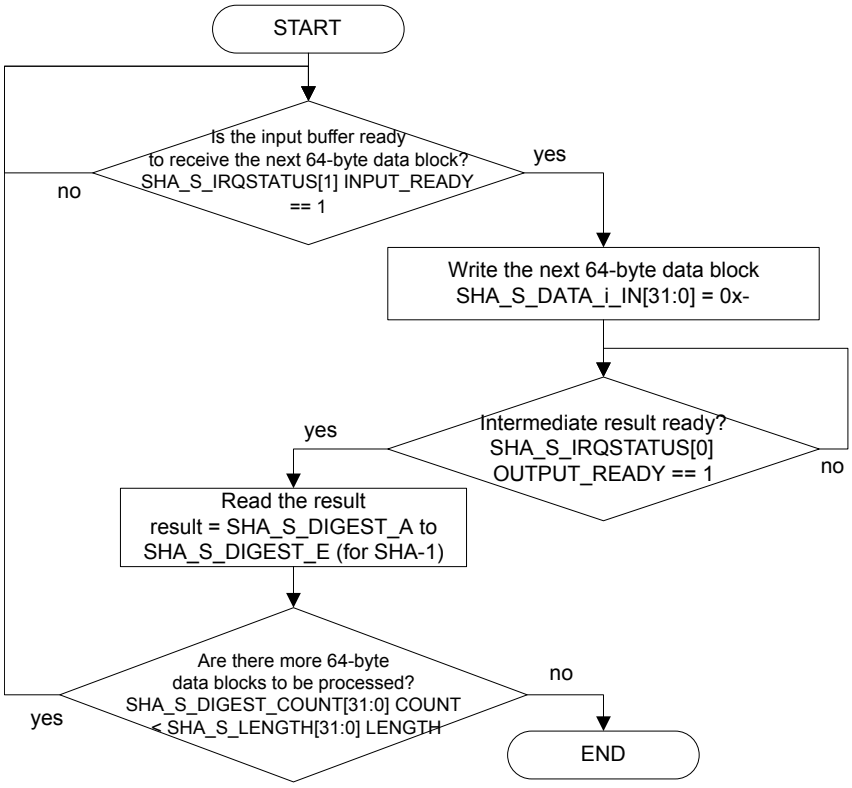
| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Load the first part of the key. (Here, the key is like a message.) | SHA_DATA_n_IN (i = 0 to 15) | - |
| Select the SHA-1 hash function. | SHA_MODE[2:1] ALGO | 0x1 |
| Select a new hash operation. | SHA_MODE[3] ALGO_CONSTANT | 0x1 |
| Close the hash; the key is processed in single pass. | SHA_MODE[4] CLOSE_HASH | 0x1 |

### *Operational Modes Configuration*

#### *SHA/MD5 Polling Mode*

Figure 15-2 on page 1081 shows the SHA/MD5 polling mode. SHA/MD5 polling mode uses the following registers: SHA_IRQSTATUS, SHA_DATA_n_IN, SHA_ODIGEST_A, SHA_DIGEST_COUNT, and SHA_LENGTH.

**Figure 15-2. SHA/MD5 Polling Mode**




#### *SHA/MD5 Interrupt Mode*

The procedure in configures the SHA/MD5 module to work in interrupt-based mode. (For the interrupt subroutine, see the Interrupt Servicing section.)

**Table 15-10. Interrupt Mode**

| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Enable the interrupt request to the Cortex-A8 MPU INTC. | SHA_SYSCONFIG[2] IT_EN | 0x1 |

**Table 15-10. Interrupt Mode *(continued)***

| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Load the message length; this is the trigger to start processing. | SHA_LENGTH[31:0] LENGTH | - |

### *SHA/MD5 DMA Mode*

The procedure in configures the SHA/MD5 module to work in DMA-based mode.

**Table 15-11. DMA Mode**

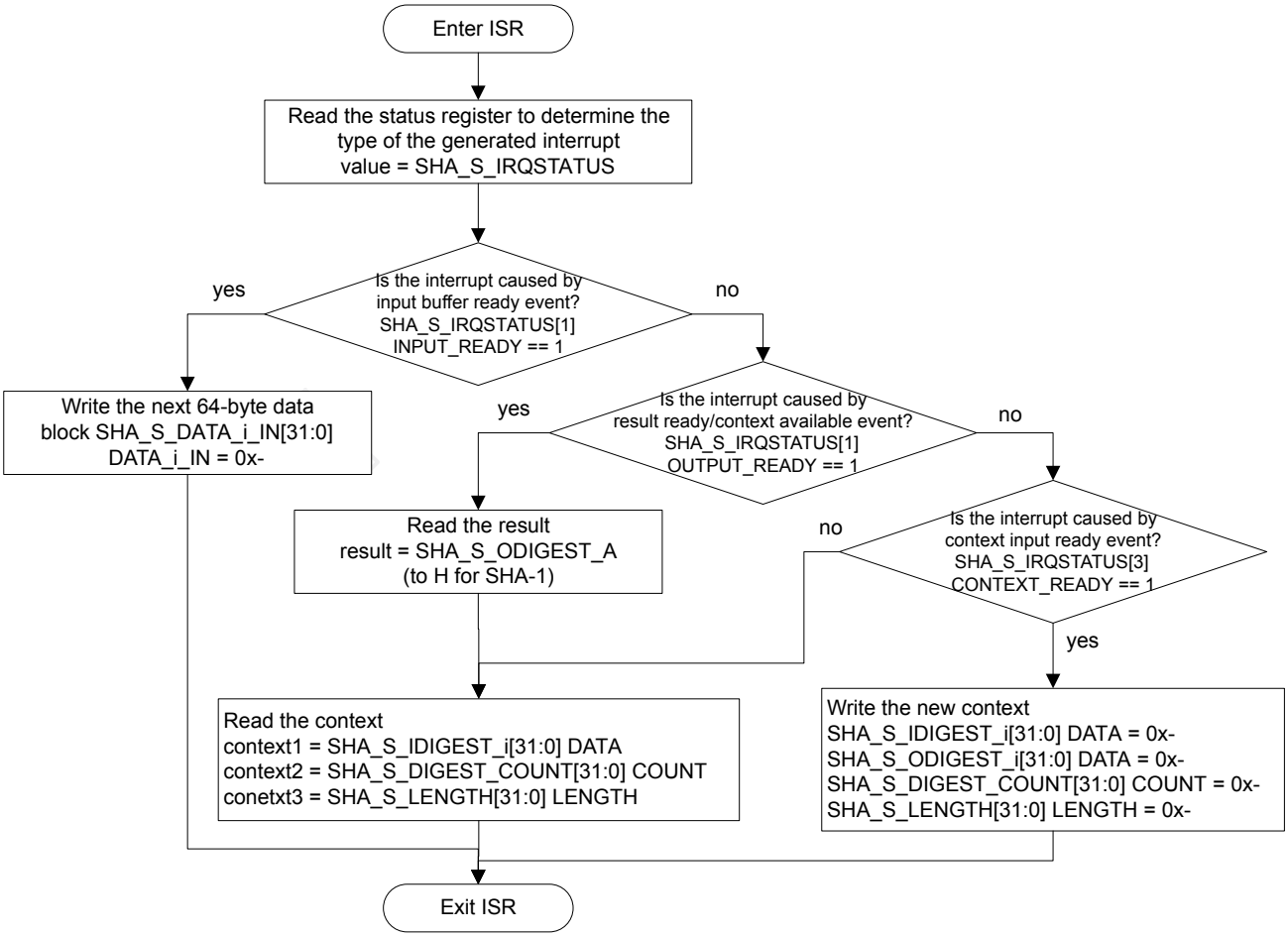
| Step | Register/Bit Field/Programming Model | Value |
|---|---|---|
| Enable the DMA request to the CDMA controller. | SHA_SYSCONFIG[3] DMA_EN | 0x1 |
| Load the message length; this is the trigger to start processing. | SHA_LENGTH[31:0] LENGTH | - |

### *SHA/MD5 Event Servicing*

#### *Interrupt Servicing*

This section describes the interrupt event servicing of the module. Figure 15-3 on page 1083 shows the interrupt subroutine. The following registers are used in the SHA/MD5 interrupt routine: SHA_IRQSTATUS, SHA_DATA_n_IN, SHA_ODIGEST_A, SHA_DIGEST_COUNT, SHA_LENGTH, and SHA_IDIGEST_A.

**Figure 15-3. SHA/MD5 Interrupt Subroutine**




## 15.2 SHA/MD5 Register Map

Table 15-12 on page 1083 lists the SHA/MD5 registers. The SHA Module comprises registers that exist at an offset relative to the SHA/MD5 Module base address and a small set of SHA/MD5 µDMA interrupt registers that exist at an offset relative to an CRC and Cryptographic Modules base address. The SHA/MD5 Module register offsets are relative to the base address 0x4403.4000. The SHA µDMA offsets are relative to the base address 0x4403.0000 .

**Table 15-12. SHA/MD5 Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **SHA/MD5 Module Registers (SHA/MD5 Module Offset)** | | | | | |
| 0x000 | SHA_ODIGEST_A | RW | 0x0000.0000 | SHA Outer Digest A | 1087 |
| 0x004 | SHA_ODIGEST_B | RW | 0x0000.0000 | SHA Outer Digest B | 1087 |
| 0x008 | SHA_ODIGEST_C | RW | 0x0000.0000 | SHA Outer Digest C | 1087 |
| 0x00C | SHA_ODIGEST_D | RW | 0x0000.0000 | SHA Outer Digest D | 1087 |

**Table 15-12. SHA/MD5 Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x010 | SHA_ODIGEST_E | RW | 0x0000.0000 | SHA Outer Digest E | 1087 |
| 0x014 | SHA_ODIGEST_F | RW | 0x0000.0000 | SHA Outer Digest F | 1087 |
| 0x018 | SHA_ODIGEST_G | RW | 0x0000.0000 | SHA Outer Digest G | 1087 |
| 0x01C | SHA_ODIGEST_H | RW | 0x0000.0000 | SHA Outer Digest H | 1087 |
| 0x020 | SHA_IDIGEST_A | RW | 0x0000.0000 | SHA Inner Digest A | 1087 |
| 0x024 | SHA_IDIGEST_B | RW | 0x0000.0000 | SHA Inner Digest B | 1087 |
| 0x028 | SHA_IDIGEST_C | RW | 0x0000.0000 | SHA Inner Digest C | 1087 |
| 0x02C | SHA_IDIGEST_D | RW | 0x0000.0000 | SHA Inner Digest D | 1087 |
| 0x030 | SHA_IDIGEST_E | RW | 0x0000.0000 | SHA Inner Digest E | 1087 |
| 0x034 | SHA_IDIGEST_F | RW | 0x0000.0000 | SHA Inner Digest F | 1087 |
| 0x038 | SHA_IDIGEST_G | RW | 0x0000.0000 | SHA Inner Digest G | 1087 |
| 0x03C | SHA_IDIGEST_H | RW | 0x0000.0000 | SHA Inner Digest H | 1087 |
| 0x040 | SHA_DIGEST_COUNT | RW | 0x0000.0000 | SHA Digest Count | 1088 |
| 0x044 | SHA_MODE | RW | 0x0000.0000 | SHA Mode | 1089 |
| 0x048 | SHA_LENGTH | RW | 0x0000.0000 | SHA Length | 1091 |
| 0x080 | SHA_DATA_0_IN | RW | 0x0000.0000 | SHA Data 0 Input | 1092 |
| 0x084 | SHA_DATA_1_IN | RW | 0x0000.0000 | SHA Data 1 Input | 1092 |
| 0x088 | SHA_DATA_2_IN | RW | 0x0000.0000 | SHA Data 2 Input | 1092 |
| 0x08C | SHA_DATA_3_IN | RW | 0x0000.0000 | SHA Data 3 Input | 1092 |
| 0x090 | SHA_DATA_4_IN | RW | 0x0000.0000 | SHA Data 4 Input | 1092 |
| 0x094 | SHA_DATA_5_IN | RW | 0x0000.0000 | SHA Data 5 Input | 1092 |
| 0x098 | SHA_DATA_6_IN | RW | 0x0000.0000 | SHA Data 6 Input | 1092 |
| 0x09C | SHA_DATA_7_IN | RW | 0x0000.0000 | SHA Data 7 Input | 1092 |
| 0x0A0 | SHA_DATA_8_IN | RW | 0x0000.0000 | SHA Data 8 Input | 1092 |
| 0x0A4 | SHA_DATA_9_IN | RW | 0x0000.0000 | SHA Data 9 Input | 1092 |
| 0x0A8 | SHA_DATA_10_IN | RW | 0x0000.0000 | SHA Data 10 Input | 1092 |
| 0x0AC | SHA_DATA_11_IN | RW | 0x0000.0000 | SHA Data 11 Input | 1092 |
| 0x0B0 | SHA_DATA_12_IN | RW | 0x0000.0000 | SHA Data 12 Input | 1092 |
| 0x0B4 | SHA_DATA_13_IN | RW | 0x0000.0000 | SHA Data 13 Input | 1092 |
| 0x0B8 | SHA_DATA_14_IN | RW | 0x0000.0000 | SHA Data 14 Input | 1092 |
| 0x0BC | SHA_DATA_15_IN | RW | 0x0000.0000 | SHA Data 15 Input | 1092 |
| 0x100 | SHA_REVISION | RO | 0x40000C03 | SHA Revision | 1093 |

**Table 15-12. SHA/MD5 Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x110 | SHA_SYSCONFIG | RW | 0x0000.0001 | SHA System Configuration | 1094 |
| 0x114 | SHA_SYSSTATUS | RO | 0x0000.0001 | SHA System Status | 1096 |
| 0x118 | SHA_IRQSTATUS | RO | 0x0000.0008 | SHA Interrupt Status | 1097 |
| 0x11C | SHA_IRQENABLE | RW | 0x0000.0007 | SHA Interrupt Enable | 1098 |
| **Encryption Control (Encryption Control Offset)** | | | | | |
| 0x010 | SHA_DMAIM | RW | 0x0000.0000 | SHA DMA Interrupt Mask | 1100 |
| 0x014 | SHA_DMARIS | RO | 0x0000.0000 | SHA DMA Raw Interrupt Status | 1101 |
| 0x018 | SHA_DMAMIS | RO | 0x0000.0000 | SHA DMA Masked Interrupt Status | 1102 |
| 0x01C | SHA_DMAIC | W1C | 0x0000.0000 | SHA DMA Interrupt Clear | 1103 |

## 15.3 SHA/MD5 Register Descriptions

This section describes the SHA/MD5 registers.

**Note:** The SHA/MD5 registers are limited to 32-bit data accesses; 8- and 16-bit accesses are not allowed and can corrupt register contents.

The first 16 registers of HIB1 are the outer and inner digest registers.

**Table 15-13. SHA/MD5 Inner/Outer Digest/HMAC Key Register Mapping**

| Register Name | Address Offset | MD5 (read/write) | SHA-1 (read/write) | SHA-2 (read/write) | | HMAC key proc (write) |
|---|---|---|---|---|---|---|
| SHA_ODIGEST_A | 0x00000000 | Outer digest [127:96] | Outer digest [159:128] | Outer digest [255:224] | | HMAC key [31,0] |
| SHA_ODIGEST_B | 0x00000004 | Outer digest [95:64] | Outer digest [127:96] | Outer digest [223:192] | | HMAC key [63,32] |
| SHA_ODIGEST_C | 0x00000008 | Outer digest [63:32] | Outer digest [95:64] | Outer digest [191:160] | | HMAC key [95,64] |
| SHA_ODIGEST_D | 0x0000000C | Outer digest [31:0] | Outer digest [63:32] | Outer digest [159:128] | | HMAC key [127,96] |
| SHA_ODIGEST_E | 0x00000010 | | Outer digest [31:0] | Outer digest [127:96] | | HMAC key [159,128] |
| SHA_ODIGEST_F | 0x00000014 | | | Outer digest [95:64] | | HMAC key [191,160] |
| SHA_ODIGEST_G | 0x00000018 | | | Outer digest [63:32] | | HMAC key [223,192] |
| SHA_ODIGEST_H | 0x0000001C | | | Outer digest [31:0] | | HMAC key [255,224] |
| SHA_IDIGEST_A | 0x00000020 | Inner digest [127:96] | Inner digest [159:128] | Inner digest [223:192] | Inner digest [255:224] | HMAC key [287,256] |
| SHA_IDIGEST_B | 0x00000024 | Inner digest [95:64] | Inner digest [127:96] | Inner digest [191:160] | Inner digest [223:192] | HMAC key [319,288] |
| SHA_IDIGEST_C | 0x00000028 | Inner digest [63:32] | Inner digest [95:64] | Inner digest [159:128] | Inner digest [191:160] | HMAC key [351,320] |

**Table 15-13. SHA/MD5 Inner/Outer Digest/HMAC Key Register Mapping *(continued)***

| Register Name | Address Offset | MD5 (read/write) | SHA-1 (read/write) | SHA-2 (read/write) | | HMAC key proc (write) |
|---|---|---|---|---|---|---|
| SHA_IDIGEST_D | 0x0000002C | Inner digest [31:0] | Inner digest [63:32] | Inner digest [127:96] | Inner digest [159:128] | HMAC key [383,352] |
| SHA_IDIGEST_E | 0x00000030 | | Inner digest[31:0] | Inner digest [95:64] | Inner digest [127:96] | HMAC key [415,384] |
| SHA_IDIGEST_F | 0x00000034 | | | Inner digest [63:32] | Inner digest [95:64] | HMAC key [447,416] |
| SHA_IDIGEST_G | 0x00000038 | | | Inner digest [31:0] | Inner digest [63:32] | HMAC key [479,448] |
| SHA_IDIGEST_H | 0x0000003C | | | | Inner digest[31:0] | HMAC key [511,480] |

**Register 1: SHA Outer Digest A (SHA_ODIGEST_A), offset 0x000**

**Register 2: SHA Outer Digest B (SHA_ODIGEST_B), offset 0x004**

**Register 3: SHA Outer Digest C (SHA_ODIGEST_C), offset 0x008**

**Register 4: SHA Outer Digest D (SHA_ODIGEST_D), offset 0x00C**

**Register 5: SHA Outer Digest E (SHA_ODIGEST_E), offset 0x010**

**Register 6: SHA Outer Digest F (SHA_ODIGEST_F), offset 0x014**

**Register 7: SHA Outer Digest G (SHA_ODIGEST_G), offset 0x018**

**Register 8: SHA Outer Digest H (SHA_ODIGEST_H), offset 0x01C**

**Register 9: SHA Inner Digest A (SHA_IDIGEST_A), offset 0x020**

**Register 10: SHA Inner Digest B (SHA_IDIGEST_B), offset 0x024**

**Register 11: SHA Inner Digest C (SHA_IDIGEST_C), offset 0x028**

**Register 12: SHA Inner Digest D (SHA_IDIGEST_D), offset 0x02C**

**Register 13: SHA Inner Digest E (SHA_IDIGEST_E), offset 0x030**

**Register 14: SHA Inner Digest F (SHA_IDIGEST_F), offset 0x034**

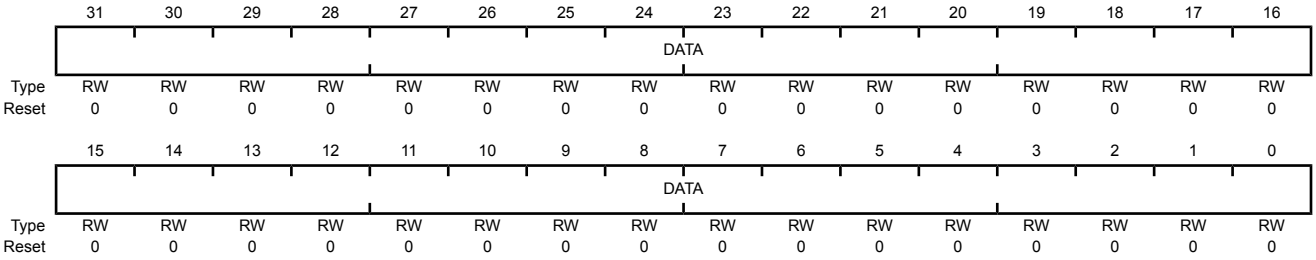
**Register 15: SHA Inner Digest G (SHA_IDIGEST_G), offset 0x038**

**Register 16: SHA Inner Digest H (SHA_IDIGEST_H), offset 0x03C**

PKA Status Register

SHA Outer Digest n (SHA_ODIGEST_n)

Base 0x4403.4000
Offset 0x000
Type RW, reset 0x0000.0000



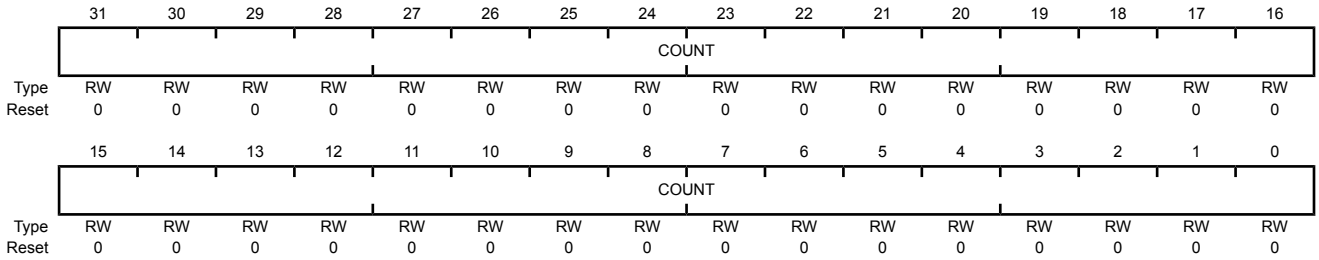

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | RW | 0x0000.0000 | Digest/Key Data |

## Register 17: SHA Digest Count (SHA_DIGEST_COUNT), offset 0x040

This register is written with the initial digest count and can be read to determine the digest count result. Note that for Initial Digest Count writes, bits 5:0 must be zero. This register is written the initial digest byte count when both the HMAC_KEY_PROC bit and the ALGO_CONSTANT bit is zero in the SHA_MODE register. When either the HMAC_KEY_PROC bit or the ALGO_CONSTANT bit is 1, this register does not need to be written because it is overwritten with 64 or 0 automatically. When starting an HMAC operation from precomputes (HMAC_KEY_PROC=0), the value 64 must be written in the **SHA_DIGESTCOUNT** register. Note that the value written should always be a 64 byte multiple, the lower 6 bits written are ignored. The updated digest byte count (initial digest byte count + bytes processed) can be read from this register when the status register indicates that the operation is done or when a µDMA context out is requested.

SHA Digest Count (SHA_DIGEST_COUNT)

Base 0x4403.4000
Offset 0x040
Type RW, reset 0x0000.0000



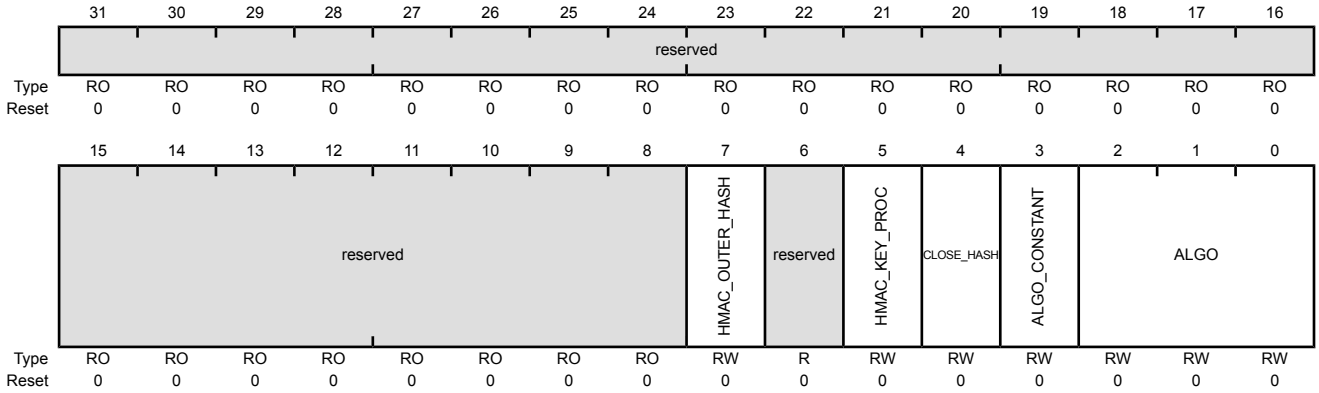

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | COUNT | RW | 0x0000.0000 | Digest Count<br>This field is written with the initial digest value for bits [31:6] only ([5:0] are assumed to be zeros).<br>When read, this outputs the result/intermediate digest count. |

## Register 18: SHA Mode (SHA_MODE), offset 0x044

This register is written to configure the hash algorithm to be used in the hash operation.

SHA Mode (SHA_MODE)

Base 0x4403.4000
Offset 0x044
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | HMAC_OUTER_HASH | RW | 0 | HMAC Outer Hash Processing Enable<br>This bit is written to indicate that the outer hash should be performed on the hash digest when the inner hash has finished. The inner hash finishes when the length of hash has been processed the final inner hash is performed (if CLOSE_HASH was set to 1).<br>This bit should normally be set together with CLOSE_HASH to finish the inner hash first, or immediately when block length is zero (HMAC continue with the just outer hash to be done). This bit is auto-cleared when outer hash is finished.<br>Value Description<br>0 No operation<br>1 Enable HMAC processing of outer hash.<br>This bit self-clears when outer hash is finished. |
| 6 | reserved | R | 0 | 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. |

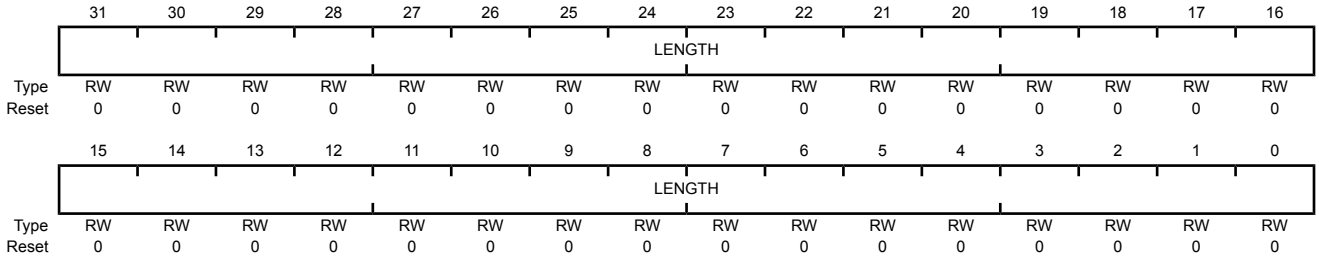
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | HMAC_KEY_PROC | RW | 0x00 | HMAC Key Processing Enable<br>This bit enables HMAC key processing on the 512-bit HMAC key loaded into the **SHA_IDIGEST_A** through **SHA_IDIGEST_H** registers and the **SHA_ODIGEST_A** through **SHA_ODIGEST_H** register block. Once HMAC key processing is finished, this bit is automatically cleared and the resulting Inner and Outer digest is available in the **SHA_IDIGEST_x** and **SHA_ODIGEST_x** respectively.<br>After key processing is complete, regular hash processing begins until the block length is exhausted.<br>Value Description<br>0 No operation<br>1 Enable HMAC key processing. This bit is automatically cleared once HMAC key processing is finished. |
| 4 | CLOSE_HASH | RW | 0 | Performs the padding, the Hash/HMAC will be 'closed' at the end of the block, as per MD5/SHA-1/SHA-2 specification (that is, appropriate padding is added), or no padding is done, allowing the hash to be continued later. However, if the Hash/HMAC is not closed, then the Block Length must be a multiple of 64 bytes to ensure correct operation. Auto cleared internally when hash closed.<br>Value Description<br>0 No padding, hash computation can be continued.<br>1 Last packet will be padded. |
| 3 | ALGO_CONSTANT | RW | 0 | The initial digest register will be overwritten with the algorithm constants for the selected algorithm when hashing and the initial digest count register will be reset to 0. This will start a normal hash operation. When continuing an existing hash or when performing an HMAC operation, this register must be set to 0 and the intermediate/inner digest or HMAC key and digest count need to be written to the context input registers prior to writing SHA_MODE. Auto cleared internally after first block processed.<br>Value Description<br>0 Use precalculated digest (from another operation)<br>1 Use constants of the selected algorithm |
| 2:0 | ALGO | RW | 0x0 | Hash Algorithm<br>Value Description<br>0x0 MD5<br>0x1 reserved<br>0x2 SHA-1<br>0x3 reserved<br>0x4 SHA-224<br>0x5 reserved<br>0x6 SHA-256<br>0x7 reserved |

## Register 19: SHA Length (SHA_LENGTH), offset 0x048

WRITE: Block Length/Remaining Byte Count (bytes) READ: Remaining Byte Count. The value programmed MUST be a 64-byte multiple if Close Hash is set to 0. This register is also the trigger to start processing: once this register is written, the core will commence requesting input data via DMA or IRQ (if programmed length > 0) and start processing. The remaining byte count for the active operation can be read from this register when the interrupt status register indicates that the operation is suspended due to a context switch request.

SHA Length (SHA_LENGTH)

Base 0x4403.4000
Offset 0x048
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LENGTH | RW | 0x0000.0000 | Block Length/Remaining Byte Count<br>This field is written with the block length in bytes of the data to be processed. When read, this field contains the remaining byte count. |

### Register 20: SHA Data 0 Input (SHA_DATA_0_IN), offset 0x080

### Register 21: SHA Data 1 Input (SHA_DATA_1_IN), offset 0x084

### Register 22: SHA Data 2 Input (SHA_DATA_2_IN), offset 0x088

### Register 23: SHA Data 3 Input (SHA_DATA_3_IN), offset 0x08C

### Register 24: SHA Data 4 Input (SHA_DATA_4_IN), offset 0x090

### Register 25: SHA Data 5 Input (SHA_DATA_5_IN), offset 0x094

### Register 26: SHA Data 6 Input (SHA_DATA_6_IN), offset 0x098

### Register 27: SHA Data 7 Input (SHA_DATA_7_IN), offset 0x09C

### Register 28: SHA Data 8 Input (SHA_DATA_8_IN), offset 0x0A0

### Register 29: SHA Data 9 Input (SHA_DATA_9_IN), offset 0x0A4

### Register 30: SHA Data 10 Input (SHA_DATA_10_IN), offset 0x0A8

### Register 31: SHA Data 11 Input (SHA_DATA_11_IN), offset 0x0AC

### Register 32: SHA Data 12 Input (SHA_DATA_12_IN), offset 0x0B0

### Register 33: SHA Data 13 Input (SHA_DATA_13_IN), offset 0x0B4

### Register 34: SHA Data 14 Input (SHA_DATA_14_IN), offset 0x0B8

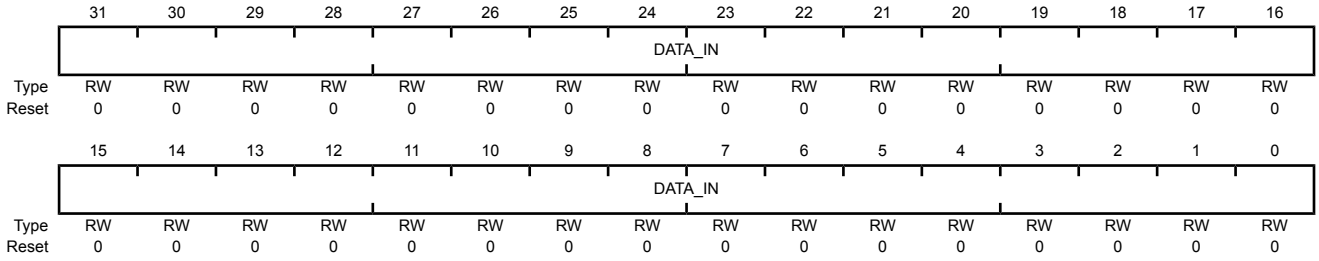
### Register 35: SHA Data 15 Input (SHA_DATA_15_IN), offset 0x0BC

Data input message

**Note:** The **SHA_DATA_n_IN_0** register acts as a FIFO and shifts data into the other **SHA_DATA_n_IN** registers.

SHA Data n Input (SHA_DATA_n_IN)

Base 0x4403.4000
Offset 0x080
Type RW, reset 0x0000.0000



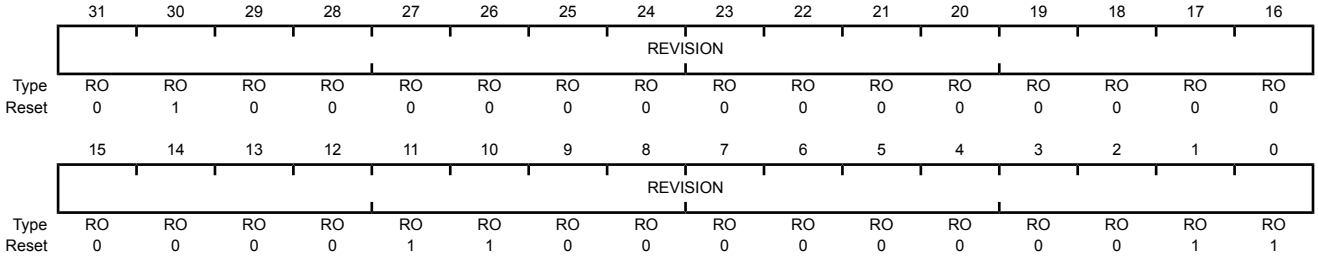

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA_IN | RW | 0x0000.0000 | Digest/Key Data |

## Register 36: SHA Revision (SHA_REVISION), offset 0x100

This register provides the unique revision number of the module. This can be used by the driver to determine the capabilities available.

SHA Revision (SHA_REVISION)

Base 0x4403.4000
Offset 0x100
Type RO, reset 0x40000C03




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | REVISION | RO | 0x40000C03 | Revision Number<br>This field indicates the revision number of the module. |

## Register 37: SHA System Configuration (SHA_SYSCONFIG), offset 0x110

System configuration register

**Note:** After one operation has completed, the **SHA_SYSCONFIG** register must be cleared and re-configured for the next operation to ensure proper µDMA and data operation functionality.

SHA System Configuration (SHA_SYSCONFIG)

Base 0x4403.4000
Offset 0x110
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | 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. |
| 7 | SADVANCED | RW | 0 | Advanced Mode Enable<br><br>Value Description<br>0 Legacy mode enabled for the Secure World. In Legacy mode, the Secure World, the context input DMA request, and the result output DMA request are masked. This means that neither DMAREQUEST_CTXIN_S and DMAREQUEST_CTXOUT_S are asserted.<br>1 Advanced mode is enabled. These DMA requests are enabled by bit 3 of this register. |
| 6 | reserved | RO | 0 | 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 | SIDLE | RW | 0 | Sidle mode<br><br>Value Description<br>0x0 Force-idle mode<br>0x1-0x3 reserved |

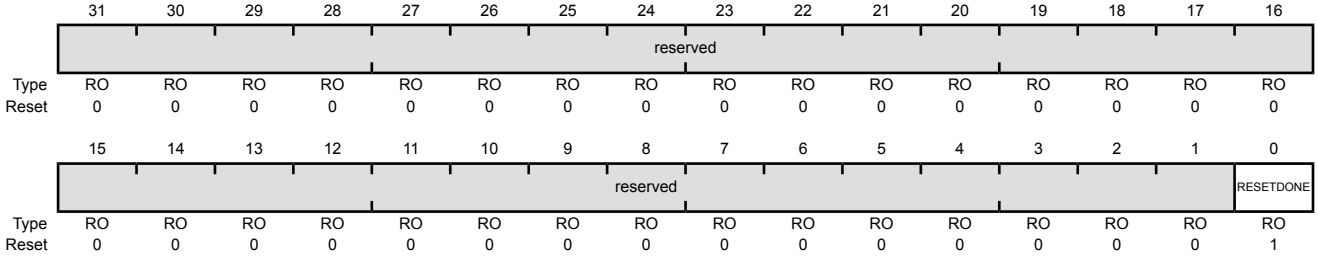
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | DMA_EN | RW | 0 | µDMA Request Enable<br>This bit controls whether the µDMA interrupts can be programmed/controlled in the **SHA_DMA_IM** register.<br>**Note:** If the µDMA is used for transferring data, then the `IT_EN` bit should be set to 0 and the **SHA_IRQENABLE** register should be clear.<br>Value Description<br>0 µDMA interrupts are disabled.<br>1 µDMA interrupts are enabled. |
| 2 | IT_EN | RW | 0 | Interrupt Enable<br>This bit controls whether the software interrupts can be programmed/controlled in the **SHA_IRQENABLE** register.<br>**Note:** When enabling the interrupts in the **SHA_IRQENABLE** register, the application should poll these interrupts and configure a software interrupt in the µDMA to handle a trigger event.<br>Value Description<br>0 SHA software Interrupts are disabled.<br>1 SHA software interrupts are enabled. |
| 1 | SOFTRESET | RW | 0 | Soft reset<br>Value Description<br>0 No operation<br>1 Start soft reset sequence |
| 0 | reserved | RO | 1 | 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. |

## Register 38: SHA System Status (SHA_SYSSTATUS), offset 0x114

System status register

SHA System Status (SHA_SYSSTATUS)

Base 0x4403.4000
Offset 0x114
Type RO, reset 0x0000.0001



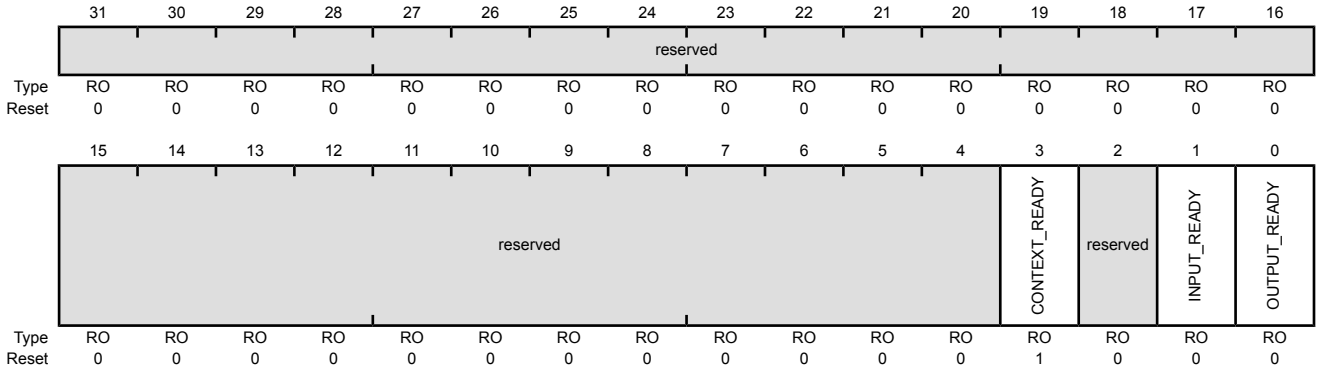

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | RESETDONE | RO | 1 | Reset done status<br>Value Description<br>0 Reset not complete<br>1 Reset complete |

## Register 39: SHA Interrupt Status (SHA_IRQSTATUS), offset 0x118

Interrupt Status Register

SHA Interrupt Status (SHA_IRQSTATUS)

Base 0x4403.4000
Offset 0x118
Type RO, reset 0x0000.0008



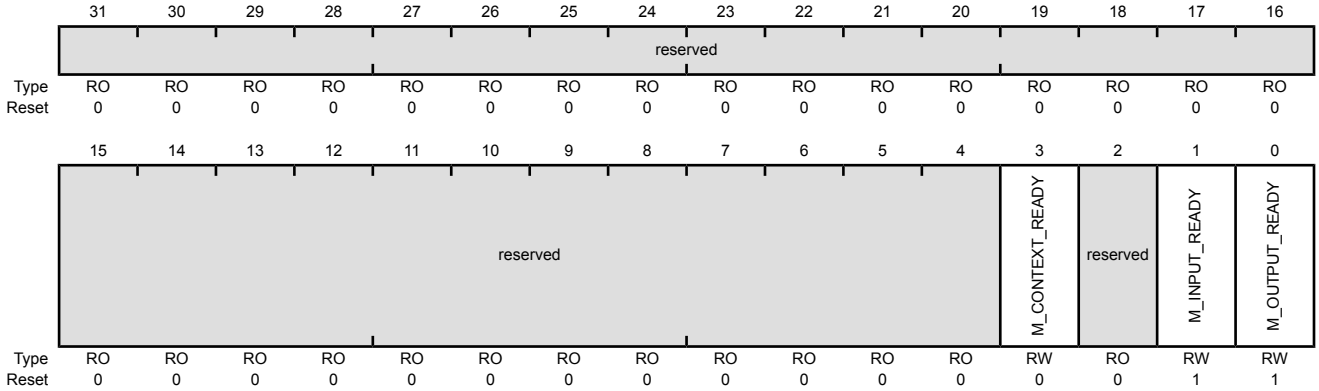

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | CONTEXT_READY | RO | 1 | Context Ready Status<br>Value Description<br>0 The context registers are not available for a new context.<br>1 The context input registers are available for a new context for the next packet to be processed. |
| 2 | reserved | RO | 0 | 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. |
| 1 | INPUT_READY | RO | 0 | Input Ready Status<br>Value Description<br>0 The Data FIFO is not ready to receive the next 64-byte data block.<br>1 The Data FIFO (**SHA_DATA_n_IN** registers) is ready to receive the next 64-byte data block. |
| 0 | OUTPUT_READY | RO | 0 | Output Ready Status<br>Value Description<br>0 No saved context available.<br>1 A saved context is available from the context output registers. |

## Register 40: SHA Interrupt Enable (SHA_IRQENABLE), offset 0x11C

The **SHA_IRQENABLE** register contains an enable bit for each unique interrupt. An interrupt is enabled when both the global enable, the IT_EN bit, in the **SHA_SYSCONFIG** register and the bit in this register are both set to 1.

SHA Interrupt Enable (SHA_IRQENABLE)

Base 0x4403.4000
Offset 0x11C
Type RW, reset 0x0000.0007




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | M_CONTEXT_READY | RW | 0 | Mask for context ready interrupt<br><br>Value Description<br>0 Context ready interrupt is disabled (masked).<br>1 Context ready interrupt is enabled. |
| 2 | reserved | RO | 0 | 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. |
| 1 | M_INPUT_READY | RW | 1 | Mask for input ready interrupt<br><br>Value Description<br>0 Input ready interrupt is disabled (masked).<br>1 Input ready interrupt is enabled. |
| 0 | M_OUTPUT_READY | RW | 1 | Mask for output ready interrupt<br><br>Value Description<br>0 Output ready interrupt is disabled (masked).<br>1 Output ready interrupt is enabled. |

## 15.4 SHA/MD5 µDMA Control Register Descriptions (Encryption Control Offset)

This section lists and describes the SHA/MD5 µDMA registers, in numerical order by address offset. Registers in this section are relative to the Encryption Control base address of 0x4403.0000.

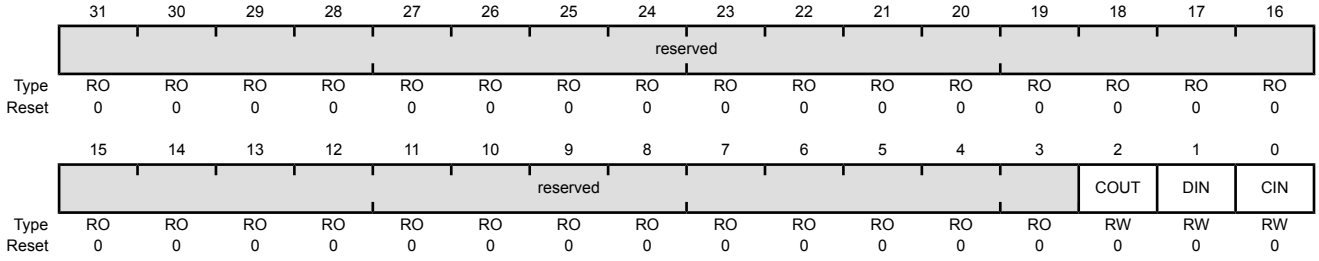
**Note:** The SHA module can only be accessed through privileged mode. If the µDMA is used for SHA transfers, then the µDMA's **DMA Channel Control (DMACHCTL)** register also needs to be programmed to allow for privileged accesses.

**Note:** If the application uses SHA µDMA interrupt triggers, the bits in the **SHA_IRQENABLE** register should be cleared.

## Register 41: SHA DMA Interrupt Mask (SHA_DMAIM), offset 0x010

The **SHA DMA Interrupt Mask (SHA_DMA_IM)** register controls interrupt behavior and are used to program which interrupts are suppressed.

SHA DMA Interrupt Mask (SHA_DMAIM)
Base
Offset 0x010
Type RW, reset 0x0000.0000



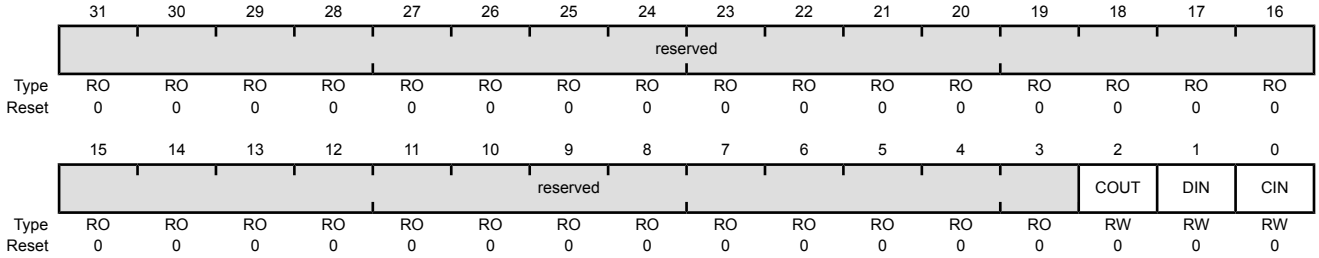

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | COUT | RW | 0 | Context Out DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA completes the output context read from the internal register.<br>Value Description<br>0 The COUT interrupt is suppressed and not sent to the interrupt controller.<br>1 The COUT interrupt is sent to the interrupt controller. |
| 1 | DIN | RW | 0 | Data In DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA writes the last word of input data to the internal FIFO of the engine.<br>Value Description<br>0 The DIN interrupt is suppressed and not sent to the interrupt controller.<br>1 The DIN interrupt is sent to the interrupt controller. |
| 0 | CIN | RW | 0 | Context In DMA Done Interrupt Mask<br>If this bit is unmasked, an interrupt is generated when the µDMA completes a context write to the internal register.<br>Value Description<br>0 The CIN interrupt is suppressed and not sent to the interrupt controller.<br>1 The CIN interrupt is sent to the interrupt controller. |

## Register 42: SHA DMA Raw Interrupt Status (SHA_DMARIS), offset 0x014

The **SHA DMA Raw Interrupt Status (SHA_DMA_RIS)** register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set to '1.'

SHA DMA Raw Interrupt Status (SHA_DMARIS)

Base
Offset 0x014
Type RO, reset 0x0000.0000



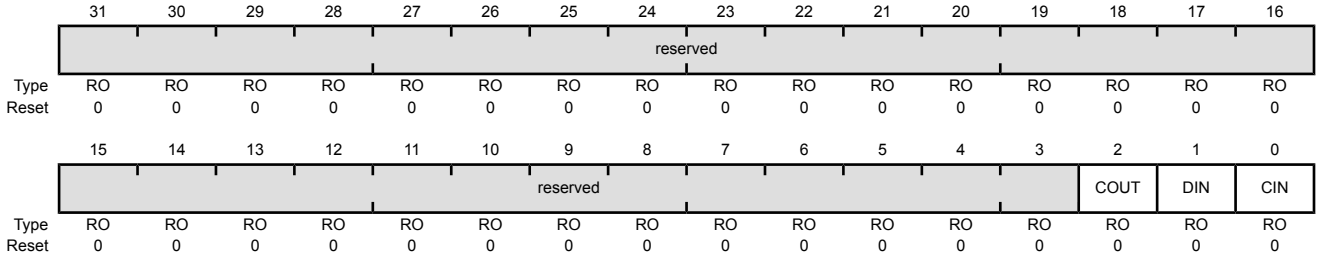

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | COUT | RW | 0 | Context Out DMA Done Raw Interrupt Status<br>Value Description<br>0 No Interrupt.<br>1 The µDMA has completed the output context read from the internal register and an interrupt has been triggered and is pending. |
| 1 | DIN | RW | 0 | Data In DMA Done Raw Interrupt Status<br>Value Description<br>0 No Interrupt.<br>1 The µDMA has written the last word of input data to the internal FIFO of the engine and an interrupt has been triggered and is pending. |
| 0 | CIN | RW | 0 | Context In DMA Done Raw Interrupt Status<br>Value Description<br>0 No interrupt.<br>1 The µDMA has completed a context write to the internal register and an interrupt has been triggered and is pending. |

## Register 43: SHA DMA Masked Interrupt Status (SHA_DMAMIS), offset 0x018

The **SHA DMA Masked Interrupt Status (SHA_DMA_MIS)** register displays the raw interrupts that are unmasked in the **SHA_DMA_RIS** register.

SHA DMA Masked Interrupt Status (SHA_DMAMIS)

Base
Offset 0x018
Type RO, reset 0x0000.0000



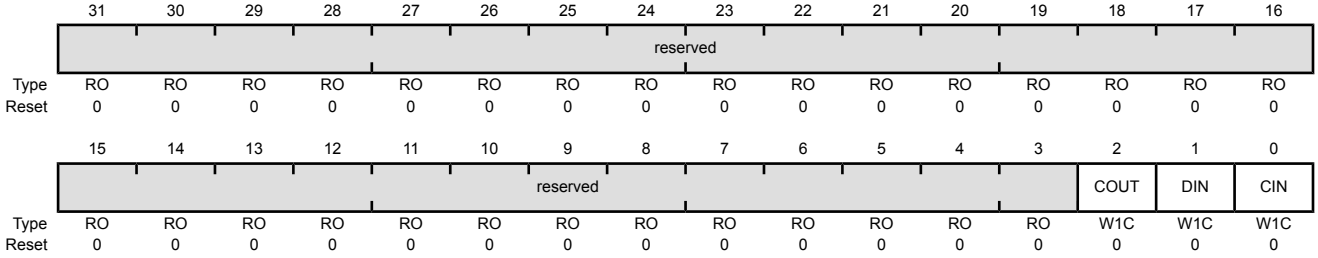

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | COUT | RO | 0 | Context Out DMA Done Masked Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A COUT interrupt has occurred. |
| 1 | DIN | RO | 0 | Data In DMA Done Masked Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A DIN interrupt has occurred. |
| 0 | CIN | RO | 0 | Context In DMA Done Raw Interrupt Status<br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 A CIN interrupt has occurred. |

## Register 44: SHA DMA Interrupt Clear (SHA_DMAIC), offset 0x01C

The **SHA DMA Interrupt Clear** register is used to clear the **SHA_DMA_RIS** and **SHA_DMA_MIS** registers by writing a 1 to each register bit.

**Note:** This registers always reads as zero.

SHA DMA Interrupt Clear (SHA_DMAIC)
Base
Offset 0x01C
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | 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. |
| 2 | COUT | W1C | 0 | Context Out DMA Done Masked Interrupt Status<br>Writing a 1 to this bit clears the `COUT` bit in the **SHA_DMA_RIS** and **SHA_DMA_MIS** register. |
| 1 | DIN | W1C | 0 | Data In DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the `DIN` bit in the **SHA_DMA_RIS** and **SHA_DMA_MIS** register. |
| 0 | CIN | W1C | 0 | Context In DMA Done Raw Interrupt Status<br>Writing a 1 to this bit clears the `CIN` bit in the **SHA_DMA_RIS** and **SHA_DMA_MIS** register.. |

# 16 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The TM4C129XKCZAD General-Purpose Timer Module (GPTM) contains 16/32-bit GPTM blocks. Each 16/32-bit GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or concatenated to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger µDMA transfers.

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

The GPT Module is one timing resource available on the Tiva™ C Series microcontrollers. Other timer resources include the System Timer (SysTick) (see 146) and the PWM timer in the PWM module (see “PWM Timer” on page 1929).

The General-Purpose Timer Module (GPTM) contains eight 16/32-bit GPTM blocks with the following functional options:

- 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
  - 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
  - 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
  - The System Clock or a global Alternate Clock (ALTCLK) resource can be used as timer clock source. The global ALTCLK can be:
    - PIOSC
    - Hibernation Module Real-time clock output (RTCOSC)
    - Low-frequency internal oscillator (LFIOSC)
- Count up or down
- Sixteen 16/32-bit Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- Timer synchronization allows selected timers to start counting on the same clock cycle
- ADC event trigger
- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode)

- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each timer
  - Burst request generated on timer interrupt

## 16.1 Block Diagram

In the block diagram, the specific Capture Compare PWM (CCP) pins available depend on the TM4C129XKCZAD device. See Table 16-1 on page 1105 for the available CCP pins and their timer assignments.

**Figure 16-1. GPTM Module Block Diagram**

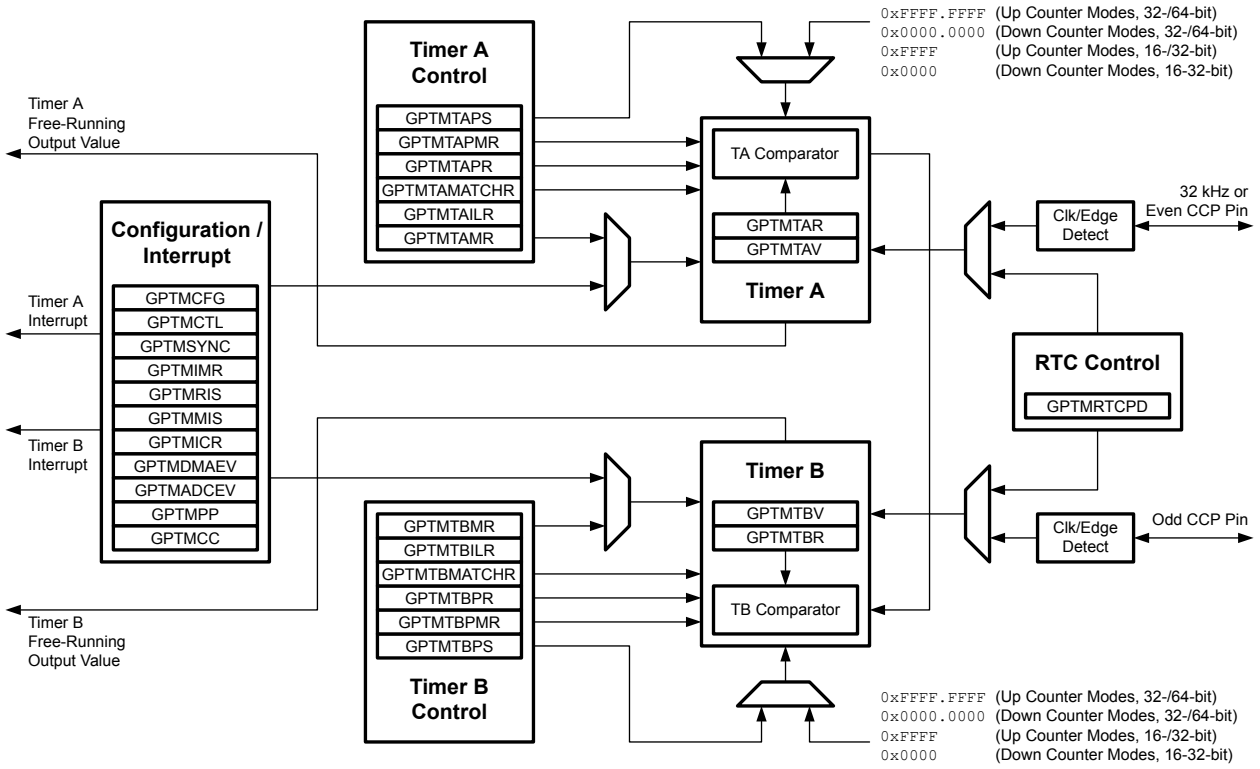



**Table 16-1. Available CCP Pins**

| Timer | Up/Down Counter | Even CCP Pin | Odd CCP Pin |
|---|---|---|---|
| 16/32-Bit Timer 0 | Timer A | T0CCP0 | - |
| | Timer B | - | T0CCP1 |
| 16/32-Bit Timer 1 | Timer A | T1CCP0 | - |
| | Timer B | - | T1CCP1 |
| 16/32-Bit Timer 2 | Timer A | T2CCP0 | - |
| | Timer B | - | T2CCP1 |

**Table 16-1. Available CCP Pins *(continued)***

| Timer | Up/Down Counter | Even CCP Pin | Odd CCP Pin |
|---|---|---|---|
| 16/32-Bit Timer 3 | Timer A | T3CCP0 | - |
| | Timer B | - | T3CCP1 |
| 16/32-Bit Timer 4 | Timer A | T4CCP0 | - |
| | Timer B | - | T4CCP1 |
| 16/32-Bit Timer 5 | Timer A | T5CCP0 | - |
| | Timer B | - | T5CCP1 |
| 16/32-Bit Timer 6 | Timer A | T6CCP0 | - |
| | Timer B | - | T6CCP1 |
| 16/32-Bit Timer 7 | Timer A | T7CCP0 | - |
| | Timer B | - | T7CCP1 |

## 16.2 Signal Description

The following table lists the external signals of the GP Timer module and describes the function of each. The GP Timer signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these GP Timer signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the GP Timer function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the GP Timer signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 16-2. General-Purpose Timers Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| T0CCP0 | V3<br>C2<br>H18<br>P3 | PA0 (3)<br>PD0 (3)<br>PL4 (3)<br>PR4 (3) | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 0. |
| T0CCP1 | W3<br>C1<br>G19<br>P2 | PA1 (3)<br>PD1 (3)<br>PL5 (3)<br>PR5 (3) | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 1. |
| T1CCP0 | T6<br>D2<br>C18<br>W9 | PA2 (3)<br>PD2 (3)<br>PL6 (3)<br>PR6 (3) | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 0. |
| T1CCP1 | U5<br>D1<br>B18<br>R10 | PA3 (3)<br>PD3 (3)<br>PL7 (3)<br>PR7 (3) | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 1. |
| T2CCP0 | V4<br>K18<br>D12 | PA4 (3)<br>PM0 (3)<br>PS0 (3) | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 0. |
| T2CCP1 | W4<br>K19<br>D13 | PA5 (3)<br>PM1 (3)<br>PS1 (3) | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 1. |

**Table 16-2. General-Purpose Timers Signals (212BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| T3CCP0 | V5<br>A4<br>L18<br>B14 | PA6 (3)<br>PD4 (3)<br>PM2 (3)<br>PS2 (3) | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 0. |
| T3CCP1 | R7<br>B4<br>L19<br>A14 | PA7 (3)<br>PD5 (3)<br>PM3 (3)<br>PS3 (3) | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 1. |
| T4CCP0 | A16<br>B3<br>M18<br>V9 | PB0 (3)<br>PD6 (3)<br>PM4 (3)<br>PS4 (3) | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 0. |
| T4CCP1 | B16<br>B2<br>G15<br>T13 | PB1 (3)<br>PD7 (3)<br>PM5 (3)<br>PS5 (3) | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 1. |
| T5CCP0 | A17<br>N19<br>U10 | PB2 (3)<br>PM6 (3)<br>PS6 (3) | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 0. |
| T5CCP1 | B17<br>N18<br>R13 | PB3 (3)<br>PM7 (3)<br>PS7 (3) | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 1. |
| T6CCP0 | F2<br>D6<br>E3<br>W10 | PB6 (3)<br>PP0 (5)<br>PQ0 (3)<br>PT0 (3) | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 0. |
| T6CCP1 | F1<br>D7<br>E2<br>V10 | PB7 (3)<br>PP1 (5)<br>PQ1 (3)<br>PT1 (3) | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 1. |
| T7CCP0 | M2<br>H4<br>E18 | PC4 (3)<br>PQ2 (3)<br>PT2 (3) | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 0. |
| T7CCP1 | M1<br>M4<br>F17 | PC5 (3)<br>PQ3 (3)<br>PT3 (3) | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 1. |

## 16.3 Functional Description

The main components of each GPTM block are two free-running up/down counters (referred to as Timer A and Timer B), two prescaler registers, two match registers, two prescaler match registers, two shadow registers, and two load/initialization registers and their associated control functions. The exact functionality 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 for the 16/32-bit GPTM blocks. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range for the 16/32-bit GPTM blocks. Note that the prescaler can only be used when the timers are used individually.

The available modes for each GPTM block are shown in Table 16-3 on page 1108. Note that when counting down in one-shot or periodic modes, the prescaler acts as a true prescaler and contains the least-significant bits 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 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 16-3. General-Purpose Timer Capabilities**

| Mode | Timer Use | Count Direction | Counter Size | Prescaler Size[a] |
|---|---|---|---|---|
| One-shot | Individual | Up or Down | 16-bit | 8-bit |
| | Concatenated | Up or Down | 32-bit | - |
| Periodic | Individual | Up or Down | 16-bit | 8-bit |
| | Concatenated | Up or Down | 32-bit | - |
| RTC | Concatenated | Up | 32-bit | - |
| Edge Count | Individual | Up or Down | 16-bit | 8-bit |
| Edge Time | Individual | Up or Down | 16-bit | 8-bit |
| PWM | Individual | Down | 16-bit | 8-bit |

a. The prescaler is only available when the timers are used individually

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 1126), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 1127), and the **GPTM Timer B Mode (GPTMTBMR)** register (see page 1132). When in one of the concatenated modes, Timer A and Timer B can only operate in one mode. However, when configured in an individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

### 16.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to all 1s, along with their corresponding registers:

- Load Registers:
  - **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 1154)
  - **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 1155)
- Shadow Registers:
  - **GPTM Timer A Value (GPTMTAV)** register (see page 1164)
  - **GPTM Timer B Value (GPTMTBV)** register (see page 1165)

The following prescale counters are initialized to all 0s:

- **GPTM Timer A Prescale (GPTMTAPR)** register (see page 1158)
- **GPTM Timer B Prescale (GPTMTBPR)** register (see page 1159)
- **GPTM Timer A Prescale Snapshot (GPTMTAPS)** register (see page 1167)
- **GPTM Timer B Prescale Snapshot (GPTMTBPS)** register (see page 1168)

### 16.3.2 Timer Clock Source

The general purpose timer has the capability of being clocked by either the system clock or an alternate clock source. By setting the ALTCLK bit in the **GPTM Clock Configuration (GPTMCC)**

register, offset 0xFC8, software can selects an alternate clock source as programmed in the **Alternate Clock Configuration (ALTCLKCFG)** register, offset 0x138 in the System Control Module. The alternate clock source options available are PIOSC, RTCOSC and LFIOSC. Refer to “System Control” on page 231 for additional information.

**Note:** When the ALTCLK bit is set in the **GPTMCC** register to enable using the alternate clock source, the synchronization imposes restrictions on the starting count value (down-count), terminal value (up-count) and the match value. This restriction applies to all modes of operation. Each event must be spaced by 4 Timer (ALTCLK) clock periods + 2 system clock periods. If some events do not meet this requirement, then it is possible that the timer block may need to be reset for correct functionality to be restored.

Example: ALTCLK= $T_{PIOSC}$ = 62.5ns (16Mhz Trimmed)

$T_{hclk}$ = 1us (1Mhz)

4*62.5ns + 2*1us = 2.25us 2.25us/62.5ns = 36 or 0x23

The minimum values for the periodic or one-shot with a match interrupt enabled are: **GPTMTAMATCHR** = 0x23 **GPTMTAILR** = 0x46

### 16.3.3 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 Timer B control and status bits. The GPTM is placed into individual/split mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 1126). 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 timeout event in down-count mode is 0x0 and in up-count mode is the value in the **GPTM Timer n Interval Load (GPTMTnILR)** and the optional **GPTM Timer n Prescale (GPTMTnPR)** registers, with the exception of RTC mode.

#### 16.3.3.1 One-Shot/Periodic Timer Mode

The selection of one-shot or periodic mode is determined by the value written to the TnMR field of the **GPTM Timer n Mode (GPTMTnMR)** register (see page 1127). The timer is configured to count up or down using the TnCDIR bit in the **GPTMTnMR** register.

When software sets the TnEN bit in the **GPTM Control (GPTMCTL)** register (see page 1136), the timer begins counting up from 0x0 or down from its preloaded value. Alternatively, if the TnWOT bit is set in the **GPTMTnMR** register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see “Wait-for-Trigger Mode” on page 1118). Table 16-4 on page 1109 shows the values that are loaded into the timer registers when the timer is enabled.

**Table 16-4. Counter Values When the Timer is Enabled in Periodic or One-Shot Modes**

| Register | Count Down Mode | Count Up Mode |
|---|---|---|
| **GPTMTnR** | **GPTMTnILR** | 0x0 |
| **GPTMTnV** | **GPTMTnILR** in concatenated mode; **GPTMTnPR** in combination with **GPTMTnILR** in individual mode | 0x0 |
| **GPTMTnPS** | **GPTMTnPR** in individual mode; not available in concatenated mode | 0x0 in individual mode; not available in concatenated mode |

When the timer is counting down and it reaches the timeout event (0x0), the timer reloads its start value from the **GPTMTnILR** and the **GPTMTnPR** registers on the next cycle. When the timer is counting up and it reaches the timeout event (the value in the **GPTMTnILR** and the optional

**GPTMTnPR** registers), the timer reloads with 0x0. If configured to be a one-shot timer, the timer stops counting and clears the TnEN bit in the **GPTMCTL** register. If configured as a periodic timer, the timer starts counting again on the next cycle.

In periodic, snap-shot mode (TnMR field is 0x2 and the TnSNAPS bit is set in the **GPTMTnMR** register), the value of the timer at the time-out event is loaded into the **GPTMTnR** register and the value of the prescaler is loaded into the **GPTMTnPS** register. The free-running counter value is shown in the **GPTMTnV** 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. Snapshot mode is not available when the timer is configured in one-shot mode.

In addition to reloading the count value, the GPTM can generate interrupts, CCP outputs and triggers when it reaches the time-out event. The GPTM sets the TnTORIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 1146), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 1152). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register (see page 1143), the GPTM also sets the TnTOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 1149). The time-out interrupt can be disabled entirely by setting the TnCINTD bit in the **GPTM Timer n Mode (GPTMTnMR)** register. In this case, the TnTORIS bit does not even set in the **GPTMRIS** register.

By setting the TnMIE bit in the **GPTMTnMR** register, an interrupt condition can also be generated when the Timer value equals the value loaded into the **GPTM Timer n Match (GPTMTnMATCHR)** and **GPTM Timer n Prescale Match (GPTMTnPMR)** registers. 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 via TnMRIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register). Note that the interrupt status bits are not updated by the hardware unless the TnMIE bit in the **GPTMTnMR** register is set, which is different than the behavior for the time-out interrupt. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL** and the event that activates the ADC is configured in the **GPTM ADC Event (GPTMADCEV)** register. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel as well as the type of trigger enable in the **GPTM DMA Event (GPTMDMAEV)** register. See “Channel Configuration” on page 713.

The TCACT field of the **GPTM Timer n Mode (GPTMTnMR)** register can be configured to clear, set or toggle an output on a time-out event.

If software updates the **GPTMTnILR** or the **GPTMTnPR** register 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 TnILD bit in the **GPTMTnMR** register is clear. If the TnILD bit is set, the counter loads the new value after the next timeout. If software updates the **GPTMTnILR** or the **GPTMTnPR** register while the counter is counting up, the timeout event is changed on the next cycle to the new value. If software updates the **GPTM Timer n Value (GPTMTnV)** register 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 **GPTMTnMATCHR** or the **GPTMTnPMR** registers, the new values are reflected on the next clock cycle if the TnMRSU bit in the **GPTMTnMR** register is clear. If the TnMRSU bit is set, the new value will not take effect until the next timeout.

If the TnSTALL bit in the **GPTMCTL** register is set and the RTCEN bit is not set in the **GPTMCTL** register, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution. If the RTCEN bit is set, it prevents the TnSTALL bit from freezing the count when the processor is halted by the debugger.

The following table shows a variety of configurations for a 16-bit free-running timer while using the prescaler. All values assume a 120-MHz clock with Tc=8.33 ns (clock period). The prescaler can only be used when a 16/32-bit timer is configured in 16-bit mode.

**Table 16-5. 16-Bit Timer With Prescaler Configurations**

| Prescale (8-bit value) | # of Timer Clocks (Tc)[a] | Max Time | Units |
|---|---|---|---|
| 00000000 | 1 | 0.548258 | ms |
| 00000001 | 2 | 1.096517 | ms |
| 00000010 | 3 | 1.644775 | ms |
| ---------- | -- | -- | -- |
| 11111101 | 254 | 139.2576 | ms |
| 11111110 | 255 | 139.8059 | ms |
| 11111111 | 256 | 140.3541 | ms |

a. Tc is the clock period.

#### *Timer Compare Action Mode*

The timer compare mode is an extension to the GPTM's existing one-shot and periodic modes. This mode can be used when an application requires a pin change state at some time in the future, regardless of the processor state. The compare mode does not operate when the PWM mode is active and is mutually exclusive to the PWM mode. The compare mode is enabled when the TAMR field is set to 0x1 or 0x2 (one-shot or periodic), the TnAMS bit is 0 (capture or compare mode) and the TCACT field is nonzero in the **GPTM Timer n Mode (GPTMTnMR)** register. Depending on the TCACT encoding, the timer can perform a set, clear or toggle on the corresponding CCPn pin when a timer match occurs. In 16-bit mode, the corresponding CCP pin can have an action applied, but when operating in 32-bit mode, the action can only be applied to the even CCP pin.

The TCACT field can be changed while the GPTM is enabled to generate different combinations of actions. For example, during a periodic event, encodings TCACT = 0x6 or 0x7 can be used to force the initial state of the CCPn pin before the first interrupt and following that, TCACT=0x2 and TCACT=0x3 can be used (alternately) to change the sense of the pin for the subsequent toggle, while possible changing load value for the next period.

The time-out interrupts used for one-shot and periodic modes are used in the compare action modes. Thus, the TnTORIS bits in the **GPTMRIS** register are triggered if the appropriate mask bits are set in the **GPTMIM** register.

### 16.3.3.2 Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the Timer A and Timer B registers are configured as an up-counter. When RTC mode is selected for the first time after reset, the counter is loaded with a value of 0x1. All subsequent load values must be written to the **GPTM Timer n Interval Load (GPTMTnILR)** registers (see page 1154). If the **GPTMTnILR** register is loaded with a new value, the counter begins counting at that value and rolls over at the fixed value of 0xFFFFFFFF. Table 16-6 on page 1111 shows the values that are loaded into the timer registers when the timer is enabled.

**Table 16-6. Counter Values When the Timer is Enabled in RTC Mode**

| Register | Count Down Mode | Count Up Mode |
|---|---|---|
| **GPTMTnR** | Not available | 0x1 |
| **GPTMTnV** | Not available | 0x1 |
| **GPTMTnPS** | Not available | Not available |

The input clock on a CCP0 input is required to be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1-Hz rate and is passed along to the input of the counter.

When software writes the TAEN bit in the **GPTMCTL** register, the counter starts counting up from its preloaded value of 0x1. When the current count value matches the preloaded value in the **GPTMTnMATCHR** registers, the GPTM asserts the RTCRIS bit in **GPTMRIS** and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When the timer value reaches the terminal count, the timer rolls over and continues counting up from 0x0. If the RTC interrupt is enabled in **GPTMIMR**, the GPTM also sets the RTCMIS bit in **GPTMMIS** and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in **GPTMICR**.

In this mode, the **GPTMTnR** and **GPTMTnV** registers always have the same value.

In addition to generating interrupts, the RTC can generate a µDMA trigger. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel as well as the type of trigger enable in the **GPTM DMA Event (GPTMDMAEV)** register. See "Channel Configuration" on page 713.

#### 16.3.3.3 Input Edge-Count Mode

**Note:** For rising-edge detection, the input signal must be High for at least two clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the frequency.

In Edge-Count mode, the timer is configured as a 24-bit up- or down-counter including the optional prescaler with the upper count value stored in the **GPTM Timer n Prescale (GPTMTnPR)** register and the lower bits in the **GPTMTnR** 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 TnCMR bit of the **GPTMTnMR** register must be cleared. The type of edge that the timer counts is determined by the TnEVENT fields of the **GPTMCTL** register. During initialization in down-count mode, the **GPTMTnMATCHR** and **GPTMTnPMR** registers are configured so that the difference between the value in the **GPTMTnILR** and **GPTMTnPR** registers and the **GPTMTnMATCHR** and **GPTMTnPMR** 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 **GPTMTnMATCHR** and **GPTMTnPMR** registers. Note that when executing an up-count, that the value of **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**. Table 16-7 on page 1112 shows the values that are loaded into the timer registers when the timer is enabled.

**Table 16-7. Counter Values When the Timer is Enabled in Input Edge-Count Mode**

| Register | Count Down Mode | Count Up Mode |
|---|---|---|
| **GPTMTnR** | **GPTMTnPR** in combination with **GPTMTnILR** | 0x0 |
| **GPTMTnV** | **GPTMTnPR** in combination with **GPTMTnILR** | 0x0 |

When software writes the TnEN bit in the **GPTM Control (GPTMCTL)** register, 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 **GPTMTnMATCHR** and **GPTMTnPMR**. When the counts match, the GPTM asserts the CnMRIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register. If the capture mode match interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register, the GPTM also sets the CnMMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. In up-count mode, the current count of the input events is held in both the **GPTMTnR** and **GPTMTnV** registers. In down-count mode, the current count of the input events can be obtained by subtracting the **GPTMTnR** or **GPTMTnV** from the value made up of the **GPTMTnPR** and **GPTMTnILR** register combination.

In addition to generating interrupts, an ADC and/or a µDMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL** and the event that activates the ADC is configured in the **GPTM ADC Event (GPTMADCEV)** register. The µDMA trigger is enabled by

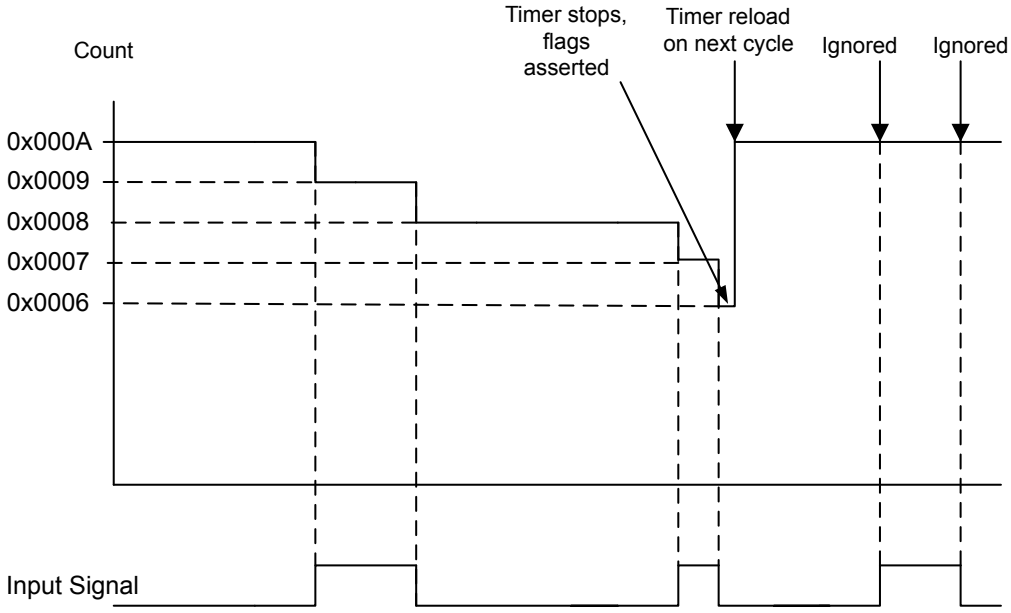
configuring and enabling the appropriate µDMA channel as well as the type of trigger enable in the **GPTM DMA Event (GPTMDMAEV)** register. See “Channel Configuration” on page 713.

After the match value is reached in down-count mode, the counter is then reloaded using the value in **GPTMTnILR** and **GPTMTnPR** registers, and stopped because the GPTM automatically clears the TnEN bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until TnEN is re-enabled by software. In up-count mode, the timer is reloaded with 0x0 and continues counting.

Figure 16-2 on page 1113 shows how Input Edge-Count mode works. In this case, the timer start value is set to **GPTMTnILR** =0x000A and the match value is set to **GPTMTnMATCHR** =0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted because the timer automatically clears the TnEN bit after the current count matches the value in the **GPTMTnMATCHR** register.

**Figure 16-2. Input Edge-Count Mode Example, Counting Down**




### 16.3.3.4 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 1/4 of the system frequency.

In Edge-Time mode, the timer is configured as a 24-bit up- or down-counter including the optional prescaler with the upper timer value stored in the **GPTMTnPR** register and the lower bits in the **GPTMTnILR** register. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR** and **GPTMTnPR** 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 TnCMR bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the TnEVENT fields of the **GPTMCTL** register. Table 16-8 on page 1114 shows the values that are loaded into the timer registers when the timer is enabled.

**Table 16-8. Counter Values When the Timer is Enabled in Input Event-Count Mode**

| Register | Count Down Mode | Count Up Mode |
|---|---|---|
| **TnR** | **GPTMTnILR** | 0x0 |
| **TnV** | **GPTMTnILR** | 0x0 |

When software writes the TnEN bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current timer counter value is captured in the **GPTMTnR** and **GPTMTnPS** register and is available to be read by the microcontroller. The GPTM then asserts the CnERIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register. If the capture mode event interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register, the GPTM also sets the CnEMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. In this mode, the **GPTMTnR** and **GPTMTnPS** registers hold the time at which the selected input event occurred while the **GPTMTnV** register holds the free-running timer value. These registers can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

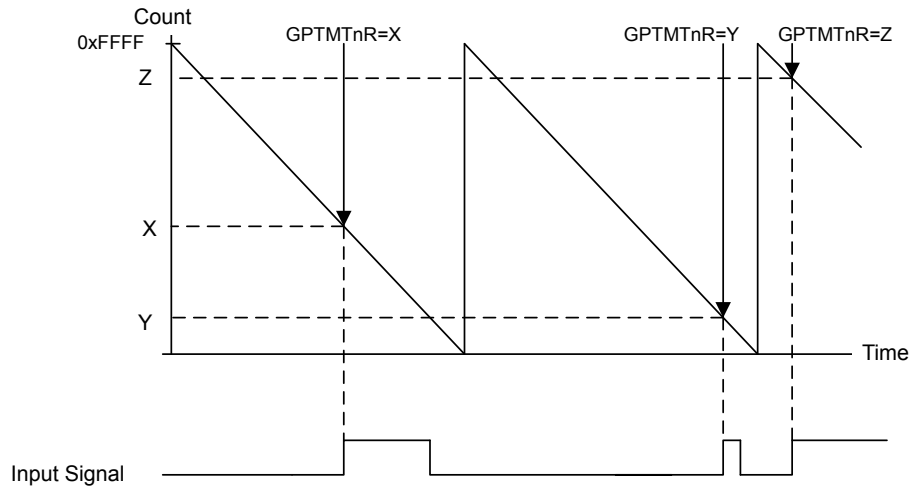
In addition to generating interrupts, an ADC and/or a µDMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL** and the event that activates the ADC is configured in the **GPTM ADC Event (GPTMADCEV)** register. The µDMA trigger is enabled by configuring the appropriate µDMA channel as well as the type of trigger selected in the **GPTM DMA Event (GPTMDMAEV)** register. See “Channel Configuration” on page 713.

After an event has been captured, the timer does not stop counting. It 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 **GPTMTnILR** and **GPTMTnPR** registers in down-count mode.

Figure 16-3 on page 1115 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 **GPTMTnR** and **GPTMTnPS** registers, and is held there until another rising edge is detected (at which point the new count value is loaded into the **GPTMTnR** and **GPTMTnPS** registers).

**Figure 16-3. 16-Bit 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 not. If there is a possibility the edge could take longer than the count, then another timer configured in periodic-timer mode can be implemented to ensure detection of the missed edge. The periodic timer should be configured in such a way that:

- The periodic timer cycles at the same rate as the edge-time timer
- The periodic timer interrupt has a higher interrupt priority than the edge-time timeout interrupt.
- If the periodic timer interrupt service routine is entered, software must check if an edge-time interrupt is pending and if it is, the value of the counter must be subtracted by 1 before being used to calculate the snapshot time of the event.

### 16.3.3.5 PWM Mode

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 24-bit down-counter with a start value (and thus period) defined by the **GPTMTnILR** and **GPTMTnPR** registers. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the **GPTMTnMR** register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2. Table 16-9 on page 1115 shows the values that are loaded into the timer registers when the timer is enabled.

**Table 16-9. Counter Values When the Timer is Enabled in PWM Mode**

| Register | Count Down Mode | Count Up Mode |
|---|---|---|
| **GPTMTnR** | **GPTMTnILR** | Not available |
| **GPTMTnV** | **GPTMTnILR** | Not available |

When software writes the TnEN bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0 state. Alternatively, if the TnWOT bit is set in the **GPTMTnMR** register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see “Wait-for-Trigger

Mode" on page 1118). On the next counter cycle in periodic mode, the counter reloads its start value from the **GPTMTnILR** and **GPTMTnPR** registers and continues counting until disabled by software clearing the TnEN bit in the **GPTMCTL** register. 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 TnEVENT field of the **GPTMCTL** register, and the interrupt is enabled by setting the TnPWMIE bit in the **GPTMTnMR** register. When the event occurs, the CnERIS bit is set in the **GPTM Raw Interrupt Status (GPTMRIS)** register, and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register . If the capture mode event interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register , the GPTM also sets the CnEMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register. Note that the interrupt status bits are not updated unless the TnPWMIE bit is set.

In addition, when the TnPWMIE bit is set and a capture event occurs, the Timer automatically generates triggers to the ADC and DMA if the trigger capability is enabled by setting the TnOTE bit in the **GPTMCTL** register and the CnEDMAEN bit in the **GPTMDMAEV** register, respectively.

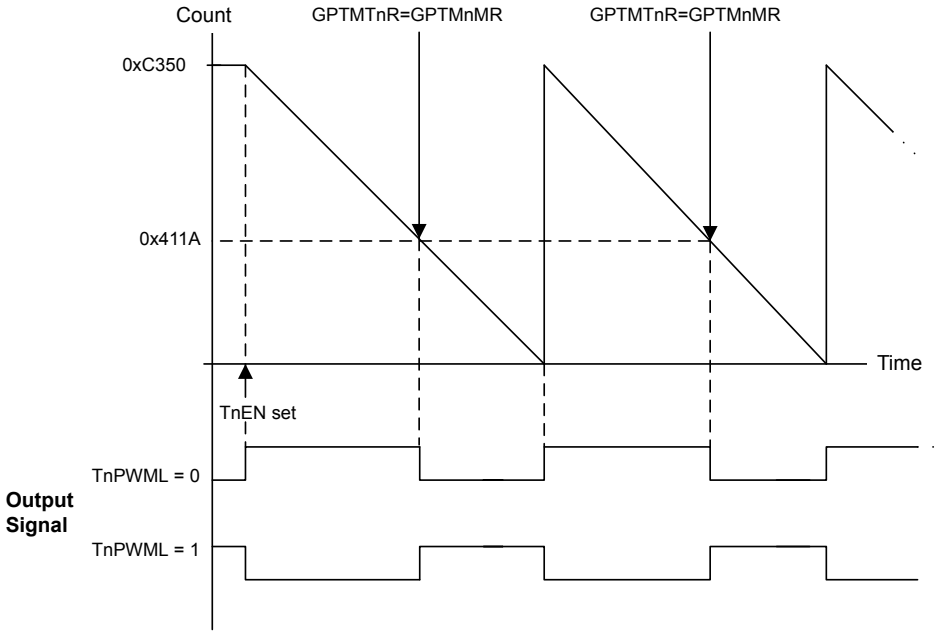
In this mode, the **GPTMTnR** and **GPTMTnV** registers always have the same value.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** and **GPTMTnPR** registers (its start state), and is deasserted when the counter value equals the value in the **GPTMTnMATCHR** and **GPTMTnPMR** registers. Software has the capability of inverting the output PWM signal by setting the TnPWML bit in the **GPTMCTL** register.

**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 positive edge, no event-trigger interrupt asserts. Instead, the interrupt is generated on the negative edge of the PWM signal.

Figure 16-4 on page 1117 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 **GPTMTnILR**=0xC350 and the match value is **GPTMTnMATCHR**=0x411A.

**Figure 16-4. 16-Bit PWM Mode Example**




When synchronizing the timers using the **GPTMSYNC** register, the timer must be properly configured to avoid glitches on the CCP outputs. Both the TnPLO and the TnMRSU bits must be set in the **GPTMTnMR** register. Figure 16-5 on page 1117 shows how the CCP output operates when the TnPLO and TnMRSU bits are set and the **GPTMTnMATCHR** value is greater than the **GPTMTnILR** value.

**Figure 16-5. CCP Output, GPTMTnMATCHR > GPTMTnILR**

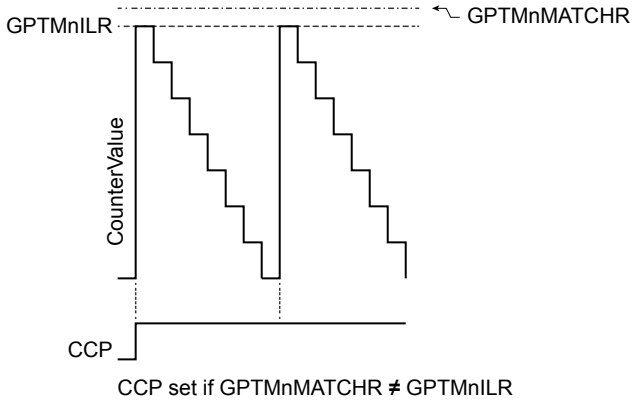



Figure 16-6 on page 1118 shows how the CCP output operates when the PLO and MRSU bits are set and the **GPTMTnMATCHR** value is the same as the **GPTMTnILR** value. In this situation, if the **PLO** bit is 0, the CCP signal goes high when the **GPTMTnILR** value is loaded and the match would be essentially ignored.

**Figure 16-6. CCP Output, GPTMTnMATCHR = GPTMTnILR**

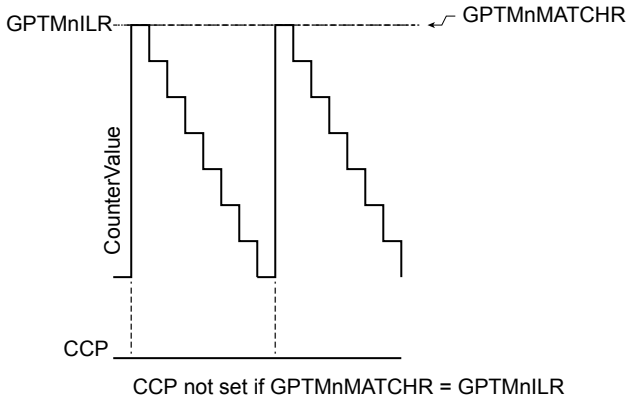
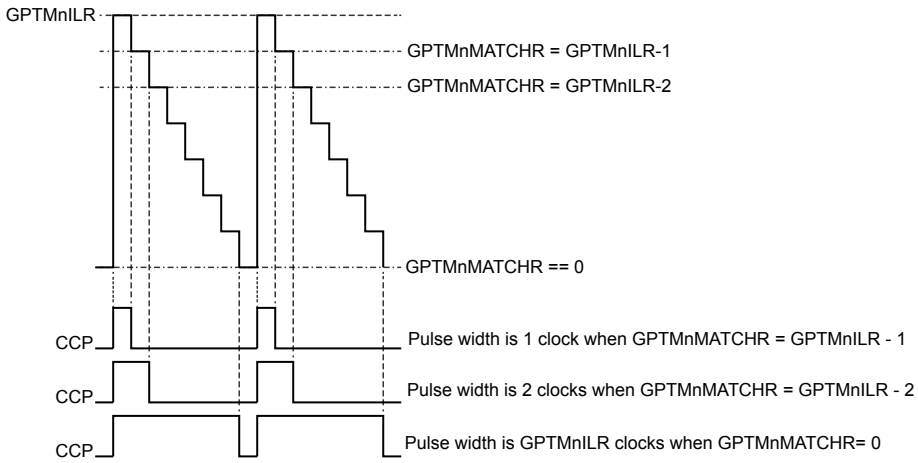



Figure 16-7 on page 1118 shows how the CCP output operates when the PLO and MRSU bits are set and the **GPTMTnILR** is greater than the **GPTMTnMATCHR** value.

**Figure 16-7. CCP Output, GPTMTnILR > GPTMTnMATCHR**

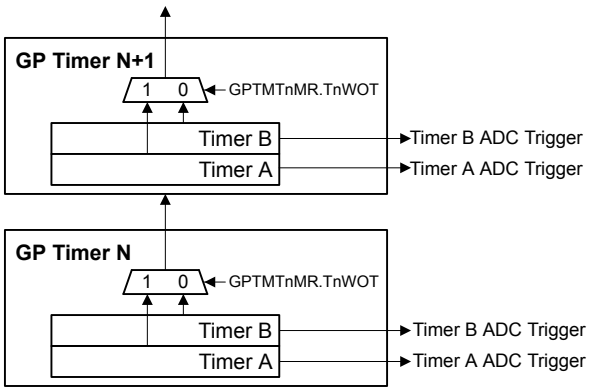



## 16.3.4 Wait-for-Trigger Mode

The 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 TnWOT bit in the **GPTMTnMR** register. 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 on. If Timer A is configured as a 32-bit (16/32-bit mode) timer (controlled by the GPTMCFG field in the **GPTMCFG** register), it triggers Timer A in the next module. If Timer A is configured as a 16-bit (16/32-bit mode) timer, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Figure 16-8 on page 1119 shows how the GPTMCFG bit affects the daisy chain. This function is valid for one-shot, periodic, and PWM modes.

**Note:** If the application requires cyclical daisy-chaining, the TAWOT bit in the **GPTMTAMR** register of Timer 0 can be set. In this case, Timer 0 waits for a trigger from the last timer module in the chain.

**Figure 16-8. Timer Daisy Chain**




## 16.3.5 Synchronizing GP Timer Blocks

The **GPTM Synchronizer Control (GPTMSYNC)** register in the GPTM0 block can be used to synchronize selected timers to begin counting at the same time. Setting a bit in the **GPTMSYNC** register causes the associated timer to perform the actions of a timeout 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 **GPTMSYNC** register.

**Note:** All timers must use the same clock source for this feature to work correctly.

Table 16-10 on page 1119 shows the actions for the timeout event performed when the timers are synchronized in the various timer modes.

**Table 16-10. Timeout Actions for GPTM Modes**

| Mode | Count Dir | Time Out Action |
|---|---|---|
| 32-bit One-Shot (concatenated timers) | – | N/A |
| 32-bit Periodic (concatenated timers) | Down | Count value = ILR |
| | Up | Count value = 0 |
| 32-bit RTC (concatenated timers) | Up | Count value = 0 |
| 16- bit One Shot (individual/split timers) | – | N/A |
| 16-bit Periodic (individual/split timers) | Down | Count value = ILR |
| | Up | Count value = 0 |
| 16-bit Edge-Count (individual/split timers) | Down | Count value = ILR |
| | Up | Count value = 0 |
| 16- bit Edge-Time (individual/split timers) | Down | Count value = ILR |
| | Up | Count value = 0 |
| 16-bit PWM | Down | Count value = ILR |

### 16.3.6 DMA Operation

The timers each have a dedicated µDMA channel and can provide a request signal to the µDMA controller. Pulse requests are generated by a timer via its own dma_req signal. A dma_done signal is provided from the µDMA to each timer to indicate transfer completion and trigger a µDMA done interrupt (DMAnRIS) in the **GPTM Raw Interrupt Status Register (GPTMRIS) register**. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the µDMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic timeout at 10 ms. Configure the µDMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the µDMA controller transfers 8 items, until all 256 items have been transferred. Refer to “Micro Direct Memory Access (µDMA)” on page 708 for more details about programming the µDMA controller.

A **GPTM DMA Event (GPTMDMAEV)** register is provided to enable the types of events that can cause a dma_req signal assertion by the timer module. Application software can enable a dma_req trigger for a match, capture or time-out event for each timer using the **GPTMDMAEV** register. For an individual timer, all active timer trigger events that have been enabled through the **GPTMDMAEV** register are ORed together to create a single dma_req pulse that is sent to the µDMA. When the µDMA transfer has completed, a dma_done signal is sent to the timer resulting in a DMAnRIS bit set in the **GPTMRIS** register.

### 16.3.7 ADC Operation

The timer has the capability to trigger the ADC when the TnOTE bit is set in the **GPTMCTL** register at offset 0x00C. The **GPTM ADC Event (GPTMADCEV)** register is additionally provided so that the type of ADC trigger can be defined. For example, by setting the CBMADCEN bit in the **GPTMADCEV** register, a trigger pulse will be sent to the ADC whenever a Capture Match event occurs in GPTM B. Similar to the µDMA operation, all active trigger events that have also been enabled in the **GPTMADCEV** register are ORed together to create an ADC trigger pulse.

### 16.3.8 Accessing Concatenated 16/32-Bit GPTM Register Values

The GPTM is placed into concatenated mode by writing a 0x0 or a 0x1 to the GPTMCFG bit field in the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain 16/32-bit GPTM registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 1154
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 1155
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 1162
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 1163
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 1164
- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 1165
- **GPTM Timer A Match (GPTMTAMATCHR)** register [15:0], see page 1156
- **GPTM Timer B Match (GPTMTBMATCHR)** register [15:0], see page 1157

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is:

GPTMTBILR[15:0]:GPTMTAILR[15:0]

Likewise, a 32-bit read access to **GPTMTAR** returns the value:

GPTMTBR[15:0]:GPTMTAR[15:0]

A 32-bit read access to **GPTMTAV** returns the value:

GPTMTBV[15:0]:GPTMTAV[15:0]

## 16.4 Initialization and Configuration

To use a GPTM, the appropriate TIMERn bit must be set in the **RCGCTIMER** register (see page 397 ). If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the **RCGCGPIO** register (see page 399). To find out which GPIO port to enable, refer to Table 31-4 on page 2064. Configure the PMCn fields in the **GPIOPCTL** register to assign the CCP signals to the appropriate pins (see page 820 and Table 31-5 on page 2081).

This section shows module initialization and configuration examples for each of the supported timer modes.

### 16.4.1 One-Shot/Periodic Timer Mode

The GPTM is configured for One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the TnEN bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0000.
3. Configure the TnMR field in the **GPTM Timer n Mode Register (GPTMTnMR)**:
   a. Write a value of 0x1 for One-Shot mode.
   b. Write a value of 0x2 for Periodic mode.
4. Optionally configure the TnSNAPS, TnWOT, TnMTE, and TnCDIR bits in the **GPTMTnMR** 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. In addition, if using CCP pins, the TCACT field can be programmed to configure the compare action.
5. Load the start value into the **GPTM Timer n Interval Load Register (GPTMTnILR)**.
6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the TnEN bit in the **GPTMCTL** register to enable the timer and start counting.
8. Poll the **GPTMRIS** 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 (GPTMICR)**.

If the TnMIE bit in the **GPTMTnMR** register is set, the RTCRIS bit in the **GPTMRIS** register is set, and the timer continues counting. 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.

### 16.4.2 Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on an even CCP input. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the TAEN bit is cleared) before making any changes.
2. If the timer has been operating in a different mode prior to this, clear any residual set bits in the **GPTM Timer n Mode (GPTMTnMR)** register before reconfiguring.
3. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0001.
4. Write the match value to the **GPTM Timer n Match Register (GPTMTnMATCHR)**.
5. Set/clear the RTCEN and TnSTALL bit in the **GPTM Control Register (GPTMCTL)** as needed.
6. If interrupts are required, set the RTCIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the TAEN bit in the **GPTMCTL** register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTnMATCHR** register, the GPTM asserts the RTCRIS bit in the **GPTMRIS** register and continues counting until Timer A is disabled or a hardware reset. The interrupt is cleared by writing the RTCCINT bit in the **GPTMICR** register. Note that if the **GPTMTnILR** register is loaded with a new value, the timer begins counting at this new value and continues until it reaches 0xFFFF.FFFF, at which point it rolls over.

### 16.4.3 Input Edge-Count Mode

A timer is configured to Input Edge-Count mode by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x0 and the TnMR field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. Program registers according to count direction:
   - In down-count mode, the **GPTMTnMATCHR** and **GPTMTnPMR** registers are configured so that the difference between the value in the **GPTMTnILR** and **GPTMTnPR** registers and the **GPTMTnMATCHR** and **GPTMTnPMR** 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 **GPTMTnMATCHR** and **GPTMTnPMR** registers. Note that when executing an up-count, the value of the **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**.

6. If interrupts are required, set the CnMIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
7. Set the TnEN bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
8. Poll the CnMRIS bit in the **GPTMRIS** 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 CnMCINT bit of the **GPTM Interrupt Clear (GPTMICR)** register.

When counting down in Input Edge-Count Mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the TnEN bit is cleared and repeat steps 4 through 8.

### 16.4.4 Input Edge Time Mode

A timer is configured to Input Edge Time mode by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x1 and the TnMR field to 0x3 and select a count direction by programming the TnCDIR bit.
4. Configure the type of event that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. If interrupts are required, set the CnEIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
9. Poll the CnERIS bit in the **GPTMRIS** 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 CnECINT bit of the **GPTM Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM Timer n (GPTMTnR)** register.

In Input Edge Timing mode, the timer continues running after an edge event has been detected, but the timer interval can be changed at any time by writing the **GPTMTnILR** register and clearing the TnILD bit in the **GPTMTnMR** register. The change takes effect at the next cycle after the write.

### 16.4.5 PWM Mode

A timer is configured to PWM mode using the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.

4. Configure the output state of the PWM signal (whether or not it is inverted) in the TnPWML field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. If PWM interrupts are used, configure the interrupt condition in the TnEVENT field in the **GPTMCTL** register and enable the interrupts by setting the TnPWMIE bit in the **GPTMTnMR** register. Note that edge detect interrupt behavior is reversed when the PWM output is inverted (see page 1136).
7. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
8. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the match value.
9. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

In PWM Time mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

## 16.5 Register Map

Table 16-11 on page 1124 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- 16/32-bit Timer 0: 0x4003.0000
- 16/32-bit Timer 1: 0x4003.1000
- 16/32-bit Timer 2: 0x4003.2000
- 16/32-bit Timer 3: 0x4003.3000
- 16/32-bit Timer 4: 0x4003.4000
- 16/32-bit Timer 5: 0x4003.5000
- 16/32-bit Timer 6: 0x400E.0000
- 16/32-bit Timer 7: 0x400E.1000

Note that the GP Timer module clock must be enabled before the registers can be programmed (see page 397). There must be a delay of 3 system clocks after the Timer module clock is enabled before any Timer module registers are accessed.

**Table 16-11. Timers Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | GPTMCFG | RW | 0x0000.0000 | GPTM Configuration | 1126 |
| 0x004 | GPTMTAMR | RW | 0x0000.0000 | GPTM Timer A Mode | 1127 |
| 0x008 | GPTMTBMR | RW | 0x0000.0000 | GPTM Timer B Mode | 1132 |
| 0x00C | GPTMCTL | RW | 0x0000.0000 | GPTM Control | 1136 |
| 0x010 | GPTMSYNC | RW | 0x0000.0000 | GPTM Synchronize | 1140 |
| 0x018 | GPTMIMR | RW | 0x0000.0000 | GPTM Interrupt Mask | 1143 |

**Table 16-11. Timers Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x01C | GPTMRIS | RO | 0x0000.0000 | GPTM Raw Interrupt Status | 1146 |
| 0x020 | GPTMMIS | RO | 0x0000.0000 | GPTM Masked Interrupt Status | 1149 |
| 0x024 | GPTMICR | W1C | 0x0000.0000 | GPTM Interrupt Clear | 1152 |
| 0x028 | GPTMTAILR | RW | 0xFFFF.FFFF | GPTM Timer A Interval Load | 1154 |
| 0x02C | GPTMTBILR | RW | 0x0000.FFFF | GPTM Timer B Interval Load | 1155 |
| 0x030 | GPTMTAMATCHR | RW | 0xFFFF.FFFF | GPTM Timer A Match | 1156 |
| 0x034 | GPTMTBMATCHR | RW | 0x0000.FFFF | GPTM Timer B Match | 1157 |
| 0x038 | GPTMTAPR | RW | 0x0000.0000 | GPTM Timer A Prescale | 1158 |
| 0x03C | GPTMTBPR | RW | 0x0000.0000 | GPTM Timer B Prescale | 1159 |
| 0x040 | GPTMTAPMR | RW | 0x0000.0000 | GPTM TimerA Prescale Match | 1160 |
| 0x044 | GPTMTBPMR | RW | 0x0000.0000 | GPTM TimerB Prescale Match | 1161 |
| 0x048 | GPTMTAR | RO | 0xFFFF.FFFF | GPTM Timer A | 1162 |
| 0x04C | GPTMTBR | RO | 0x0000.FFFF | GPTM Timer B | 1163 |
| 0x050 | GPTMTAV | RW | 0xFFFF.FFFF | GPTM Timer A Value | 1164 |
| 0x054 | GPTMTBV | RW | 0x0000.FFFF | GPTM Timer B Value | 1165 |
| 0x058 | GPTMRTCPD | RO | 0x0000.7FFF | GPTM RTC Predivide | 1166 |
| 0x05C | GPTMTAPS | RO | 0x0000.0000 | GPTM Timer A Prescale Snapshot | 1167 |
| 0x060 | GPTMTBPS | RO | 0x0000.0000 | GPTM Timer B Prescale Snapshot | 1168 |
| 0x06C | GPTMDMAEV | RW | 0x0000.0000 | GPTM DMA Event | 1169 |
| 0x070 | GPTMADCEV | RW | 0x0000.0000 | GPTM ADC Event | 1172 |
| 0xFC0 | GPTMPP | RO | 0x0000.0070 | GPTM Peripheral Properties | 1175 |
| 0xFC8 | GPTMCC | RW | 0x0000.0000 | GPTM Clock Configuration | 1177 |

## 16.6 Register Descriptions

The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

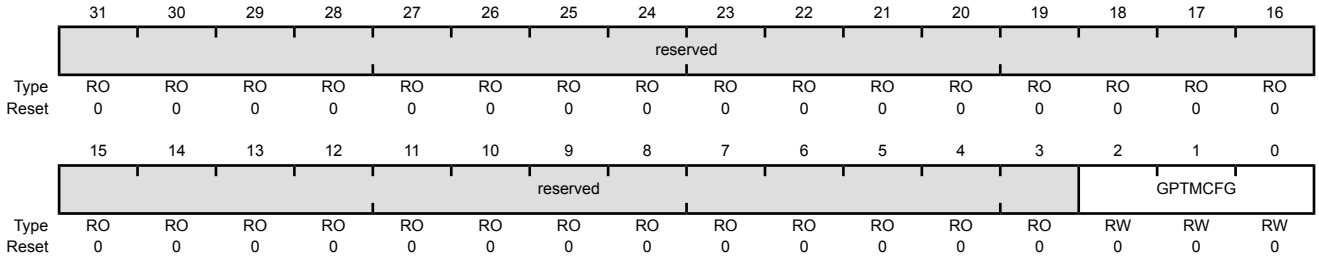
## Register 1: GPTM Configuration (GPTMCFG), offset 0x000

This register configures the global operation of the GPTM module. The value written to this register determines whether the GPTM is in 32- or 16-bit mode.

**Important:** Bits in this register should only be changed when the TAEN and TBEN bits in the **GPTMCTL** register are cleared.

GPTM Configuration (GPTMCFG)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | 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. |
| 2:0 | GPTMCFG | RW | 0x0 | GPTM Configuration<br>The GPTMCFG values are defined as follows:<br><br>Value — Description<br>0x0 — For a 16/32-bit timer, this value selects the 32-bit timer configuration.<br>0x1 — For a 16/32-bit timer, this value selects the 32-bit real-time clock (RTC) counter configuration.<br>0x2-0x3 — Reserved<br>0x4 — For a 16/32-bit timer, this value selects the 16-bit timer configuration. The function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**.<br>0x5-0x7 — Reserved |

## Register 2: GPTM Timer A Mode (GPTMTAMR), offset 0x004

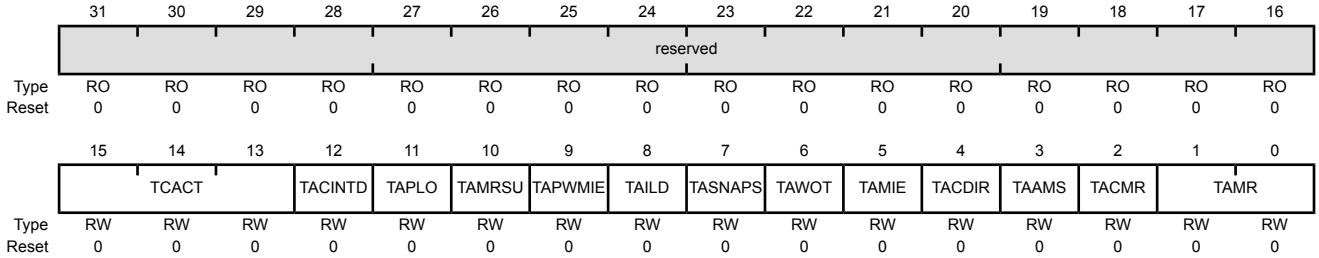
This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in PWM mode, set the TAAMS bit, clear the TACMR bit, and configure the TAMR field to 0x1 or 0x2.

This register controls the modes for Timer A when it is used individually. When Timer A and Timer B are concatenated, this register controls the modes for both Timer A and Timer B, and the contents of **GPTMTBMR** are ignored.

**Important:** Except for the TCACT bit field, all other bits in this register should only be changed when the TAEN bit in the **GPTMCTL** register is cleared.

GPTM Timer A Mode (GPTMTAMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x004
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000.00 | 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. |
| 15:13 | TCACT | RW | 0x0 | Timer Compare Action Select |

| Value | Description |
|---|---|
| 0x0 | Disable compare operations. |
| 0x1 | Toggle State on Time-Out |
| 0x2 | Clear CCP on Time-Out |
| 0x3 | Set CCP on Time-Out |
| 0x4 | Set CCP immediately and toggle on Time-Out |
| 0x5 | Clear CCP immediately and toggle on Time-Out |
| 0x6 | Set CCP immediately and clear on Time-Out |
| 0x7 | Clear CCP immediately and set on Time-Out |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | TACINTD | RW | 0 | One-shot/Periodic Interrupt Disable<br><br>Value Description<br>0 Time-out interrupt functions as normal.<br>1 Time-out interrupt are disabled.<br><br>**Note:** Setting the TACINTD bit in the **GPTMTAMR** register does not have an effect on the µDMA or ADC interrupt time-out event trigger assertions. If the TATODMAEN bit is set in the **GPTMDMAEV** register or the TATOADCEN bit is set in the **GPTMADCEV** register, a µDMA or ADC time-out trigger is sent to the µDMA or ADC, respectively, even if the TACINTD bit is set. |
| 11 | TAPLO | RW | 0 | GPTM Timer A PWM Legacy Operation<br><br>Value Description<br>0 Legacy operation with CCP pin driven Low when the **GPTMTAILR** is reloaded after the timer reaches 0.<br>1 CCP is driven High when the **GPTMTAILR** is reloaded after the timer reaches 0.<br><br>This bit is only valid in PWM mode. |
| 10 | TAMRSU | RW | 0 | GPTM Timer A Match Register Update<br><br>Value Description<br>0 Update the **GPTMTAMATCHR** register and the **GPTMTAPR** register, if used, on the next cycle.<br>1 Update the **GPTMTAMATCHR** register and the **GPTMTAPR** register, if used, on the next timeout.<br><br>If the timer is disabled (TAEN is clear) when this bit is set, **GPTMTAMATCHR** and **GPTMTAPR** are updated when the timer is enabled. If the timer is stalled (TASTALL is set), **GPTMTAMATCHR** and **GPTMTAPR** are updated according to the configuration of this bit. |
| 9 | TAPWMIE | RW | 0 | GPTM Timer A PWM Interrupt Enable<br><br>This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output, as defined by the TAEVENT field in the **GPTMCTL** register.<br><br>In addition, when this bit is set and a capture event occurs, Timer A automatically generates triggers to the ADC and DMA if the trigger capability is enabled by setting the TAOTE bit in the **GPTMCTL** register and the CAEDMAEN bit in the **GPTMDMAEV** register, respectively.<br><br>Value Description<br>0 Capture event interrupt is disabled.<br>1 Capture event interrupt is enabled.<br><br>This bit is only valid in PWM mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TAILD | RW | 0 | GPTM Timer A Interval Load Write<br><br>Value Description<br>0 Update the **GPTMTAR** and **GPTMTAV** registers with the value in the **GPTMTAILR** register on the next cycle. Also update the **GPTMTAPS** register with the value in the **GPTMTAPR** register on the next cycle.<br>1 Update the **GPTMTAR** and **GPTMTAV** registers with the value in the **GPTMTAILR** register on the next timeout. Also update the **GPTMTAPS** register with the value in the **GPTMTAPR** register on the next timeout.<br><br>Note the state of this bit has no effect when counting up.<br><br>The bit descriptions above apply if the timer is enabled and running. If the timer is disabled (TAEN is clear) when this bit is set, **GPTMTAR GPTMTAV** and **GPTMTAPs**, are updated when the timer is enabled. If the timer is stalled (TASTALL is set), **GPTMTAR** and **GPTMTAPS** are updated according to the configuration of this bit. |
| 7 | TASNAPS | RW | 0 | GPTM Timer A Snap-Shot Mode<br><br>Value Description<br>0 Snap-shot mode is disabled.<br>1 If Timer A is configured in the periodic mode, the actual free-running, capture or snapshot value of Timer A is loaded at the time-out event/capture or snapshot event into the **GPTM Timer A (GPTMTAR)** register. If the timer prescaler is used, the prescaler snapshot is loaded into the **GPTM Timer A (GPTMTAPR).** |
| 6 | TAWOT | RW | 0 | GPTM Timer A Wait-on-Trigger<br><br>**Note:** If the application requires cyclical daisy-chaining, the TAWOT bit in the **GPTMTAMR** register of Timer 0 can be set. In this case, Timer 0 waits for a trigger from the last timer module in the chain.<br><br>Value Description<br>0 Timer A begins counting as soon as it is enabled.<br>1 If Timer A is enabled (TAEN is set in the **GPTMCTL** register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see Figure 16-8 on page 1119. This function is valid for one-shot, periodic, and PWM modes. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TAMIE | RW | 0 | GPTM Timer A Match Interrupt Enable<br>Value Description<br>0 The match interrupt is disabled for match events.<br>**Note:** Clearing the TAMIE bit in the **GPTMTAMR** register prevents assertion of µDMA or ADC requests generated on a match event. Even if the TATODMAEN bit is set in the **GPTMDMAEV** register or the TATOADCEN bit is set in the **GPTMADCEV** register, a µDMA or ADC match trigger is not sent to the µDMA or ADC, respectively, when the TAMIE bit is clear.<br>1 An interrupt is generated when the match value in the **GPTMTAMATCHR** register is reached in the one-shot and periodic modes. |
| 4 | TACDIR | RW | 0 | GPTM Timer A Count Direction<br>Value Description<br>0 The timer counts down.<br>1 The timer counts up. When counting up, the timer starts from a value of 0x0.<br>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up. |
| 3 | TAAMS | RW | 0 | GPTM Timer A Alternate Mode Select<br>The TAAMS values are defined as follows:<br>Value Description<br>0 Capture or compare mode is enabled.<br>1 PWM mode is enabled.<br>**Note:** To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x1 or 0x2. |
| 2 | TACMR | RW | 0 | GPTM Timer A Capture Mode<br>The TACMR values are defined as follows:<br>Value Description<br>0 Edge-Count mode<br>1 Edge-Time mode |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | TAMR | RW | 0x0 | GPTM Timer A Mode<br>The TAMR values are defined as follows:<br><br>Value Description<br>0x0 Reserved<br>0x1 One-Shot Timer mode<br>0x2 Periodic Timer mode<br>0x3 Capture mode<br><br>The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register. |

## Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in PWM mode, set the TBAMS bit, clear the TBCMR bit, and configure the TBMR field to 0x1 or 0x2.

This register controls the modes for Timer B when it is used individually. When Timer A and Timer B are concatenated, this register is ignored and **GPTMTAMR** controls the modes for both Timer A and Timer B.

**Important:** Except for the TCACT bit field, all other bits in this register should only be changed when the TBEN bit in the **GPTMCTL** register is cleared.

GPTM Timer B Mode (GPTMTBMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x008
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCACT | | | TBCINTD | TBPLO | TBMRSU | TBPWMIE | TBILD | TBSNAPS | TBWOT | TBMIE | TBCDIR | TBAMS | TBCMR | TBMR | |
| RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000.00 | 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. |
| 15:13 | TCACT | RW | 0x0 | Timer Compare Action Select |

| Value | Description |
|---|---|
| 0x0 | Disable compare operations |
| 0x1 | Toggle State on Time-Out |
| 0x2 | Clear CCP on Time-Out |
| 0x3 | Set CCP on Time-Out |
| 0x4 | Set CCP immediately and toggle on Time-Out |
| 0x5 | Clear CCP immediately and toggle on Time-Out |
| 0x6 | Set CCP immediately and clear on Time-Out |
| 0x7 | Clear CCP immediately and set on Time-Out |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | TBCINTD | RW | 0 | One-Shot/Periodic Interrupt Disable<br><br>Value Description<br>0 Time-out interrupt functions normally<br>1 Time-out interrupt functionality is disabled<br><br>**Note:** Setting the TBCINTD bit in the **GPTMTBMR** register does not have an effect on the µDMA or ADC interrupt time-out event trigger assertions. If the TBTODMAEN bit is set in the **GPTMDMAEV** register or the TBTOADCEN bit is set in the **GPTMADCEV** register, a µDMA or ADC time-out trigger is sent to the µDMA or ADC, respectively, even if the TBCINTD bit is set. |
| 11 | TBPLO | RW | 0 | GPTM Timer B PWM Legacy Operation<br><br>Value Description<br>0 Legacy operation with CCP pin driven Low when the **GPTMTAILR** is reloaded after the timer reaches 0.<br>1 CCP is driven High when the **GPTMTAILR** is reloaded after the timer reaches 0.<br><br>This bit is only valid in PWM mode. |
| 10 | TBMRSU | RW | 0 | GPTM Timer B Match Register Update<br><br>Value Description<br>0 Update the **GPTMTBMATCHR** register and the **GPTMTBPR** register, if used, on the next cycle.<br>1 Update the **GPTMTBMATCHR** register and the **GPTMTBPR** register, if used, on the next timeout.<br><br>If the timer is disabled (TBEN is clear) when this bit is set, **GPTMTBMATCHR** and **GPTMTBPR** are updated when the timer is enabled. If the timer is stalled (TBSTALL is set), **GPTMTBMATCHR** and **GPTMTBPR** are updated according to the configuration of this bit. |
| 9 | TBPWMIE | RW | 0 | GPTM Timer B PWM Interrupt Enable<br><br>This bit enables interrupts in PWM mode on rising, falling, or both edges of the CCP output as defined by the TBEVENT field in the **GPTMCTL** register.<br><br>In addition, when this bit is set and a capture event occurs, Timer B automatically generates triggers to the ADC and DMA if the trigger capability is enabled by setting the TBOTE bit in the **GPTMCTL** register and the CBEDMAEN bit in the **GPTMDMAEV** register, respectively.<br><br>Value Description<br>0 Capture event interrupt is disabled.<br>1 Capture event is enabled.<br><br>This bit is only valid in PWM mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TBILD | RW | 0 | GPTM Timer B Interval Load Write<br><br>Value Description<br>0 Update the **GPTMTBR** and **GPTMTBV** registers with the value in the **GPTMTBILR** register on the next cycle. Also update the **GPTMTBPS** register with the value in the **GPTMTBPR** register on the next cycle.<br>1 Update the **GPTMTBR** and **GPTMTBV** registers with the value in the **GPTMTBILR** register on the next timeout. Also update the **GPTMTBPS** register with the value in the **GPTMTBPR** register on the next timeout.<br><br>Note the state of this bit has no effect when counting up.<br><br>The bit descriptions above apply if the timer is enabled and running. If the timer is disabled (TBEN is clear) when this bit is set, **GPTMTBR, GPTMTBV** and are updated when the timer is enabled. If the timer is stalled (TBSTALL is set), **GPTMTBR** and **GPTMTBPS** are updated according to the configuration of this bit. |
| 7 | TBSNAPS | RW | 0 | GPTM Timer B Snap-Shot Mode<br><br>Value Description<br>0 Snap-shot mode is disabled.<br>1 If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the **GPTM Timer B (GPTMTBR)** register. If the timer prescaler is used, the prescaler snapshot is loaded into the **GPTM Timer B (GPTMTBPR).** |
| 6 | TBWOT | RW | 0 | GPTM Timer B Wait-on-Trigger<br><br>Value Description<br>0 Timer B begins counting as soon as it is enabled.<br>1 If Timer B is enabled (TBEN is set in the **GPTMCTL** register), Timer B does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see . This function is valid for one-shot, periodic, and PWM modes. |
| 5 | TBMIE | RW | 0 | GPTM Timer B Match Interrupt Enable<br><br>Value Description<br>0 The match interrupt is disabled for match events. Additionally, triggers to the DMA and ADC on match events are prevented.<br>1 An interrupt is generated when the match value in the **GPTMTBMATCHR** register is reached in the one-shot and periodic modes.<br><br>**Note:** Clearing the TBMIE bit in the **GPTMTBMR** register prevents assertion of µDMA or ADC requests generated on a match event. Even if the TBTODMAEN bit is set in the **GPTMDMAEV** register or the TBTOADCEN bit is set in the **GPTMADCEV** register, a µDMA or ADC match trigger is not sent to the µDMA or ADC, respectively, when the TBMIE bit is clear. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | TBCDIR | RW | 0 | GPTM Timer B Count Direction<br><br>Value Description<br>0 The timer counts down.<br>1 The timer counts up. When counting up, the timer starts from a value of 0x0.<br><br>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up. |
| 3 | TBAMS | RW | 0 | GPTM Timer B Alternate Mode Select<br>The TBAMS values are defined as follows:<br><br>Value Description<br>0 Capture or compare mode is enabled.<br>1 PWM mode is enabled.<br><br>**Note:** To enable PWM mode, you must also clear the TBCMR bit and configure the TBMR field to 0x1 or 0x2. |
| 2 | TBCMR | RW | 0 | GPTM Timer B Capture Mode<br>The TBCMR values are defined as follows:<br><br>Value Description<br>0 Edge-Count mode<br>1 Edge-Time mode |
| 1:0 | TBMR | RW | 0x0 | GPTM Timer B Mode<br>The TBMR values are defined as follows:<br><br>Value Description<br>0x0 Reserved<br>0x1 One-Shot Timer mode<br>0x2 Periodic Timer mode<br>0x3 Capture mode<br><br>The timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register. |

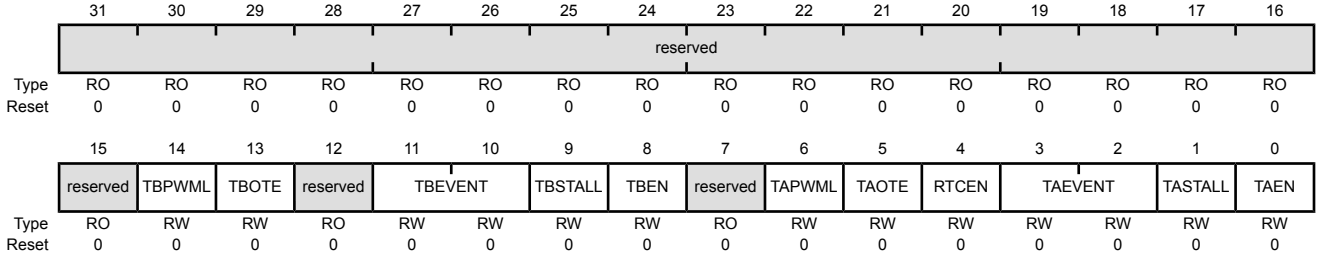
## Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

**Important:** Bits in this register should only be changed when the TnEN bit for the respective timer is cleared.

GPTM Control (GPTMCTL)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x00C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:15 | reserved | RO | 0x0000 | 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. |
| 14 | TBPWML | RW | 0 | GPTM Timer B PWM Output Level<br>The TBPWML values are defined as follows:<br><br>Value Description<br>0 Output is unaffected.<br>1 Output is inverted. |
| 13 | TBOTE | RW | 0 | GPTM Timer B Output Trigger Enable<br>The TBOTE values are defined as follows:<br><br>Value Description<br>0 The output Timer B ADC trigger is disabled.<br>1 The output Timer B ADC trigger is enabled.<br><br>In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the **ADCEMUX** register (see page 1242). |
| 12 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:10 | TBEVENT | RW | 0x0 | GPTM Timer B Event Mode<br>The TBEVENT values are defined as follows:<br><br>Value Description<br>0x0 Positive edge<br>0x1 Negative edge<br>0x2 Reserved<br>0x3 Both edges<br><br>**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. |
| 9 | TBSTALL | RW | 0 | GPTM Timer B Stall Enable<br>The TBSTALL values are defined as follows:<br><br>Value Description<br>0 Timer B continues counting while the processor is halted by the debugger.<br>1 Timer B freezes counting while the processor is halted by the debugger.<br><br>If the processor is executing normally, the TBSTALL bit is ignored. |
| 8 | TBEN | RW | 0 | GPTM Timer B Enable<br>The TBEN values are defined as follows:<br><br>Value Description<br>0 Timer B is disabled.<br>1 Timer B is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |
| 7 | reserved | RO | 0 | 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. |
| 6 | TAPWML | RW | 0 | GPTM Timer A PWM Output Level<br>The TAPWML values are defined as follows:<br><br>Value Description<br>0 Output is unaffected.<br>1 Output is inverted. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TAOTE | RW | 0 | GPTM Timer A Output Trigger Enable<br>The TAOTE values are defined as follows:<br><br>Value Description<br>0 The output Timer A ADC trigger is disabled.<br>1 The output Timer A ADC trigger is enabled.<br><br>In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the **ADCEMUX** register (see page 1242). |
| 4 | RTCEN | RW | 0 | GPTM RTC Stall Enable<br>The RTCEN values are defined as follows:<br><br>Value Description<br>0 RTC counting freezes while the processor is halted by the debugger.<br>1 RTC counting continues while the processor is halted by the debugger.<br><br>If the RTCEN bit is set, it prevents the timer from stalling in all operating modes, even if TnSTALL is set. |
| 3:2 | TAEVENT | RW | 0x0 | GPTM Timer A Event Mode<br>The TAEVENT values are defined as follows:<br><br>Value Description<br>0x0 Positive edge<br>0x1 Negative edge<br>0x2 Reserved<br>0x3 Both edges<br><br>**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. |
| 1 | TASTALL | RW | 0 | GPTM Timer A Stall Enable<br>The TASTALL values are defined as follows:<br><br>Value Description<br>0 Timer A continues counting while the processor is halted by the debugger.<br>1 Timer A freezes counting while the processor is halted by the debugger.<br><br>If the processor is executing normally, the TASTALL bit is ignored. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TAEN | RW | 0 | GPTM Timer A Enable<br>The TAEN values are defined as follows:<br><br>Value Description<br>0 Timer A is disabled.<br>1 Timer A is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register. |

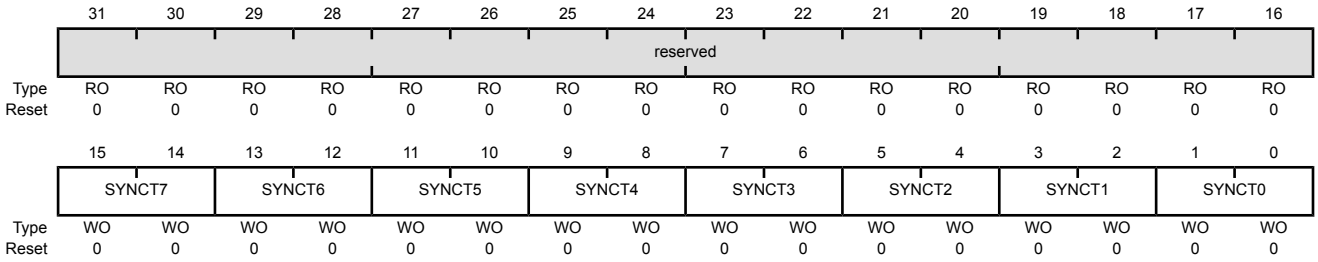
## Register 5: GPTM Synchronize (GPTMSYNC), offset 0x010

**Note:** This register is only implemented on GPTM Module 0 only.

This register allows software to synchronize a number of timers.

GPTM Synchronize (GPTMSYNC)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x010
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x00 | 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. |
| 15:14 | SYNCT7 | WO | 0x0 | Synchronize GPTM Timer 7<br><br>Value Description<br>0x0 GPT7 is not affected.<br>0x1 A timeout event for Timer A of GPTM7 is triggered.<br>0x2 A timeout event for Timer B of GPTM7 is triggered.<br>0x3 A timeout event for both Timer A and Timer B of GPTM7 is triggered. |
| 13:12 | SYNCT6 | WO | 0x0 | Synchronize GPTM Timer 6<br><br>Value Description<br>0x0 GPTM6 is not affected.<br>0x1 A timeout event for Timer A of GPTM6 is triggered.<br>0x2 A timeout event for Timer B of GPTM6 is triggered.<br>0x3 A timeout event for both Timer A and Timer B of GPTM6 is triggered. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:10 | SYNCT5 | WO | 0x0 | Synchronize GPTM Timer 5<br>Value Description<br>0x0 GPTM5 is not affected.<br>0x1 A timeout event for Timer A of GPTM5 is triggered.<br>0x2 A timeout event for Timer B of GPTM5 is triggered.<br>0x3 A timeout event for both Timer A and Timer B of GPTM5 is triggered. |
| 9:8 | SYNCT4 | WO | 0x0 | Synchronize GPTM Timer 4<br>Value Description<br>0x0 GPTM4 is not affected.<br>0x1 A timeout event for Timer A of GPTM4 is triggered.<br>0x2 A timeout event for Timer B of GPTM4 is triggered.<br>0x3 A timeout event for both Timer A and Timer B of GPTM4 is triggered. |
| 7:6 | SYNCT3 | WO | 0x0 | Synchronize GPTM Timer 3<br>Value Description<br>0x0 GPTM3 is not affected.<br>0x1 A timeout event for Timer A of GPTM3 is triggered.<br>0x2 A timeout event for Timer B of GPTM3 is triggered.<br>0x3 A timeout event for both Timer A and Timer B of GPTM3 is triggered. |
| 5:4 | SYNCT2 | WO | 0x0 | Synchronize GPTM Timer 2<br>Value Description<br>0x0 GPTM2 is not affected.<br>0x1 A timeout event for Timer A of GPTM2 is triggered.<br>0x2 A timeout event for Timer B of GPTM2 is triggered.<br>0x3 A timeout event for both Timer A and Timer B of GPTM2 is triggered. |
| 3:2 | SYNCT1 | WO | 0x0 | Synchronize GPTM Timer 1<br>Value Description<br>0x0 GPTM1 is not affected.<br>0x1 A timeout event for Timer A of GPTM1 is triggered.<br>0x2 A timeout event for Timer B of GPTM1 is triggered.<br>0x3 A timeout event for both Timer A and Timer B of GPTM1 is triggered. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | SYNCT0 | WO | 0x0 | Synchronize GPTM Timer 0 |

| Value | Description |
|---|---|
| 0x0 | GPTM0 is not affected. |
| 0x1 | A timeout event for Timer A of GPTM0 is triggered. |
| 0x2 | A timeout event for Timer B of GPTM0 is triggered. |
| 0x3 | A timeout event for both Timer A and Timer B of GPTM0 is triggered. |

## Register 6: GPTM Interrupt Mask (GPTMIMR), offset 0x018

This register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

GPTM Interrupt Mask (GPTMIMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x018
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | DMABIM | reserved | TBMIM | CBEIM | CBMIM | TBTOIM | reserved | | DMAAIM | TAMIM | RTCIM | CAEIM | CAMIM | TATOIM |
| RO | RO | RW | RO | RW | RW | RW | RW | RO | RO | RW | RW | RW | RW | RW | RW |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000 | 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. |
| 13 | DMABIM | RW | 0 | GPTM Timer B DMA Done Interrupt Mask<br>The DMABIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 12 | reserved | RO | 0x0 | 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. |
| 11 | TBMIM | RW | 0 | GPTM Timer B Match Interrupt Mask<br>The TBMIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 10 | CBEIM | RW | 0 | GPTM Timer B Capture Mode Event Interrupt Mask<br>The CBEIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | CBMIM | RW | 0 | GPTM Timer B Capture Mode Match Interrupt Mask<br>The CBMIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 8 | TBTOIM | RW | 0 | GPTM Timer B Time-Out Interrupt Mask<br>The TBTOIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 7:6 | reserved | RO | 0 | 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 | DMAAIM | RW | 0 | GPTM Timer A DMA Done Interrupt Mask<br>The DMAAIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 4 | TAMIM | RW | 0 | GPTM Timer A Match Interrupt Mask<br>The TAMIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 3 | RTCIM | RW | 0 | GPTM RTC Interrupt Mask<br>The RTCIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 2 | CAEIM | RW | 0 | GPTM Timer A Capture Mode Event Interrupt Mask<br>The CAEIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CAMIM | RW | 0 | GPTM Timer A Capture Mode Match Interrupt Mask<br>The CAMIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 0 | TATOIM | RW | 0 | GPTM Timer A Time-Out Interrupt Mask<br>The TATOIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |

## Register 7: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the **GPTMIMR** register. Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

**Note:** The state of the **GPTMRIS** register is not affected by disabling and then re-enabling the timer using the TnEN bits in the **GPTM Control (GPTMCTL)** register. If an application requires that all or certain status bits should not carry over after re-enabling the timer, then the appropriate bits in the **GPTMRIS** register should be cleared using the **GPTMICR** register prior to re-enabling the timer. If this is not done, any status bits set in the **GPTMRIS** register and unmasked in the **GPTMIMR** register generate an interrupt once the timer is re-enabled.

GPTM Raw Interrupt Status (GPTMRIS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x01C
Type RO, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | DMABRIS | reserved | TBMRIS | CBERIS | CBMRIS | TBTORIS | reserved | | DMAARIS | TAMRIS | RTCRIS | CAERIS | CAMRIS | TATORIS |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000 | 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. |
| 13 | DMABRIS | RO | 0 | GPTM Timer B DMA Done Raw Interrupt Status<br>Value Description<br>0 The Timer B DMA transfer has not completed.<br>1 The Timer B DMA transfer has completed. |
| 12 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | TBMRIS | RO | 0 | GPTM Timer B Match Raw Interrupt<br><br>Value Description<br>0 The match value has not been reached.<br>1 The TBMIE bit is set in the **GPTMTBMR** register, and the match values in the **GPTMTBMATCHR** and (optionally) **GPTMTBPMR** registers have been reached when configured in one-shot or periodic mode.<br><br>This bit is cleared by writing a 1 to the TBMCINT bit in the **GPTMICR** register. |
| 10 | CBERIS | RO | 0 | GPTM Timer B Capture Mode Event Raw Interrupt<br><br>Value Description<br>0 The capture mode event for Timer B has not occurred.<br>1 A capture mode event has occurred for Timer B. This interrupt asserts when the subtimer is configured in Input Edge-Time mode.<br><br>This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register. |
| 9 | CBMRIS | RO | 0 | GPTM Timer B Capture Mode Match Raw Interrupt<br><br>Value Description<br>0 The capture mode match for Timer B has not occurred.<br>1 The capture mode match has occurred for Timer B. This interrupt asserts when the values in the **GPTMTBR** and **GPTMTBPR** match the values in the **GPTMTBMATCHR** and **GPTMTBPMR** when configured in Input Edge-Time mode.<br><br>This bit is cleared by writing a 1 to the CBMCINT bit in the **GPTMICR** register. |
| 8 | TBTORIS | RO | 0 | GPTM Timer B Time-Out Raw Interrupt<br><br>Value Description<br>0 Timer B has not timed out.<br>1 Timer B has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into **GPTMTBILR**, depending on the count direction).<br><br>This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register. |
| 7:6 | reserved | RO | 0 | 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 | DMAARIS | RO | 0 | GPTM Timer A DMA Done Raw Interrupt Status<br><br>Value Description<br>0 The Timer A DMA transfer has not completed.<br>1 The Timer A DMA transfer has completed. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | TAMRIS | RO | 0 | GPTM Timer A Match Raw Interrupt<br><br>Value Description<br>0 The match value has not been reached.<br>1 The TAMIE bit is set in the **GPTMTAMR** register, and the match value in the **GPTMTAMATCHR** and (optionally) **GPTMTAPMR** registers have been reached when configured in one-shot or periodic mode.<br><br>This bit is cleared by writing a 1 to the TAMCINT bit in the **GPTMICR** register. |
| 3 | RTCRIS | RO | 0 | GPTM RTC Raw Interrupt<br><br>Value Description<br>0 The RTC event has not occurred.<br>1 The RTC event has occurred.<br><br>This bit is cleared by writing a 1 to the RTCCINT bit in the **GPTMICR** register. |
| 2 | CAERIS | RO | 0 | GPTM Timer A Capture Mode Event Raw Interrupt<br><br>Value Description<br>0 The capture mode event for Timer A has not occurred.<br>1 A capture mode event has occurred for Timer A. This interrupt asserts when the subtimer is configured in Input Edge-Time mode.<br><br>This bit is cleared by writing a 1 to the CAECINT bit in the **GPTMICR** register. |
| 1 | CAMRIS | RO | 0 | GPTM Timer A Capture Mode Match Raw Interrupt<br><br>Value Description<br>0 The capture mode match for Timer A has not occurred.<br>1 A capture mode match has occurred for Timer A. This interrupt asserts when the values in the **GPTMTAR** and **GPTMTAPR** match the values in the **GPTMTAMATCHR** and **GPTMTAPMR** when configured in Input Edge-Time mode.<br><br>This bit is cleared by writing a 1 to the CAMCINT bit in the **GPTMICR** register. |
| 0 | TATORIS | RO | 0 | GPTM Timer A Time-Out Raw Interrupt<br><br>Value Description<br>0 Timer A has not timed out.<br>1 Timer A has timed out. This interrupt is asserted when a one-shot or periodic mode timer reaches it's count limit (0 or the value loaded into **GPTMTAILR**, depending on the count direction).<br><br>This bit is cleared by writing a 1 to the TATOCINT bit in the **GPTMICR** register. |

## Register 8: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

GPTM Masked Interrupt Status (GPTMMIS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x020
Type RO, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | DMABMIS | reserved | TBMMIS | CBEMIS | CBMMIS | TBTOMIS | reserved | | DMAAMIS | TAMMIS | RTCMIS | CAEMIS | CAMMIS | TATOMIS |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000 | 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. |
| 13 | DMABMIS | RO | 0 | GPTM Timer B DMA Done Masked Interrupt<br><br>Value Description<br>0 A Timer B DMA done interrupt has not occurred or is masked.<br>1 An unmasked Timer B DMA done interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the DMABINT bit in the **GPTMICR** register. |
| 12 | reserved | RO | 0x0 | 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. |
| 11 | TBMMIS | RO | 0 | GPTM Timer B Match Masked Interrupt<br><br>Value Description<br>0 A Timer B Mode Match interrupt has not occurred or is masked.<br>1 An unmasked Timer B Mode Match interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the TBMCINT bit in the **GPTMICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | CBEMIS | RO | 0 | GPTM Timer B Capture Mode Event Masked Interrupt<br><br>Value Description<br>0 A Capture B event interrupt has not occurred or is masked.<br>1 An unmasked Capture B event interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register. |
| 9 | CBMMIS | RO | 0 | GPTM Timer B Capture Mode Match Masked Interrupt<br><br>Value Description<br>0 A Capture B Mode Match interrupt has not occurred or is masked.<br>1 An unmasked Capture B Match interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the CBMCINT bit in the **GPTMICR** register. |
| 8 | TBTOMIS | RO | 0 | GPTM Timer B Time-Out Masked Interrupt<br><br>Value Description<br>0 A Timer B Time-Out interrupt has not occurred or is masked.<br>1 An unmasked Timer B Time-Out interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register. |
| 7:6 | reserved | RO | 0 | 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 | DMAAMIS | RO | 0 | GPTM Timer A DMA Done Masked Interrupt<br><br>Value Description<br>0 A Timer A DMA done interrupt has not occurred or is masked.<br>1 An unmasked Timer A DMA done interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the DMAAINT bit in the **GPTMICR** register. |
| 4 | TAMMIS | RO | 0 | GPTM Timer A Match Masked Interrupt<br><br>Value Description<br>0 A Timer A Mode Match interrupt has not occurred or is masked.<br>1 An unmasked Timer A Mode Match interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the TAMCINT bit in the **GPTMICR** register. |

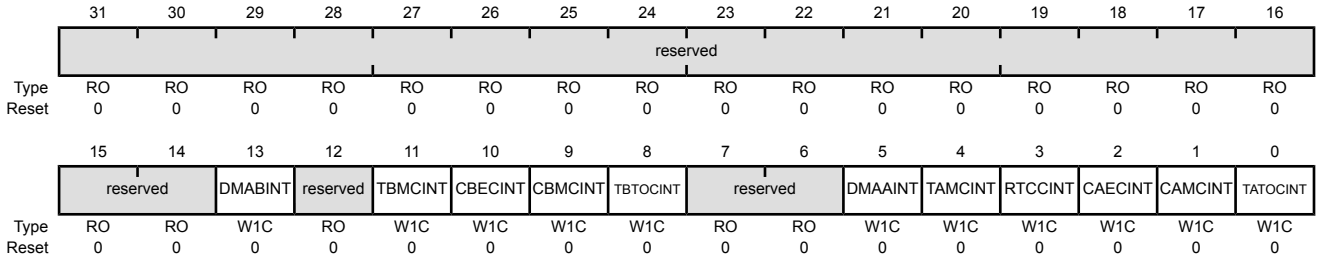
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | RTCMIS | RO | 0 | GPTM RTC Masked Interrupt<br><br>Value Description<br>0 An RTC event interrupt has not occurred or is masked.<br>1 An unmasked RTC event interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the RTCCINT bit in the **GPTMICR** register. |
| 2 | CAEMIS | RO | 0 | GPTM Timer A Capture Mode Event Masked Interrupt<br><br>Value Description<br>0 A Capture A event interrupt has not occurred or is masked.<br>1 An unmasked Capture A event interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the CAECINT bit in the **GPTMICR** register. |
| 1 | CAMMIS | RO | 0 | GPTM Timer A Capture Mode Match Masked Interrupt<br><br>Value Description<br>0 A Capture A Mode Match interrupt has not occurred or is masked.<br>1 An unmasked Capture A Match interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the CAMCINT bit in the **GPTMICR** register. |
| 0 | TATOMIS | RO | 0 | GPTM Timer A Time-Out Masked Interrupt<br><br>Value Description<br>0 A Timer A Time-Out interrupt has not occurred or is masked.<br>1 An unmasked Timer A Time-Out interrupt has occurred.<br><br>This bit is cleared by writing a 1 to the TATOCINT bit in the **GPTMICR** register. |

## Register 9: GPTM Interrupt Clear (GPTMICR), offset 0x024

This register is used to clear the status bits in the **GPTMRIS** and **GPTMMIS** registers. Writing a 1 to a bit clears the corresponding bit in the **GPTMRIS** and **GPTMMIS** registers.

GPTM Interrupt Clear (GPTMICR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x024
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000 | 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. |
| 13 | DMABINT | W1C | 0 | GPTM Timer B DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the DMABRIS bit in the **GPTMRIS** register and the DMABMIS bit in the **GPTMMIS** register. |
| 12 | reserved | RO | 0x0 | 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. |
| 11 | TBMCINT | W1C | 0 | GPTM Timer B Match Interrupt Clear<br>Writing a 1 to this bit clears the TBMRIS bit in the **GPTMRIS** register and the TBMMIS bit in the **GPTMMIS** register. |
| 10 | CBECINT | W1C | 0 | GPTM Timer B Capture Mode Event Interrupt Clear<br>Writing a 1 to this bit clears the CBERIS bit in the **GPTMRIS** register and the CBEMIS bit in the **GPTMMIS** register. |
| 9 | CBMCINT | W1C | 0 | GPTM Timer B Capture Mode Match Interrupt Clear<br>Writing a 1 to this bit clears the CBMRIS bit in the **GPTMRIS** register and the CBMMIS bit in the **GPTMMIS** register. |
| 8 | TBTOCINT | W1C | 0 | GPTM Timer B Time-Out Interrupt Clear<br>Writing a 1 to this bit clears the TBTORIS bit in the **GPTMRIS** register and the TBTOMIS bit in the **GPTMMIS** register. |
| 7:6 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | DMAAINT | W1C | 0 | GPTM Timer A DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the DMAARIS bit in the **GPTMRIS** register and the DMAAMIS bit in the **GPTMMIS** register. |
| 4 | TAMCINT | W1C | 0 | GPTM Timer A Match Interrupt Clear<br>Writing a 1 to this bit clears the TAMRIS bit in the **GPTMRIS** register and the TAMMIS bit in the **GPTMMIS** register. |
| 3 | RTCCINT | W1C | 0 | GPTM RTC Interrupt Clear<br>Writing a 1 to this bit clears the RTCRIS bit in the **GPTMRIS** register and the RTCMIS bit in the **GPTMMIS** register. |
| 2 | CAECINT | W1C | 0 | GPTM Timer A Capture Mode Event Interrupt Clear<br>Writing a 1 to this bit clears the CAERIS bit in the **GPTMRIS** register and the CAEMIS bit in the **GPTMMIS** register. |
| 1 | CAMCINT | W1C | 0 | GPTM Timer A Capture Mode Match Interrupt Clear<br>Writing a 1 to this bit clears the CAMRIS bit in the **GPTMRIS** register and the CAMMIS bit in the **GPTMMIS** register. |
| 0 | TATOCINT | W1C | 0 | GPTM Timer A Time-Out Raw Interrupt<br>Writing a 1 to this bit clears the TATORIS bit in the **GPTMRIS** register and the TATOMIS bit in the **GPTMMIS** register. |

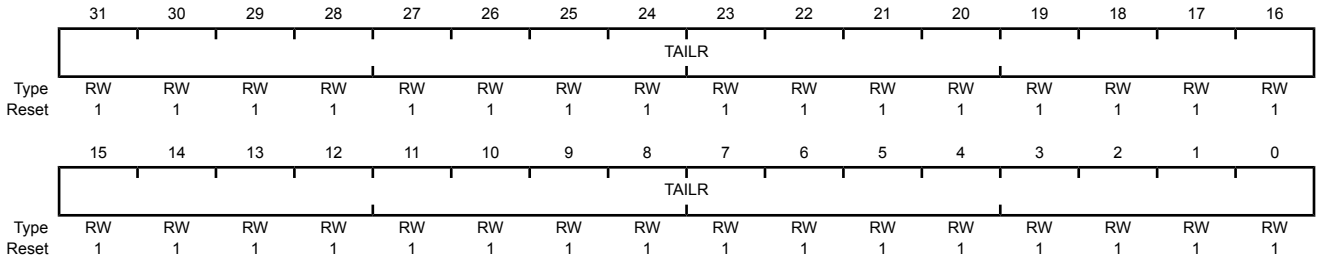
## Register 10: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAILR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Interval Load (GPTMTBILR)** register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

GPTM Timer A Interval Load (GPTMTAILR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x028
Type RW, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAILR | RW | 0xFFFF.FFFF | GPTM Timer A Interval Load Register<br>Writing this field loads the counter for Timer A. A read returns the current value of **GPTMTAILR**. |

## Register 11: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

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 **GPTMTAILR** register. Reads from this register return the current value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the load value. Bits 31:16 are reserved in both cases.

GPTM Timer B Interval Load (GPTMTBILR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x02C
Type RW, reset 0x0000.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBILR | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBILR | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TBILR | RW | 0x0000.FFFF | GPTM Timer B Interval Load Register<br>Writing this field loads the counter for Timer B. A read returns the current value of **GPTMTBILR**.<br>When a 16/32-bit GPTM is in 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**. |

## Register 12: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode.

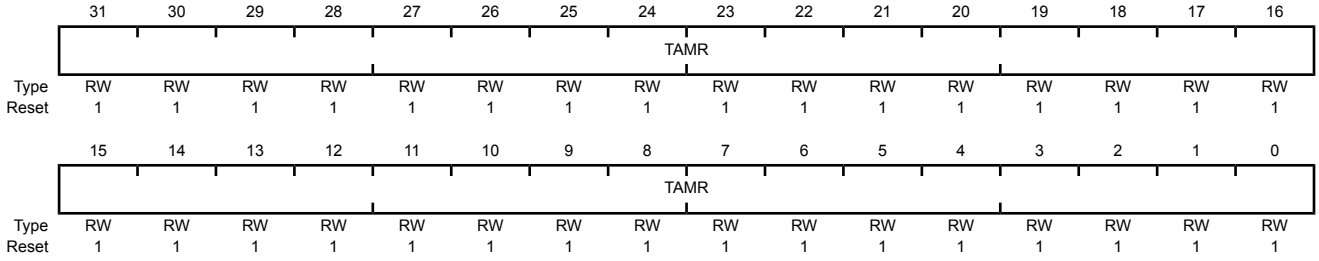
In Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value. Note that in edge-count mode, when executing an up-count, the value of **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**.

In PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, **GPTMTAMATCHR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Match (GPTMTBMATCHR)** register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBMATCHR**.

GPTM Timer A Match (GPTMTAMATCHR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x030
Type RW, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAMR | RW | 0xFFFF.FFFF | GPTM Timer A Match Register<br>This value is compared to the **GPTMTAR** register to determine match events. |

## Register 13: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034

This register is loaded with a match value. 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 **GPTMTBILR** determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTBILR** minus this value. Note that in edge-count mode, when executing an up-count, the value of **GPTMTnPR** and **GPTMTnILR** must be greater than the value of **GPTMTnPMR** and **GPTMTnMATCHR**.

In PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal.

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 **GPTMTAMATCHR** register. 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.

GPTM Timer B Match (GPTMTBMATCHR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x034
Type RW, reset 0x0000.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBMR | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBMR | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

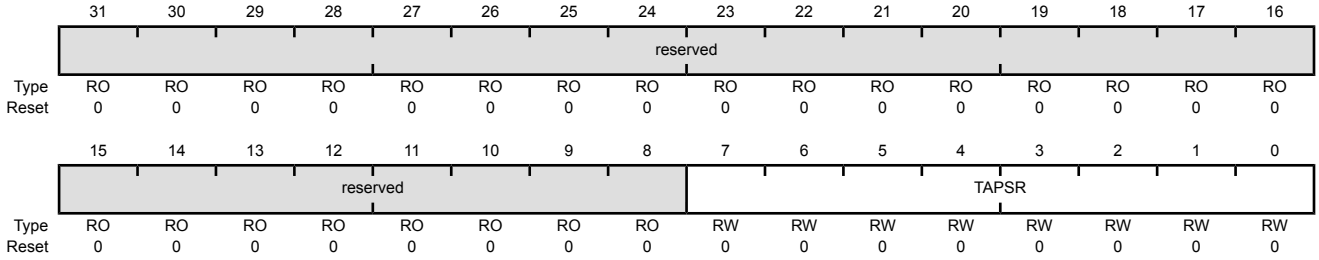
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TBMR | RW | 0x0000.FFFF | GPTM Timer B Match Register<br>This value is compared to the **GPTMTBR** register to determine match events. |

## Register 14: GPTM Timer A Prescale (GPTMTAPR), offset 0x038

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 the **GPTMTAR** and **GPTMTAV** 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 GPTM.

GPTM Timer A Prescale (GPTMTAPR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x038
Type RW, reset 0x0000.0000



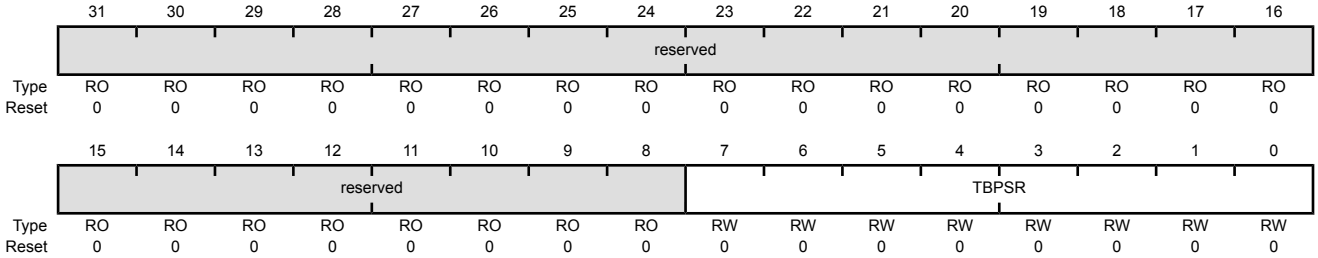

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TAPSR | RW | 0x00 | GPTM Timer A Prescale<br>The register loads this value on a write. A read returns the current value of the register.<br>For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler.<br>Refer to Table 16-5 on page 1111 for more details and an example. |

## Register 15: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C

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 the **GPTMTBR** and **GPTMTBV** 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 GPTM.

GPTM Timer B Prescale (GPTMTBPR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x03C
Type RW, reset 0x0000.0000



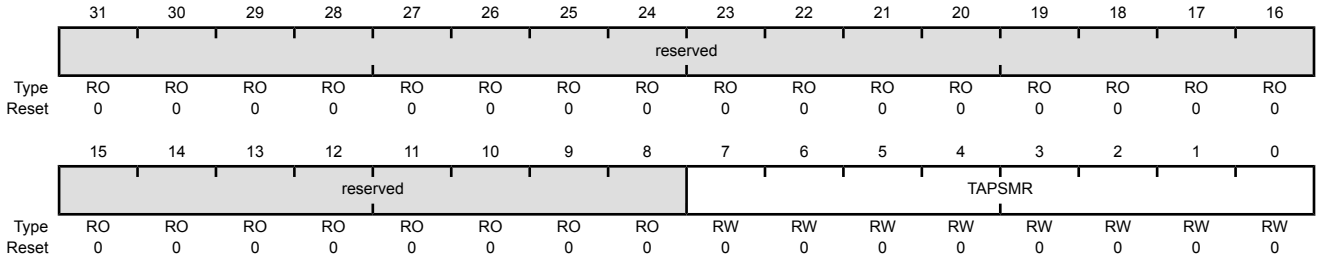

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TBPSR | RW | 0x00 | GPTM Timer B Prescale<br>The register loads this value on a write. A read returns the current value of this register.<br>For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler.<br>Refer to Table 16-5 on page 1111 for more details and an example. |

## Register 16: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040

This register allows software to extend the range of the **GPTMTAMATCHR** when the timers are used individually. This register holds bits 23:16 in the 16-bit modes of the 16/32-bit GPTM.

GPTM TimerA Prescale Match (GPTMTAPMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x040
Type RW, reset 0x0000.0000



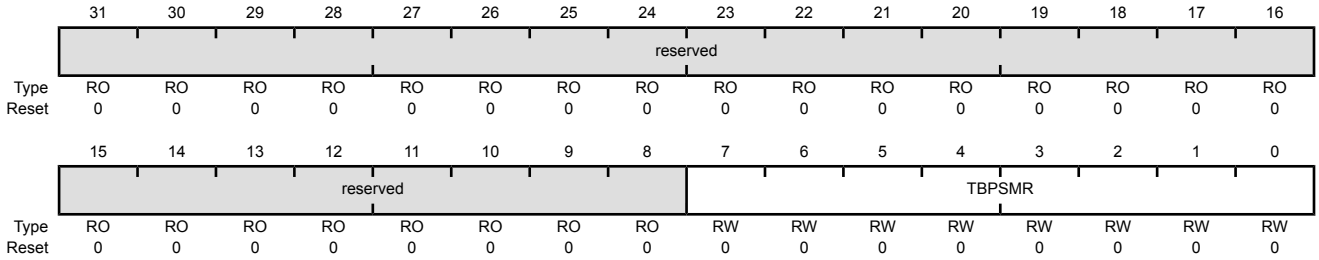

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TAPSMR | RW | 0x00 | GPTM TimerA Prescale Match<br>This value is used alongside **GPTMTAMATCHR** to detect timer match events while using a prescaler.<br>For the 16/32-bit GPTM, this field contains the entire 8-bit prescaler match value. |

## Register 17: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044

This register allows software to extend the range of the **GPTMTBMATCHR** when the timers are used individually. This register holds bits 23:16 in the 16-bit modes of the 16/32-bit GPTM.

GPTM TimerB Prescale Match (GPTMTBPMR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x044
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000 | 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. |
| 7:0 | TBPSMR | RW | 0x00 | GPTM TimerB Prescale Match<br>This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler. |

## Register 18: GPTM Timer A (GPTMTAR), offset 0x048

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.

**Note:** When an alternate clock source is enabled, a read of this register returns the current count -1.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B (GPTMTBR)** 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 **GPTMTAV** register. To read the value of the prescalar in periodic snapshot mode, read the **Timer A Prescale Snapshot (GPTMTAPS)** register.

GPTM Timer A (GPTMTAR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x048
Type RO, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAR | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAR | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAR | RO | 0xFFFF.FFFF | GPTM Timer A Register<br>A read returns the current value of the **GPTM Timer A Count Register**, 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. |

## Register 19: GPTM Timer B (GPTMTBR), offset 0x04C

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.

**Note:** When an alternate clock source is enabled, a read of this register returns the current count -1.

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 **GPTMTAR** 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 **GPTMTBV** register. To read the value of the prescalar in periodic snapshot mode, read the **Timer B Prescale Snapshot (GPTMTBPS)** register.

GPTM Timer B (GPTMTBR)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x04C
Type RO, reset 0x0000.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBR | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBR | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TBR | RO | 0x0000.FFFF | GPTM Timer B Register<br>A read returns the current value of the **GPTM Timer B Count Register**, 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. |

## Register 20: GPTM Timer A Value (GPTMTAV), offset 0x050

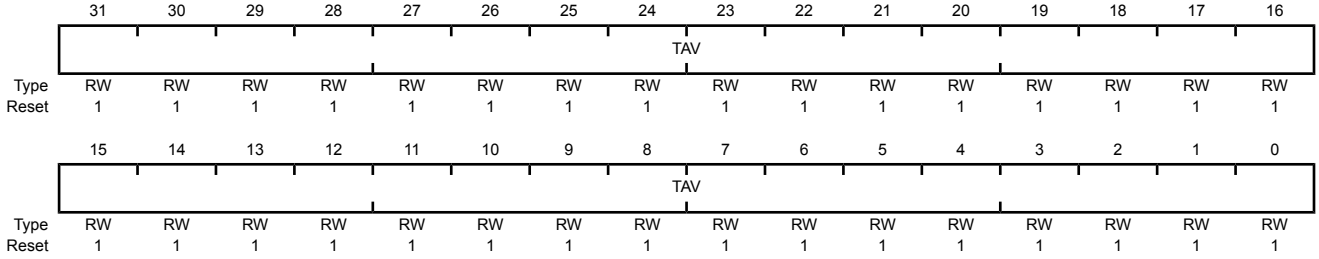
When read, this register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry when using the snapshot feature with the periodic operating mode. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle.

**Note:** When an alternate clock source is enabled, a read of this register returns the current count -1.

When a 16/32-bit GPTM is configured to one of the 32-bit modes, **GPTMTAV** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Value (GPTMTBV)** 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.

GPTM Timer A Value (GPTMTAV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x050
Type RW, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAV | RW | 0xFFFF.FFFF | GPTM Timer A Value<br>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 **GPTMTAR** register on the next clock cycle.<br>**Note:** In 16-bit mode, only the lower 16-bits of the **GPTMTAV** register can be written with a new value. Writes to the prescaler bits have no effect. |

## Register 21: GPTM Timer B Value (GPTMTBV), offset 0x054

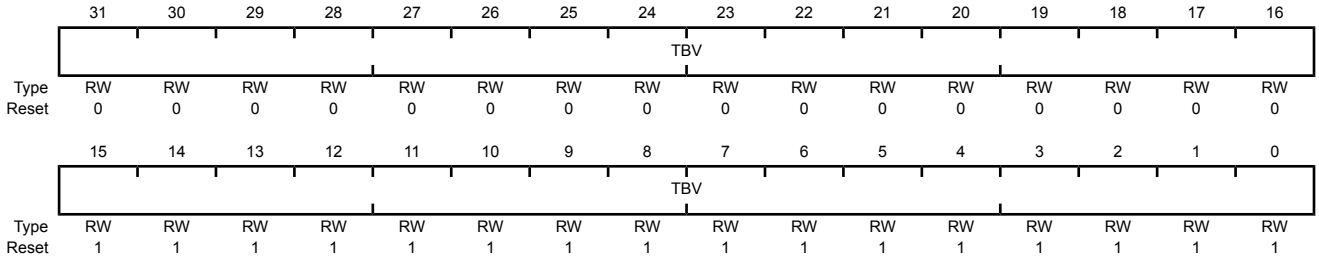
When read, this register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the **GPTMTBR** register on the next clock cycle.

**Note:** When an alternate clock source is enabled, a read of this register returns the current count -1.

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 **GPTMTAV** 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.

GPTM Timer B Value (GPTMTBV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x054
Type RW, reset 0x0000.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TBV | RW | 0x0000.FFFF | GPTM Timer B Value<br>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 **GPTMTAR** register on the next clock cycle.<br>**Note:** In 16-bit mode, only the lower 16-bits of the **GPTMTBV** register can be written with a new value. Writes to the prescaler bits have no effect. |

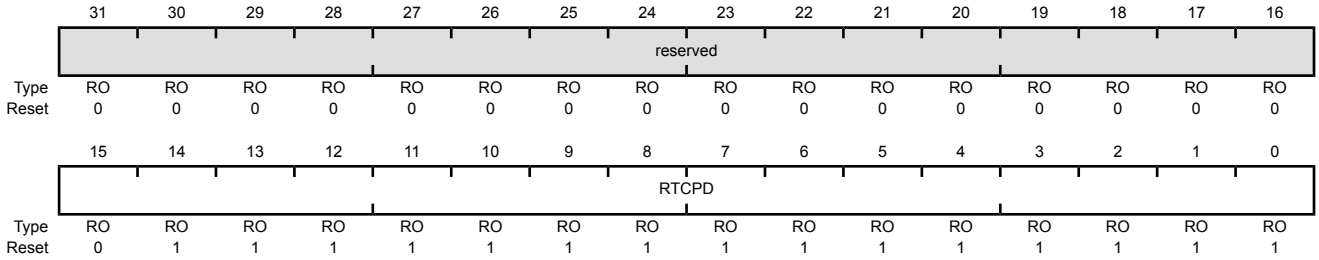
## Register 22: GPTM RTC Predivide (GPTMRTCPD), offset 0x058

This register provides the current RTC predivider value when the timer is operating in RTC mode. Software must perform an atomic access with consecutive reads of the **GPTMTAR**, **GPTMTBR**, and **GPTMRTCPD** registers.

**Note:** When an alternate clock source is enabled, a read of this register returns the current count -1.

GPTM RTC Predivide (GPTMRTCPD)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x058
Type RO, reset 0x0000.7FFF



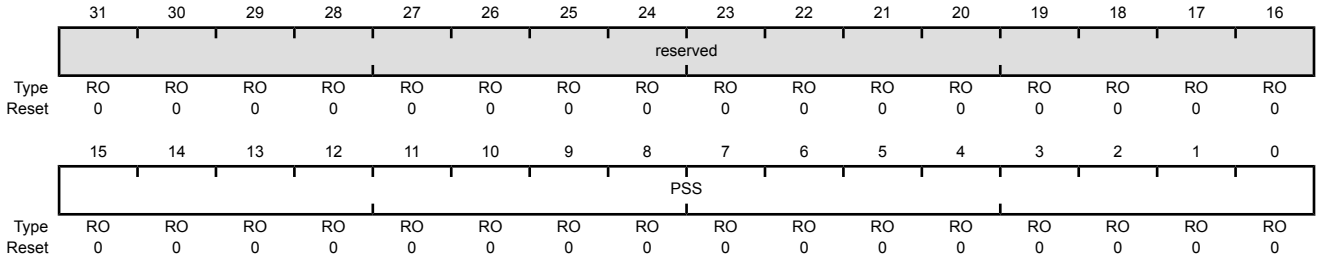

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | RTCPD | RO | 0x0000.7FFF | RTC Predivide Counter Value<br>The current RTC predivider value when the timer is operating in RTC mode. This field has no meaning in other timer modes. |

## Register 23: GPTM Timer A Prescale Snapshot (GPTMTAPS), offset 0x05C

For 16-/32-bit wide GPTM, this register shows the current value of the Timer A prescaler for periodic snapshot mode.

GPTM Timer A Prescale Snapshot (GPTMTAPS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x05C
Type RO, reset 0x0000.0000



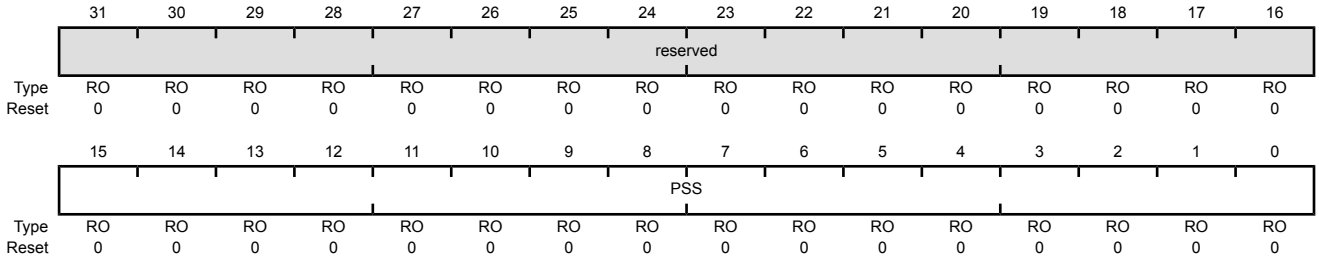

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | PSS | RO | 0x0000 | GPTM Timer A Prescaler Snapshot<br>A read returns the current value of the **GPTM Timer A Prescaler.** |

## Register 24: GPTM Timer B Prescale Snapshot (GPTMTBPS), offset 0x060

For 16-/32-bit wide GPTM, this register shows the current value of the Timer B prescaler for periodic snapshot mode.

GPTM Timer B Prescale Snapshot (GPTMTBPS)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x060
Type RO, reset 0x0000.0000



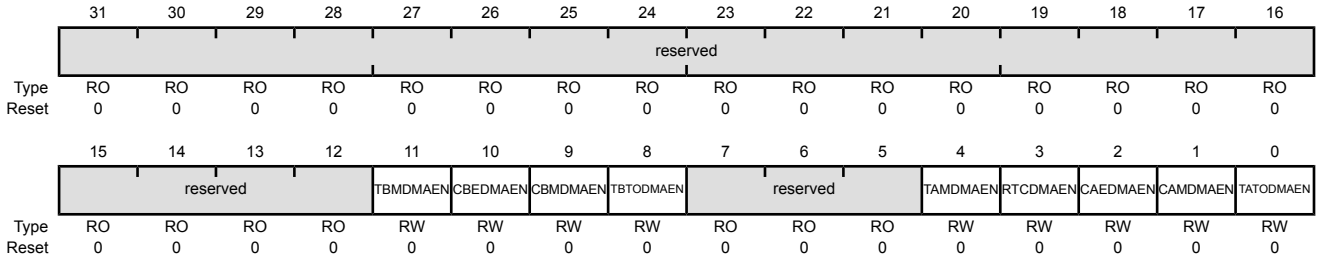

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | PSS | RO | 0x0000 | GPTM Timer A Prescaler Value<br>A read returns the current value of the **GPTM Timer A Prescaler.** |

## Register 25: GPTM DMA Event (GPTMDMAEV), offset 0x06C

This register allows software to enable/disable GPTM DMA trigger events. Setting a bit enables the corresponding DMA trigger, while clearing a bit disables it.

GPTM DMA Event (GPTMDMAEV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x06C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0 | 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. |
| 11 | TBMDMAEN | RW | 0 | GPTM B Mode Match Event DMA Trigger Enable<br>When this bit is enabled, a Timer B dma_req signal is sent to the µDMA when a mode match has occurred.<br>Value Description<br>0 Timer B Mode Match DMA trigger is disabled.<br>1 Timer B DMA Mode Match trigger is enabled. |
| 10 | CBEDMAEN | RW | 0 | GPTM B Capture Event DMA Trigger Enable<br>When this bit is enabled, a Timer B dma_req signal is sent to the µDMA when a capture event has occurred.<br>Value Description<br>0 Timer B Capture Event DMA trigger is disabled.<br>1 Timer B Capture Event DMA trigger is enabled. |
| 9 | CBMDMAEN | RW | 0 | GPTM B Capture Match Event DMA Trigger Enable<br>When this bit is enabled, a Timer B dma_req signal is sent to the µDMA when a capture match event has occurred.<br>Value Description<br>0 Timer B Capture Match DMA trigger is disabled.<br>1 Timer B Capture Match DMA trigger is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TBTODMAEN | RW | 0 | GPTM B Time-Out Event DMA Trigger Enable<br>When this bit is enabled, a Timer B dma_req signal is sent to the µDMA on a time-out event.<br><br>Value Description<br>0 Timer B Time-Out DMA trigger is disabled.<br>1 Timer B Time-Out DMA trigger is enabled. |
| 7:5 | reserved | RO | 0 | 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. |
| 4 | TAMDMAEN | RW | 0 | GPTM A Mode Match Event DMA Trigger Enable<br>When this bit is enabled, a Timer A dma_req signal is sent to the µDMA when a mode match has occurred.<br><br>Value Description<br>0 Timer A Mode Match DMA trigger is disabled.<br>1 Timer A DMA Mode Match trigger is enabled. |
| 3 | RTCDMAEN | RW | 0 | GPTM A RTC Match Event DMA Trigger Enable<br>When this bit is enabled, a Timer A dma_req signal is sent to the µDMA when a RTC match has occurred.<br><br>Value Description<br>0 Timer A RTC Match DMA trigger is disabled.<br>1 Timer A RTC Match DMA trigger is enabled. |
| 2 | CAEDMAEN | RW | 0 | GPTM A Capture Event DMA Trigger Enable<br>When this bit is enabled, a Timer A dma_req signal is sent to the µDMA when a capture event has occurred.<br><br>Value Description<br>0 Timer A Capture Event DMA trigger is disabled.<br>1 Timer A Capture Event DMA trigger is enabled. |
| 1 | CAMDMAEN | RW | 0 | GPTM A Capture Match Event DMA Trigger Enable<br>When this bit is enabled, a Timer A dma_req signal is sent to the µDMA when a capture match event has occurred.<br><br>Value Description<br>0 Timer A Capture Match DMA trigger is disabled.<br>1 Timer A Capture Match DMA trigger is enabled. |

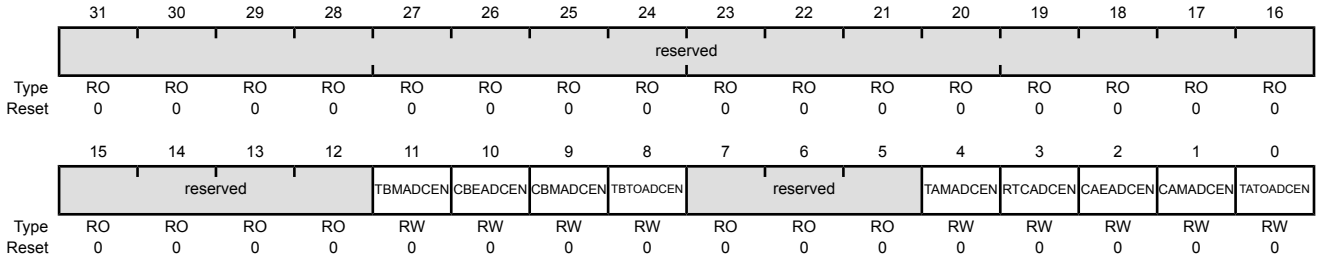
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TATODMAEN | RW | 0 | GPTM A Time-Out Event DMA Trigger Enable<br>When this bit is enabled, a Timer A `dma_req` signal is sent to the µDMA on a time-out event.<br><br>Value Description<br>0 Timer A Time-Out DMA trigger is disabled.<br>1 Timer A Time-Out DMA trigger is enabled. |

## Register 26: GPTM ADC Event (GPTMADCEV), offset 0x070

This register allows software to enable/disable GPTM ADC trigger events. Setting a bit enables the corresponding ADC trigger, while clearing a bit disables it.

GPTM ADC Event (GPTMADCEV)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0x070
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0 | 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. |
| 11 | TBMADCEN | RW | 0 | GPTM B Mode Match Event ADC Trigger Enable<br>When this bit is enabled, a a trigger pulse is sent to the ADC when a mode match has occurred.<br><br>Value Description<br>0 Timer B Mode Match ADC trigger is disabled.<br>1 Timer B Mode Match ADC trigger is enabled. |
| 10 | CBEADCEN | RW | 0 | GPTM B Capture Event ADC Trigger Enable<br>When this bit is enabled, a trigger pulse is sent to the ADC when a capture event has occurred.<br><br>Value Description<br>0 Timer B Capture Event ADC trigger is disabled.<br>1 Timer B Capture Event ADC trigger is enabled. |
| 9 | CBMADCEN | RW | 0 | GPTM B Capture Match Event ADC Trigger Enable<br>When this bit is enabled, a trigger signal is sent to the ADC when a capture match event has occurred.<br><br>Value Description<br>0 Timer B Capture Match ADC trigger is disabled.<br>1 Timer B Capture Match ADC trigger is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TBTOADCEN | RW | 0 | GPTM B Time-Out Event ADC Trigger Enable<br>When this bit is enabled, a trigger signal is sent to the ADC on a time-out event.<br><br>Value Description<br>0 Timer B Time-Out ADC trigger is disabled.<br>1 Timer B Time-Out ADC trigger is enabled. |
| 7:5 | reserved | RO | 0 | 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. |
| 4 | TAMADCEN | RW | 0 | GPTM A Mode Match Event ADC Trigger Enable<br>When this bit is enabled, a a trigger pulse is sent to the ADC when a mode match has occurred.<br><br>Value Description<br>0 Timer A Mode Match ADC trigger is disabled.<br>1 Timer A Mode Match ADC trigger is enabled. |
| 3 | RTCADCEN | RW | 0 | GPTM RTC Match Event ADC Trigger Enable<br>When this bit is enabled, a trigger signal is sent to the ADC when a RTC match has occurred.<br><br>Value Description<br>0 Timer A RTC Match ADC trigger is disabled.<br>1 Timer A RTC Match ADC trigger is enabled. |
| 2 | CAEADCEN | RW | 0 | GPTM A Capture Event ADC Trigger Enable<br>When this bit is enabled, a trigger pulse is sent to the ADC when a capture event has occurred.<br><br>Value Description<br>0 Timer A Capture Event ADC trigger is disabled.<br>1 Timer A Capture Event ADC trigger is enabled. |
| 1 | CAMADCEN | RW | 0 | GPTM A Capture Match Event ADC Trigger Enable<br>When this bit is enabled, a trigger signal is sent to the ADC when a capture match event has occurred.<br><br>Value Description<br>0 Timer A Capture Match ADC trigger is disabled.<br>1 Timer A Capture Match ADC trigger is enabled. |

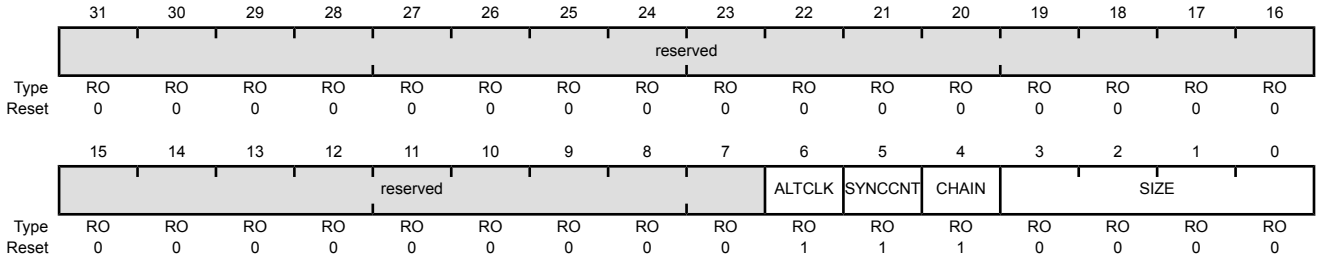
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TATOADCEN | RW | 0 | GPTM A Time-Out Event ADC Trigger Enable<br>When this bit is enabled, a trigger signal is sent to the ADC on a time-out event.<br><br>Value Description<br>0 Timer A Time-Out Event ADC trigger is disabled.<br>1 Timer A Time-Out Event ADC trigger is enabled. |

## Register 27: GPTM Peripheral Properties (GPTMPP), offset 0xFC0

The **GPTMPP** register provides information regarding the properties of the General-Purpose Timer module.

GPTM Peripheral Properties (GPTMPP)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0xFC0
Type RO, reset 0x0000.0070




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0 | 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. |
| 6 | ALTCLK | RO | 0x1 | Alternate Clock Source<br><br>Value Description<br>0 The alternate clock source (ALTCLK) is not available to the Timer module.<br>1 The alternate clock source (ALTCLK) is available to the Timer module. |
| 5 | SYNCCNT | RO | 0x1 | Synchronize Start<br><br>Value Description<br>0 Timer is not capable of synchronizing the counter value with other GPTimers.<br>1 Timer is capable of synchronizing the counter value with other Timers. |
| 4 | CHAIN | RO | 0x1 | Chain with Other Timers<br><br>Value Description<br>0 Timer is not capable of chaining with the previously numbered Timer.<br>1 Timer is capable of chaining with the previously numbered Timer.<br><br>Note that although this bit is set for Timer 0A, this timer cannot chain because there is not a previously numbered Timer. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | SIZE | RO | 0x0 | Count Size |

| Value | Description |
|---|---|
| 0 | Timer A and Timer B counters are 16 bits each with an 8-bit prescale counter. |
| 1 | Timer A and Timer B counters are 32 bits each with a 16-bit prescale counter. |

## Register 28: GPTM Clock Configuration (GPTMCC), offset 0xFC8

The **GPTMCC** register controls the clock source for the General-Purpose Timer module.

**Note:** When the ALTCLK bit is set in the **GPTMCC** register to enable using the alternate clock source, the synchronization imposes restrictions on the starting count value (down-count), terminal value (up-count) and the match value. This restriction applies to all modes of operation. Each event must be spaced by 4 Timer (ALTCLK) clock periods + 2 system clock periods. If some events do not meet this requirement, then it is possible that the timer block may need to be reset for correct functionality to be restored.

Example: ALTCLK= $T_{PIOSC}$ = 62.5ns (16Mhz Trimmed)

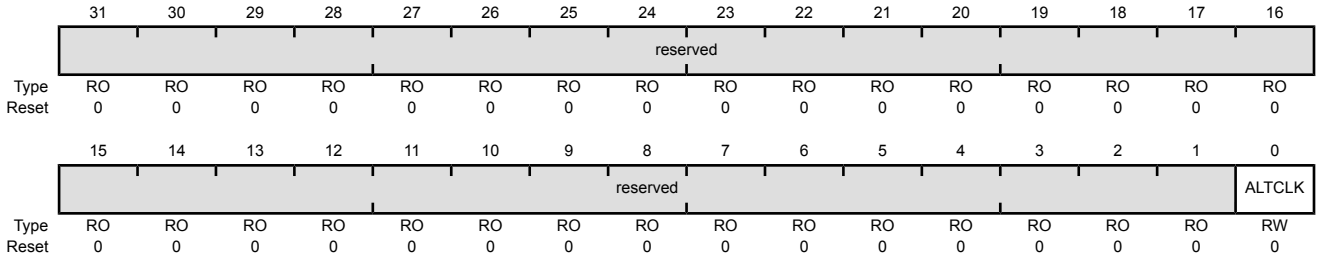
$T_{hclk}$ = 1us (1Mhz)

4*62.5ns + 2*1us = 2.25us 2.25us/62.5ns = 36 or 0x23

The minimum values for the periodic or one-shot with a match interrupt enabled are: **GPTMTAMATCHR** = 0x23 **GPTMTAILR** = 0x46

GPTM Clock Configuration (GPTMCC)

16/32-bit Timer 0 base: 0x4003.0000
16/32-bit Timer 1 base: 0x4003.1000
16/32-bit Timer 2 base: 0x4003.2000
16/32-bit Timer 3 base: 0x4003.3000
16/32-bit Timer 4 base: 0x4003.4000
16/32-bit Timer 5 base: 0x4003.5000
16/32-bit Timer 6 base: 0x400E.0000
16/32-bit Timer 7 base: 0x400E.1000
Offset 0xFC8
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.0000 | 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. |
| 0 | ALTCLK | RW | 0x0 | Alternate Clock Source |

| Value | Description |
|---|---|
| 0 | System clock (based on clock source and divisor factor programmed in **RSCLKCFG** register in the System Control Module) |
| 1 | Alternate clock source as defined by **ALTCLKCFG** register in System Control Module. |

# 17 Watchdog Timers

A watchdog timer can generate a non-maskable interrupt (NMI), a regular interrupt or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way. The TM4C129XKCZAD microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other (Watchdog Timer 1) is clocked by the clock source programmed in the ALTCLK field of the **Alternate Clock Configuration (ALTCLKCFG)** register, System Control offset 0x138. The two modules are identical except that WDT1 is in a different clock domain, and therefore requires synchronizers. As a result, WDT1 has a bit defined in the **Watchdog Timer Control (WDTCTL)** register to indicate when a write to a WDT1 register is complete. Software can use this bit to ensure that the previous access has completed before starting the next access.

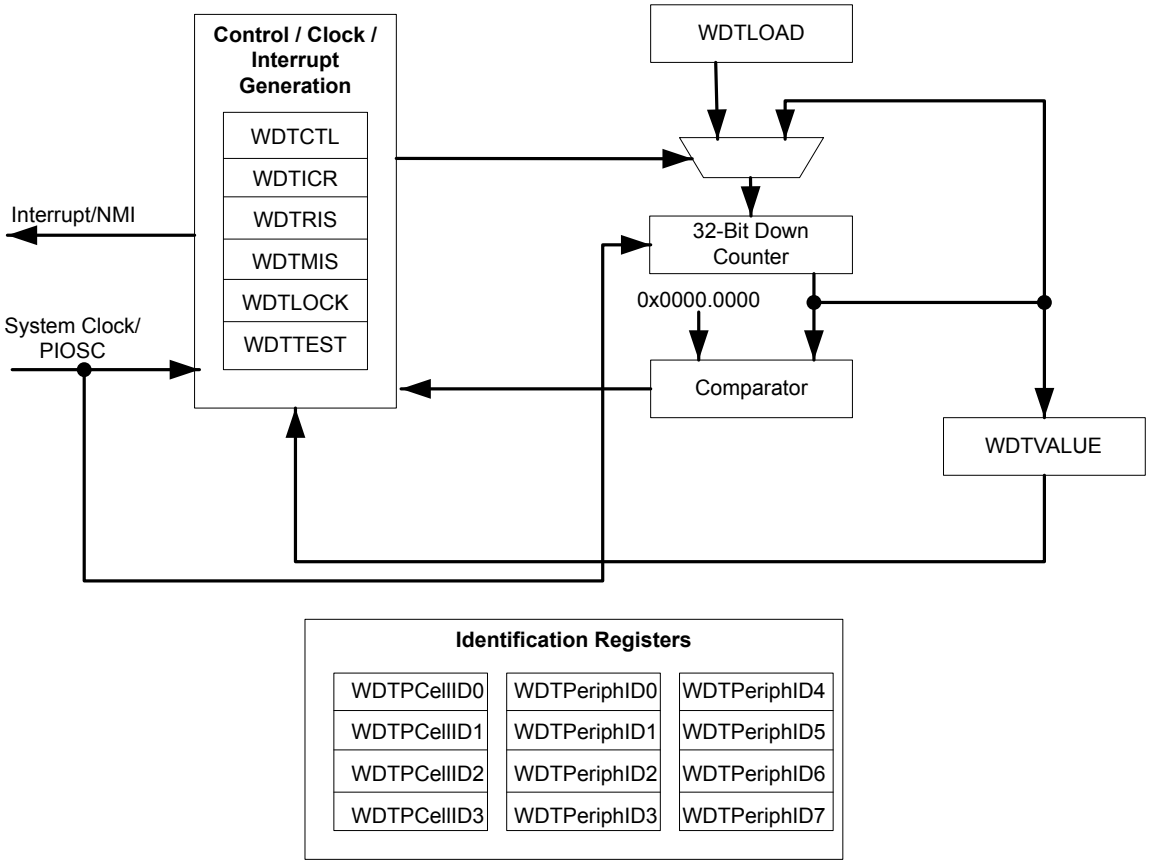
The TM4C129XKCZAD controller has two Watchdog Timer modules with the following features:

- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking and optional NMI function
- Lock register protection from runaway software
- Reset generation logic with an enable/disable
- User-enabled stalling when the microcontroller asserts the CPU Halt flag during debug

The Watchdog Timer 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 Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

## 17.1 Block Diagram

**Figure 17-1. WDT Module Block Diagram**




## 17.2 Functional Description

The Watchdog Timer 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 watchdog timer interrupt. The watchdog interrupt can be programmed to be a non-maskable interrupt (NMI) using the INTTYPE bit in the **WDTCTL** register. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

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 RESEN bit in the **WDTCTL** register, the Watchdog timer 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 **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

The watchdog timer is disabled by default out of reset. To achieve maximum watchdog protection of the device, the watchdog timer can be enabled at the start of the reset vector.

### 17.2.1 Register Access Timing

Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The WRC bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for WRC=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock.

## 17.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the Rn bit in the **Watchdog Timer Run Mode Clock Gating Control (RCGCWD)** register, see page 396.

The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If WDT1, wait for the WRC bit in the **WDTCTL** register to be set.
3. Set the INTEN bit (if interrupts are required) or the RESEN bit (if a reset is required after two timeouts) in the **WDTCTL** register. The Watchdog Timer starts when either of them is enabled.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of 0x1ACC.E551.

To service the watchdog, periodically reload the count value into the **WDTLOAD** register to restart the count. The interrupt can be enabled using the INTEN bit in the **WDTCTL** register to allow the processor to attempt corrective action if the watchdog is not serviced often enough. The RESEN bit in **WDTCTL** can be set so that the system resets if the failure is not recoverable using the ISR.

**Note:** The application should be sure not to modify the ALTCLK encoding in the **ALTCLKCFG** register while the WDT1 is enabled and running.

## 17.4 Register Map

Table 17-1 on page 1181 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address:

- WDT0: 0x4000.0000
- WDT1: 0x4000.1000

Note that the Watchdog Timer module clock must be enabled before the registers can be programmed (see page 396).

**Table 17-1. Watchdog Timers Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | WDTLOAD | RW | 0xFFFF.FFFF | Watchdog Load | 1182 |
| 0x004 | WDTVALUE | RO | 0xFFFF.FFFF | Watchdog Value | 1183 |
| 0x008 | WDTCTL | RW | 0x0000.0000 (WDT0) 0x8000.0000 (WDT1) | Watchdog Control | 1184 |
| 0x00C | WDTICR | WO | - | Watchdog Interrupt Clear | 1186 |
| 0x010 | WDTRIS | RO | 0x0000.0000 | Watchdog Raw Interrupt Status | 1187 |
| 0x014 | WDTMIS | RO | 0x0000.0000 | Watchdog Masked Interrupt Status | 1188 |
| 0x418 | WDTTEST | RW | 0x0000.0000 | Watchdog Test | 1189 |
| 0xC00 | WDTLOCK | RW | 0x0000.0000 | Watchdog Lock | 1190 |
| 0xFD0 | WDTPeriphID4 | RO | 0x0000.0000 | Watchdog Peripheral Identification 4 | 1191 |
| 0xFD4 | WDTPeriphID5 | RO | 0x0000.0000 | Watchdog Peripheral Identification 5 | 1192 |
| 0xFD8 | WDTPeriphID6 | RO | 0x0000.0000 | Watchdog Peripheral Identification 6 | 1193 |
| 0xFDC | WDTPeriphID7 | RO | 0x0000.0000 | Watchdog Peripheral Identification 7 | 1194 |
| 0xFE0 | WDTPeriphID0 | RO | 0x0000.0005 | Watchdog Peripheral Identification 0 | 1195 |
| 0xFE4 | WDTPeriphID1 | RO | 0x0000.0018 | Watchdog Peripheral Identification 1 | 1196 |
| 0xFE8 | WDTPeriphID2 | RO | 0x0000.0018 | Watchdog Peripheral Identification 2 | 1197 |
| 0xFEC | WDTPeriphID3 | RO | 0x0000.0001 | Watchdog Peripheral Identification 3 | 1198 |
| 0xFF0 | WDTPCellID0 | RO | 0x0000.000D | Watchdog PrimeCell Identification 0 | 1199 |
| 0xFF4 | WDTPCellID1 | RO | 0x0000.00F0 | Watchdog PrimeCell Identification 1 | 1200 |
| 0xFF8 | WDTPCellID2 | RO | 0x0000.0006 | Watchdog PrimeCell Identification 2 | 1201 |
| 0xFFC | WDTPCellID3 | RO | 0x0000.00B1 | Watchdog PrimeCell Identification 3 | 1202 |

## 17.5 Register Descriptions

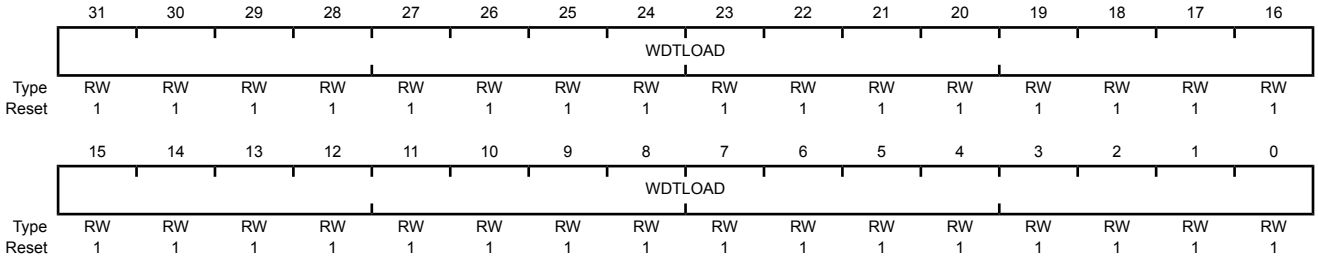
The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

## Register 1: Watchdog Load (WDTLOAD), offset 0x000

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 restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

Watchdog Load (WDTLOAD)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x000
Type RW, reset 0xFFFF.FFFF



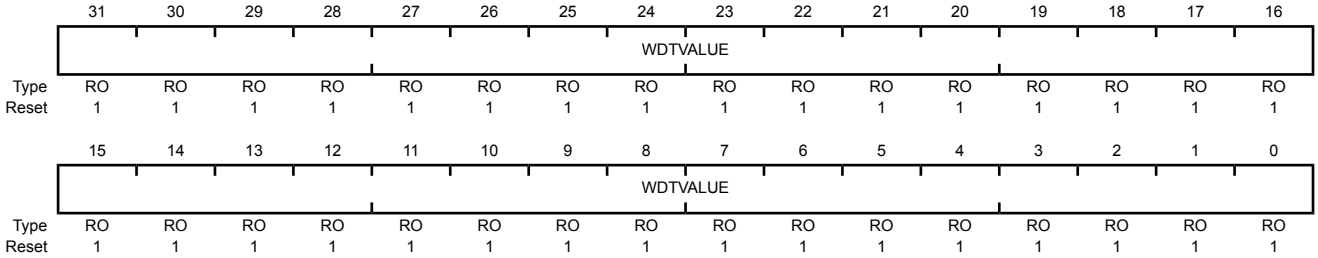

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WDTLOAD | RW | 0xFFFF.FFFF | Watchdog Load Value |

## Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x004
Type RO, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WDTVALUE | RO | 0xFFFF.FFFF | Watchdog Value<br>Current value of the 32-bit down counter. |

## Register 3: Watchdog Control (WDTCTL), offset 0x008

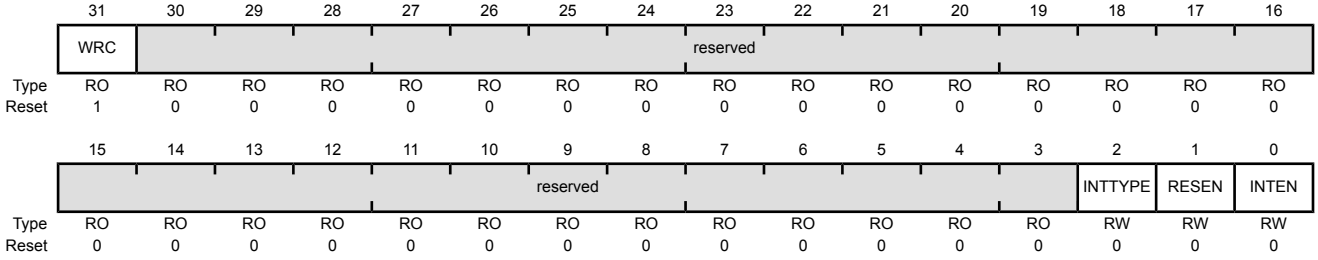
This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled by setting the INTEN bit, all subsequent writes to the INTEN bit are ignored. The only mechanisms that can re-enable writes to this bit are a hardware reset or a software reset initiated by setting the appropriate bit in the **Watchdog Timer Software Reset (SRWD)** register.

**Important:** Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The WRC bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for WRC=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock and therefore does not have a WRC bit.

Watchdog Control (WDTCTL)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x008
Type RW, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1)




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | WRC | RO | 1 | Write Complete<br>The WRC values are defined as follows:<br><br>Value Description<br>0 A write access to one of the WDT1 registers is in progress.<br>1 A write access is not in progress, and WDT1 registers can be read or written.<br><br>**Note:** This bit is reserved for WDT0 and has a reset value of 0. |
| 30:3 | reserved | RO | 0x000.000 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTTYPE | RW | 0 | Watchdog Interrupt Type<br>The INTTYPE values are defined as follows:<br><br>Value Description<br>0 Watchdog interrupt is a standard interrupt.<br>1 Watchdog interrupt is a non-maskable interrupt. |
| 1 | RESEN | RW | 0 | Watchdog Reset Enable<br>The RESEN values are defined as follows:<br><br>Value Description<br>0 Disabled.<br>1 Enable the Watchdog module reset output.<br>Setting this bit enables the Watchdog Timer. |
| 0 | INTEN | RW | 0 | Watchdog Interrupt Enable<br>The INTEN values are defined as follows:<br><br>Value Description<br>0 Interrupt event disabled. Once this bit is set, it can only be cleared by a hardware reset or a software reset initiated by setting the appropriate bit in the **Watchdog Timer Software Reset (SRWD)** register.<br>1 Interrupt event enabled. Once enabled, all writes are ignored.<br>Setting this bit enables the Watchdog Timer. |

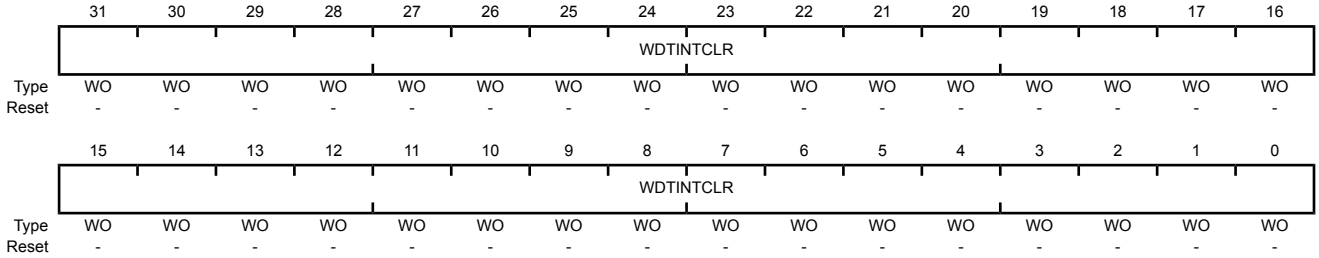
## Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C

This register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Write to this register when a watchdog time-out interrupt has occurred to properly service the Watchdog. Value for a read or reset is indeterminate.

**Note:** Locking the watchdog registers by using the **WDTLOCK** register does not affect the **WDTICR** register and allows interrupts to always be serviced. Thus, a write at any time of the **WDTICR** register clears the **WDTMIS** register and reloads the 32-bit counter from the **WDTLOAD** register. The **WDTICR** register should only be written when interrupts have triggered and need to be serviced.

Watchdog Interrupt Clear (WDTICR)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x00C
Type WO, reset -



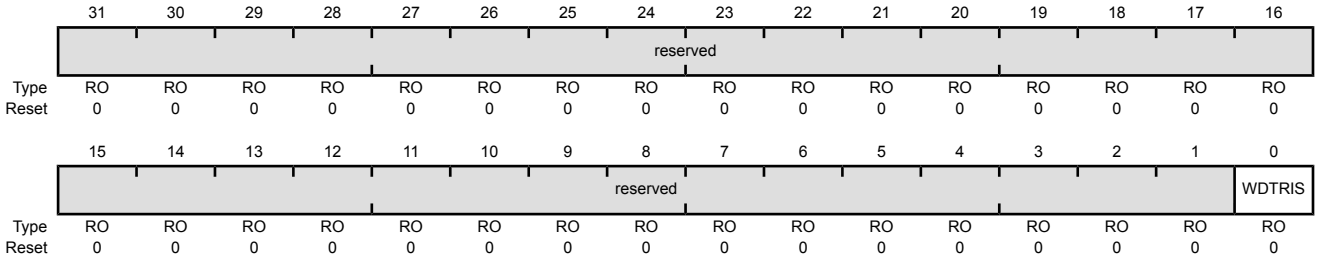

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WDTINTCLR | WO | - | Watchdog Interrupt Clear<br>A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Write to this register when a watchdog time-out interrupt has occurred to properly service the Watchdog. Value for a read or reset is indeterminate. |

## Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

Watchdog Raw Interrupt Status (WDTRIS)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x010
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | WDTRIS | RO | 0 | Watchdog Raw Interrupt Status |

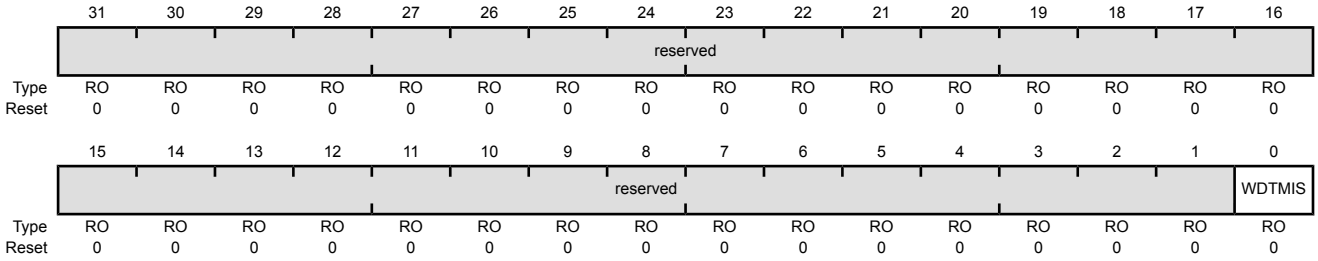
| Value | Description |
|---|---|
| 0 | The watchdog has not timed out. |
| 1 | A watchdog time-out event has occurred. |

## Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

Watchdog Masked Interrupt Status (WDTMIS)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x014
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | WDTMIS | RO | 0 | Watchdog Masked Interrupt Status |

| Value | Description |
|---|---|
| 0 | The watchdog has not timed out or the watchdog timer interrupt is masked. |
| 1 | A watchdog time-out event has been signalled to the interrupt controller. |

## Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Watchdog Test (WDTTEST)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x418
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | 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. |
| 8 | STALL | RW | 0 | Watchdog Stall Enable<br>Value Description<br>0 The watchdog timer continues counting if the microcontroller is stopped with a debugger.<br>1 If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting. |
| 7:0 | reserved | RO | 0x00 | 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. |

## Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers, except for the **Watchdog Test (WDTTEST)** register. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Watchdog Lock (WDTLOCK)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xC00
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WDTLOCK | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WDTLOCK | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WDTLOCK | RW | 0x0000.0000 | Watchdog Lock<br>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, except for the **WDTTEST** register. Avoid writes to the **WDTTEST** register when the watchdog registers are locked.<br>A read of this register returns the following values: |

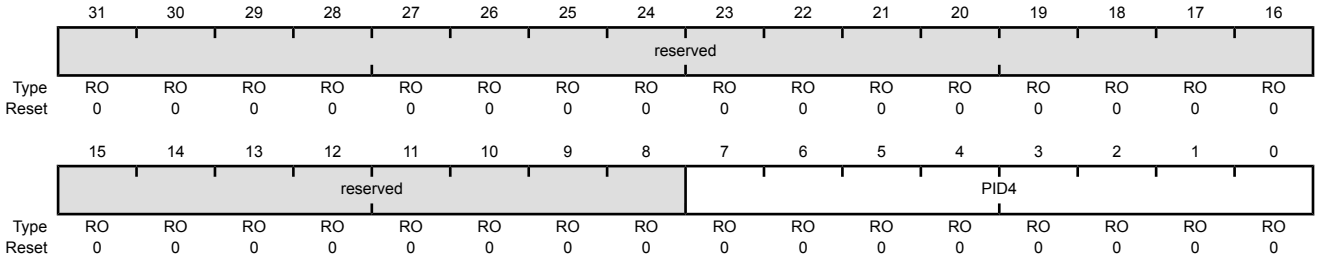
| Value | Description |
|---|---|
| 0x0000.0001 | Locked |
| 0x0000.0000 | Unlocked |

## Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 4 (WDTPeriphID4)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD0
Type RO, reset 0x0000.0000



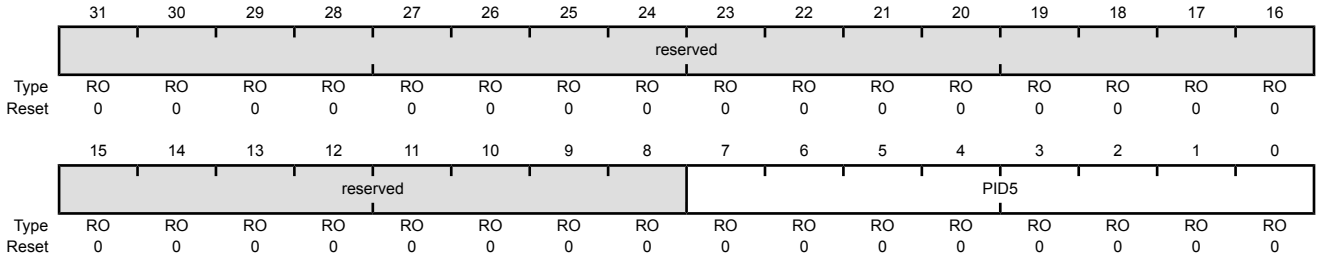

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | WDT Peripheral ID Register [7:0] |

## Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 5 (WDTPeriphID5)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD4
Type RO, reset 0x0000.0000



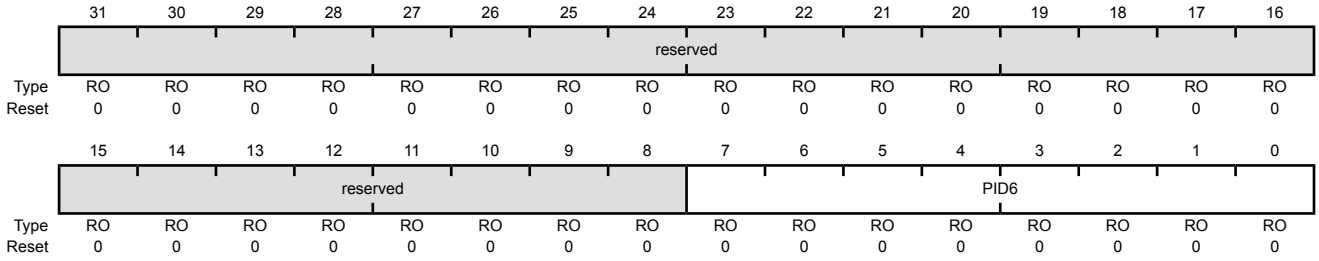

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | WDT Peripheral ID Register [15:8] |

## Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 6 (WDTPeriphID6)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD8
Type RO, reset 0x0000.0000



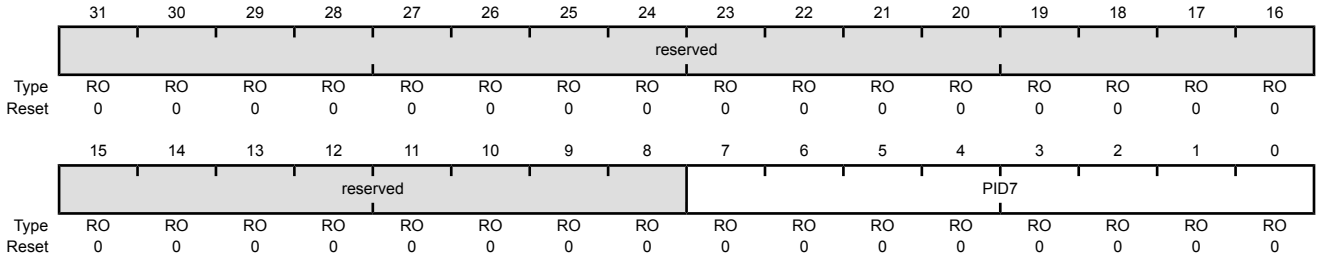

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | WDT Peripheral ID Register [23:16] |

## Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 7 (WDTPeriphID7)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFDC
Type RO, reset 0x0000.0000



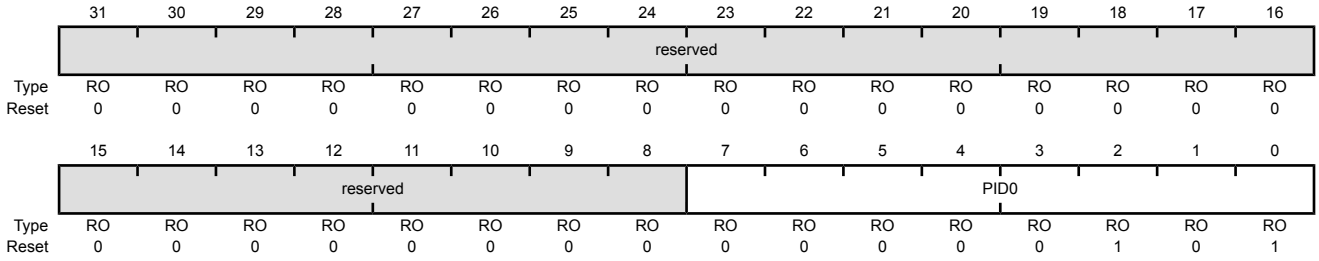

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | WDT Peripheral ID Register [31:24] |

## Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 0 (WDTPeriphID0)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE0
Type RO, reset 0x0000.0005



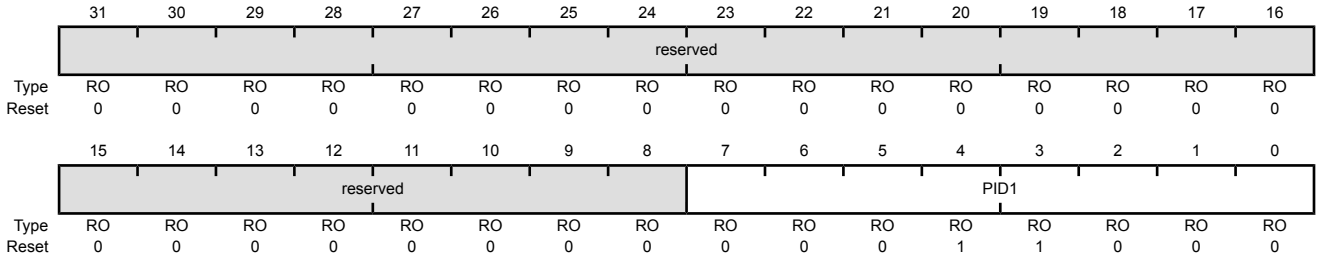

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x05 | Watchdog Peripheral ID Register [7:0] |

## Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 1 (WDTPeriphID1)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE4
Type RO, reset 0x0000.0018



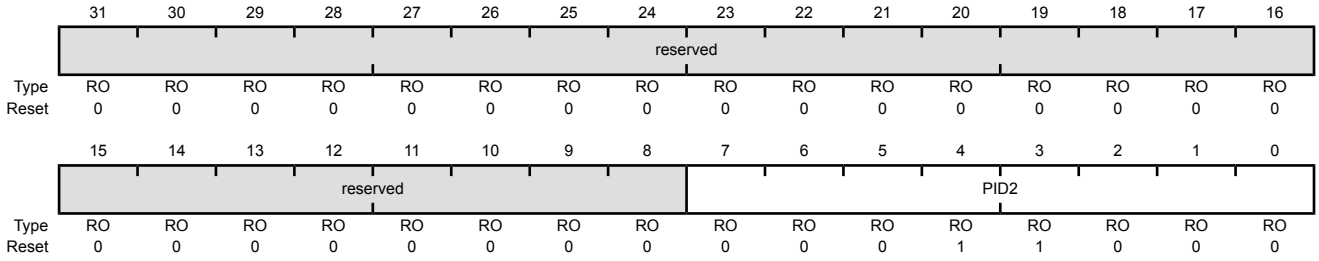

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x18 | Watchdog Peripheral ID Register [15:8] |

## Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 2 (WDTPeriphID2)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE8
Type RO, reset 0x0000.0018



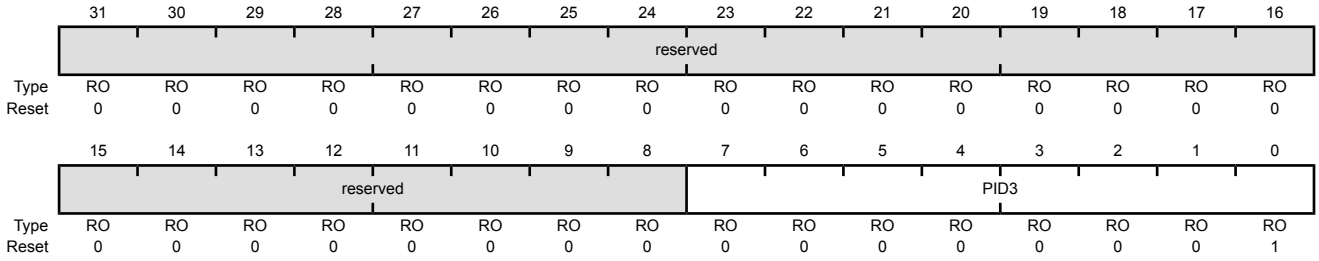

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | Watchdog Peripheral ID Register [23:16] |

## Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 3 (WDTPeriphID3)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFEC
Type RO, reset 0x0000.0001



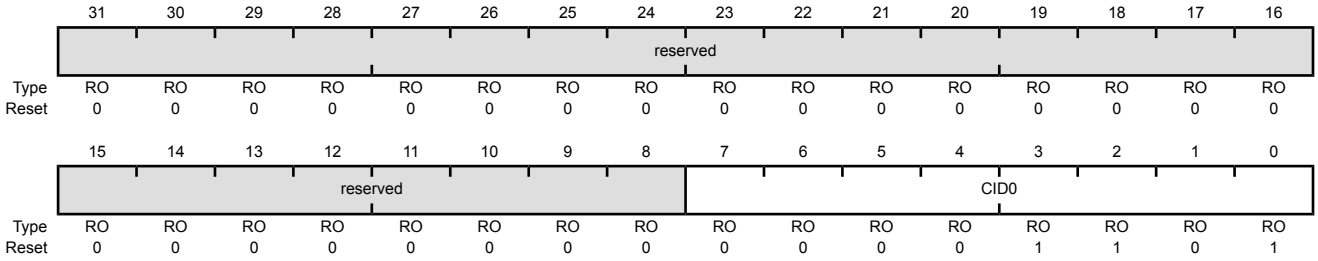

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | Watchdog Peripheral ID Register [31:24] |

## Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 0 (WDTPCellID0)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF0
Type RO, reset 0x0000.000D



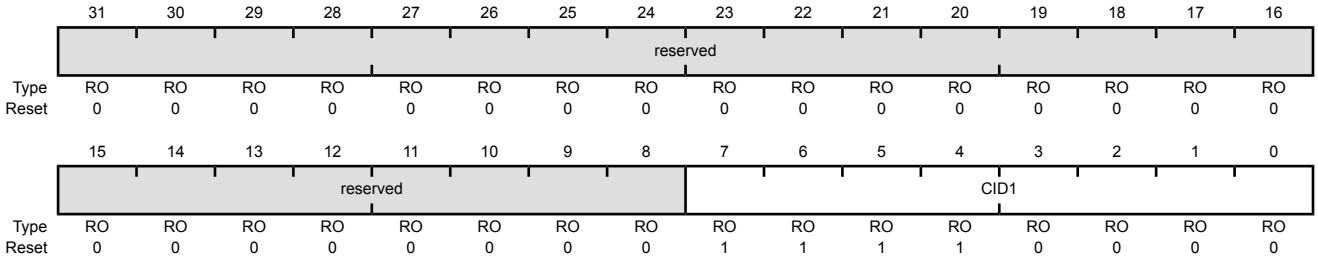

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | Watchdog PrimeCell ID Register [7:0] |

## Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 1 (WDTPCellID1)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF4
Type RO, reset 0x0000.00F0



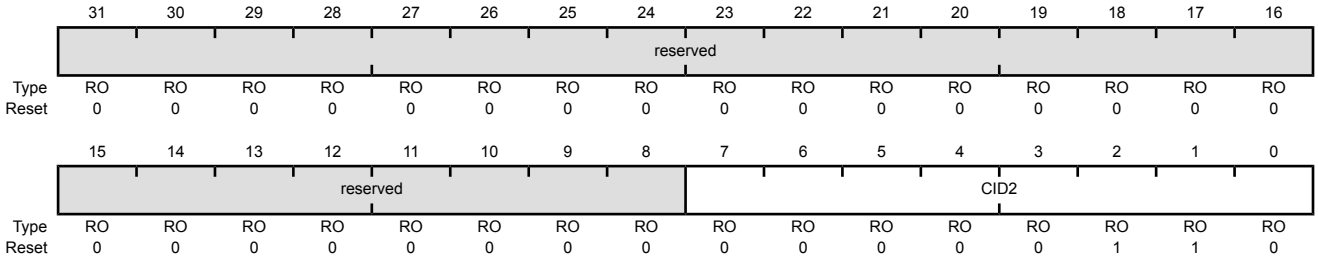

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | Watchdog PrimeCell ID Register [15:8] |

## Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 2 (WDTPCellID2)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF8
Type RO, reset 0x0000.0006



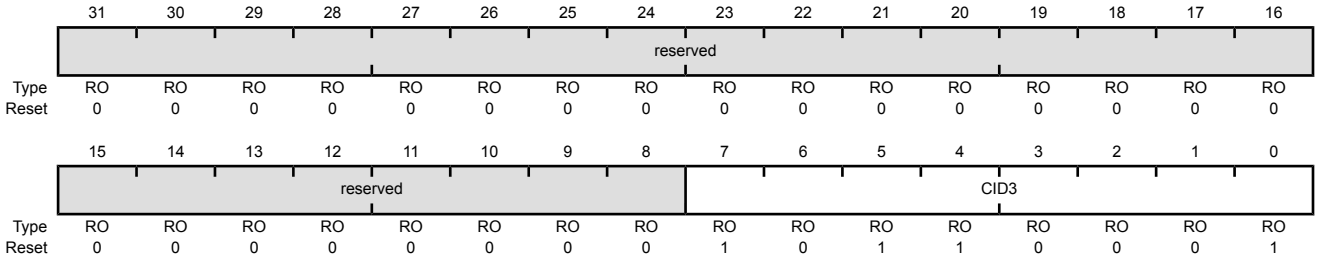

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x06 | Watchdog PrimeCell ID Register [23:16] |

## Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3 ), offset 0xFFC

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 3 (WDTPCellID3)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFFC
Type RO, reset 0x0000.00B1




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | CID3 | RO | 0xB1 | Watchdog PrimeCell ID Register [31:24] |

# 18 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. Two identical converter modules are included, which share 24 input channels.

The TM4C129XKCZAD ADC module features 12-bit conversion resolution and supports 24 input channels, plus an internal temperature sensor. Each ADC module contains four programmable sequencers allowing the sampling of multiple analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. In addition, the conversion value can optionally be diverted to a digital comparator module. Each ADC module provides eight digital comparators. Each digital comparator evaluates the ADC conversion value against its two user-defined values to determine the operational range of the signal. The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. A phase shifter can delay the start of sampling by a specified phase angle. When using both ADC modules, it is possible to configure the converters to start the conversions coincidentally or within a relative phase from each other, see “Sample Phase Control” on page 1210.

The TM4C129XKCZAD microcontroller provides two ADC modules with each having the following features:

- 24 shared analog input channels
- 12-bit precision ADC
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of two million samples/second
- Optional, programmable phase delay
- Sample and hold window programmability
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - PWM
  - GPIO
- Hardware averaging of up to 64 samples
- Eight digital comparators

- Converter uses two external reference signals (VREFA+ and VREFA-) or VDDA and GNDA as the voltage reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each sample sequencer
  - ADC module uses burst requests for DMA
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate ADC clock

## 18.1 Block Diagram

The TM4C129XKCZAD microcontroller contains two identical Analog-to-Digital Converter modules. These two modules, ADC0 and ADC1, share the same 24 analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers. Figure 18-1 on page 1204 shows how the two modules are connected to analog inputs and the system bus.

**Figure 18-1. Implementation of Two ADC Blocks**

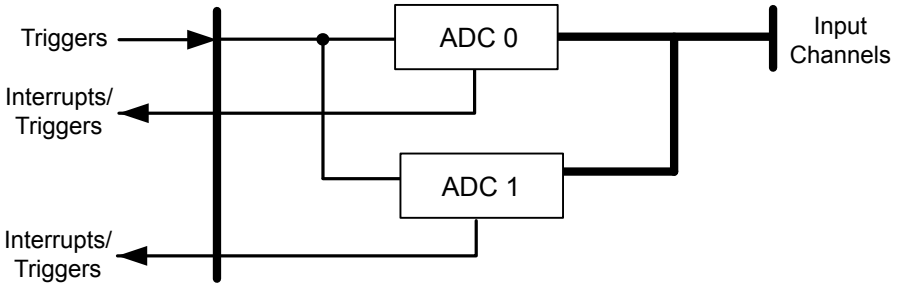
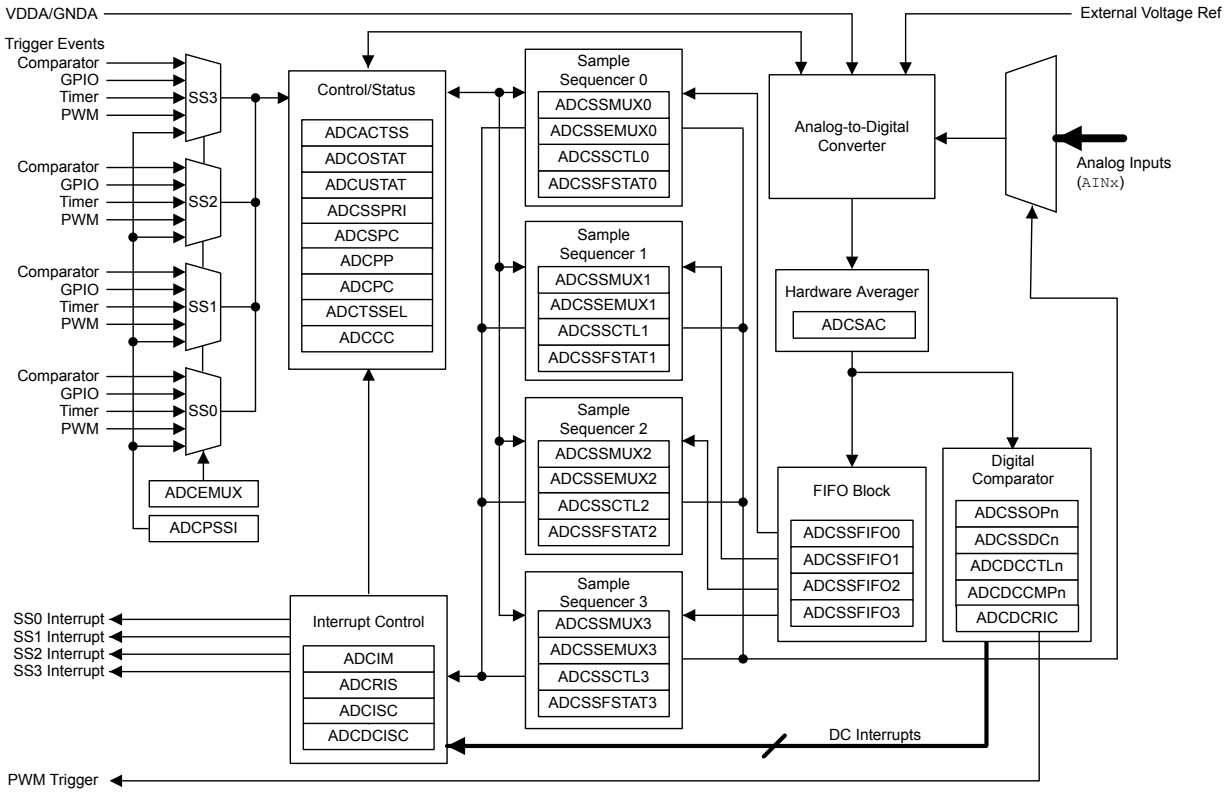



Figure 18-2 on page 1205 provides details on the internal configuration of the ADC controls and data registers.

**Figure 18-2. ADC Module Block Diagram**




## 18.2 Signal Description

The following table lists the external signals of the ADC module and describes the function of each. The AINx signals are analog functions for some GPIO signals. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the ADC signals. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding AMSEL bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772. The VREFA+ and VREFA- signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

**Table 18-1. ADC Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| AIN0 | G2 | PE3 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | G1 | PE2 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | H2 | PE1 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | H3 | PE0 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | B2 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | B3 | PD6 | I | Analog | Analog-to-digital converter input 5. |
| AIN6 | B4 | PD5 | I | Analog | Analog-to-digital converter input 6. |
| AIN7 | A4 | PD4 | I | Analog | Analog-to-digital converter input 7. |

**Table 18-1. ADC Signals (212BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| AIN8 | B5 | PE5 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | A5 | PE4 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | C6 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | B6 | PB5 | I | Analog | Analog-to-digital converter input 11. |
| AIN12 | D1 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | D2 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | C1 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | C2 | PD0 | I | Analog | Analog-to-digital converter input 15. |
| AIN16 | J1 | PK0 | I | Analog | Analog-to-digital converter input 16. |
| AIN17 | J2 | PK1 | I | Analog | Analog-to-digital converter input 17. |
| AIN18 | K1 | PK2 | I | Analog | Analog-to-digital converter input 18. |
| AIN19 | K2 | PK3 | I | Analog | Analog-to-digital converter input 19. |
| AIN20 | A7 | PE6 | I | Analog | Analog-to-digital converter input 20. |
| AIN21 | B7 | PE7 | I | Analog | Analog-to-digital converter input 21. |
| AIN22 | A8 | PP7 | I | Analog | Analog-to-digital converter input 22. |
| AIN23 | B8 | PP6 | I | Analog | Analog-to-digital converter input 23. |
| VREFA+ | F4 | fixed | - | Analog | A reference voltage used to specify the voltage at which the ADC converts to a maximum value. This pin is used in conjunction with VREFA-, which specifies the minimum value . The voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA+ voltage is limited to the range specified in Table 32-44 on page 2138. |
| VREFA- | G5 | fixed | - | Analog | A reference voltage used to specify the input voltage at which the ADC converts to a minimum value. This pin is used in conjunction with VREFA+, which specifies the maximum value. In other words, the voltage that is applied to VREFA- is the voltage with which an AINn signal is converted to 0, while the voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA- voltage is limited to the range specified in Table 32-44 on page 2138. |

## 18.3 Functional Description

The TM4C129XKCZAD ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence. In addition, the µDMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

### 18.3.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 18-2 on page 1207 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. Each sample that is captured is stored in the FIFO. In this implementation, each FIFO entry is a 32-bit word, with the lower 12 bits containing the conversion result.

**Table 18-2. Samples and FIFO Depth of Sequencers**

| Sequencer | Number of Samples | Depth of FIFO |
|---|---|---|
| SS3 | 1 | 1 |
| SS2 | 4 | 4 |
| SS1 | 4 | 4 |
| SS0 | 8 | 8 |

For a given sample sequence, each sample is defined by bit fields in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)**, **ADC Sample Sequence Extended Input Multiplexer Select (ADCSSEMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** and **ADCSSEMUXn** fields select the input pin, while the **ADCSSCTLn** fields contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective ASENn bit in the **ADC Active Sample Sequencer (ADCACTSS)** register and should be configured before being enabled. Sampling is then initiated by setting the SSn bit in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. In addition, sample sequences may be initiated on multiple ADC modules simultaneously using the GSYNC and SYNCWAIT bits in the **ADCPSSI** register during the configuration of each ADC module. For more information on using these bits, refer to page 1254.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence are allowed. In the **ADCSSCTLn** register, the IEn bits can be set for any combination of samples, allowing interrupts to be generated after every sample in the sequence if necessary. Also, the END bit can be set at any point within a sample sequence. For example, if Sequencer 0 is used, the END bit can be set in the nibble associated with the fifth sample, allowing Sequencer 0 to complete execution of the sample sequence after the fifth sample.

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFOn)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with FULL and EMPTY status flags. If a write is attempted when the FIFO is full, the write does not occur and an overflow condition is indicated. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

### 18.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- DMA operation

- Sequence prioritization
- Trigger configuration
- Comparator configuration
- External voltage reference
- Sample phase control
- Module clocking

#### 18.3.2.1 Interrupts

The register configurations of the sample sequencers and digital comparators dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the MASK bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of the various interrupt signals; and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding IN bit in **ADCISC**. Digital comparator interrupts are cleared by writing a 1 to the **ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)** register.

#### 18.3.2.2 DMA Operation

DMA may be used to increase efficiency by allowing each sample sequencer to operate independently and transfer data without processor intervention or reconfiguration.

The ADC asserts single and burst µDMA request signals (dma_sreq and dma_req) to the µDMA controller based on the FIFO level. The dma_req signal is generated when the FIFO in question is half-full (that is, at 4 samples for SS0, 2 samples for SS1 and SS2, and at 1 sample for SS3). If, for example, the **ADCSSCTL0** register has six samples to transfer, a burst of four values occurs followed by two single transfers (dma_sreq). The dma_done signals (one per sample sequencer) are sent to the ADC to allow for a triggering of DMAINRn interrupt bits in the **ADCRIS** register. The µDMA is enabled for a specific sample sequencer by setting the appropriate ADENn bit in the **ADCACTSS** register at offset 0x000.

To use the µDMA with the ADC module, the application must enable the ADC channel through **DMA Channel Map Select n (DMACHMAPn)** register in the µDMA.

Refer to the “Micro Direct Memory Access (µDMA)” on page 708 for more details about programming the µDMA controller.

#### 18.3.2.3 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the **ADC Sample Sequencer Priority (ADCSSPRI)** register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active sample sequencer units with the same priority do not provide consistent results, so software must ensure that all active sample sequencer units have a unique priority value.

#### 18.3.2.4 Sampling Events

Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), analog comparators, an external signal on a GPIO specified by the **GPIO ADC Control (GPIOADCCTL)** register, a GP Timer, a

PWM generator, and continuous sampling. The processor triggers sampling by setting the SSx bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

Care must be taken when using the continuous sampling trigger. If a sequencer's priority is too high, it is possible to starve other lower priority sequencers. Generally, a sample sequencer using continuous sampling should be set to the lowest priority. Continuous sampling can be used with a digital comparator to cause an interrupt when a particular voltage is seen on an input.

#### 18.3.2.5 Sample and Hold Window Control

The ADC module provides the capability of programming the sample and hold window of each step in a sequence through the **ADC Sample Sequence n Sample and Hold Time (ADCSSTSHn)** register. Each TSHn field can be written with a different sample and hold width, which is represented in ADC clocks. The table below gives the allowed encodings:

**Table 18-3. Sample and Hold Width in ADC Clocks**

| TSHn Encoding | $N_{SH}$ |
|---|---|
| 0x0 | 4 |
| 0x1 | reserved |
| 0x2 | 8 |
| 0x3 | reserved |
| 0x4 | 16 |
| 0x5 | reserved |
| 0x6 | 32 |
| 0x7 | reserved |
| 0x8 | 64 |
| 0x9 | reserved |
| 0xA | 128 |
| 0xB | reserved |
| 0xC | 256 |
| 0xD-0xF | reserved |

The ADC conversion frequency is a function of the Sample and Hold number, given by the following equation:

$F_{CONV} = 1/((N_{SH} + 12)*T_{ADC})$

where:

- $N_{SH}$ is the sample and hold width in ADC clocks
- $T_{ADC}$ is the ADC conversion clock period, which is the inverse of the ADC clock frequency $F_{ADC}$

Now, the maximum allowable external source resistance ($R_S$) also changes with the value of $N_{SH}$, as the total settling time of the input circuitry must be fast enough to settle to within the ADC resolution in a single sampling interval. The input circuitry includes the external source resistance as well as the input resistance and capacitance of the ADC ($R_{ADC}$ and $C_{ADC}$).

The values for $R_S$ and $F_{CONV}$ for varying $N_{SH}$ values, with $F_{ADC}$=16MHz and $F_{ADC}$=32MHz are given in tables 18-4-a and 18-4-b. The system designer must take into consideration both of these factors for optimal ADC operation.

**Table 18-4. $R_S$ and $F_{CONV}$ Values with Varying $N_{SH}$ Values and $F_{ADC}$ = 16 MHz**

| $N_{SH}$ (Cycles) | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|
| $F_{CONV}$ (Ksps) | 1000 | 800 | 571 | 364 | 211 | 114 | 60 |
| $R_S$ Max (Ω) | 500 | 3500 | 9500 | 21500 | 45500 | 93500 | 189500 |

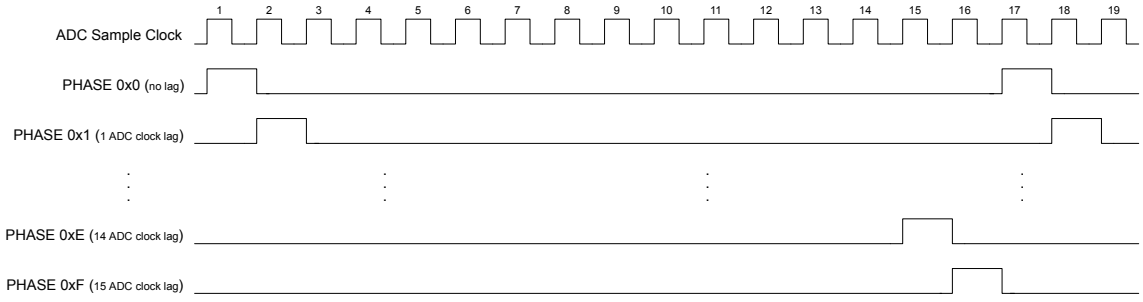
**Table 18-5. $R_S$ and $F_{CONV}$ Values with Varying $N_{SH}$ Values and $F_{ADC}$ = 32 MHz**

| $N_{SH}$ (Cycles) | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|
| $F_{CONV}$ (Ksps) | 2000 | 1600 | 1143 | 727 | 421 | 229 | 119 |
| $R_S$ Max (Ω) | 250 | 500 | 3500 | 9500 | 21500 | 45500 | 93500 |

### 18.3.2.6 Sample Phase Control

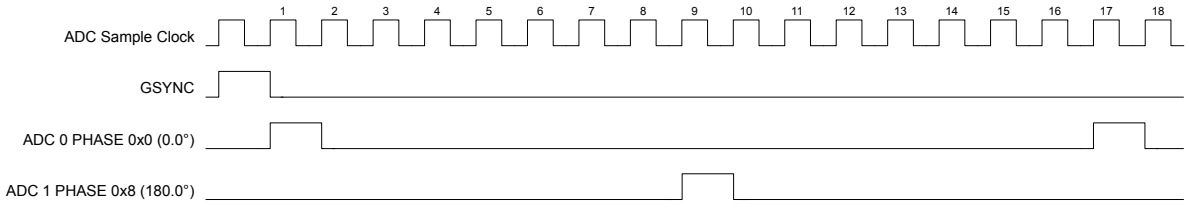
The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. If the converters are running at the same sample rate, they may be configured to start the conversions coincidentally or one ADC may be programmed to lag up to 15 clock cycles relative to the other ADC. The sample time can be delayed from the standard sampling time by programming the PHASE field in the **ADC Sample Phase Control (ADCSPC)** register. Figure 18-3 on page 1210 shows an example of various phase relationships.

**Figure 18-3. ADC Sample Phases**




This feature can be used to double the sampling rate of an input. Both ADC Module 0 and ADC Module 1 can be programmed to sample the same input. ADC module 0 can sample at the standard position (the PHASE field in the **ADCSPC** register is 0x0). ADC Module 1 can be configured to sample with a phase lag (PHASE is nonzero). For a sample rate of two million samples/second at 16MHz, the TSHn field of all of the sequencer samples of both ADCs must be programmed to 0x0 and the PHASE field of one of the ADC modules must be set to 0x8. The two modules can be be synchronized using the GSYNC and SYNCWAIT bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. Software can then combine the results from the two modules to create a sample rate of two million samples/second at 16MHz as shown in Figure 18-4 on page 1211.

**Figure 18-4. Doubling the ADC Sample Rate**




Using the **ADCSPC** register, ADC0 and ADC1 may provide a number of interesting applications:

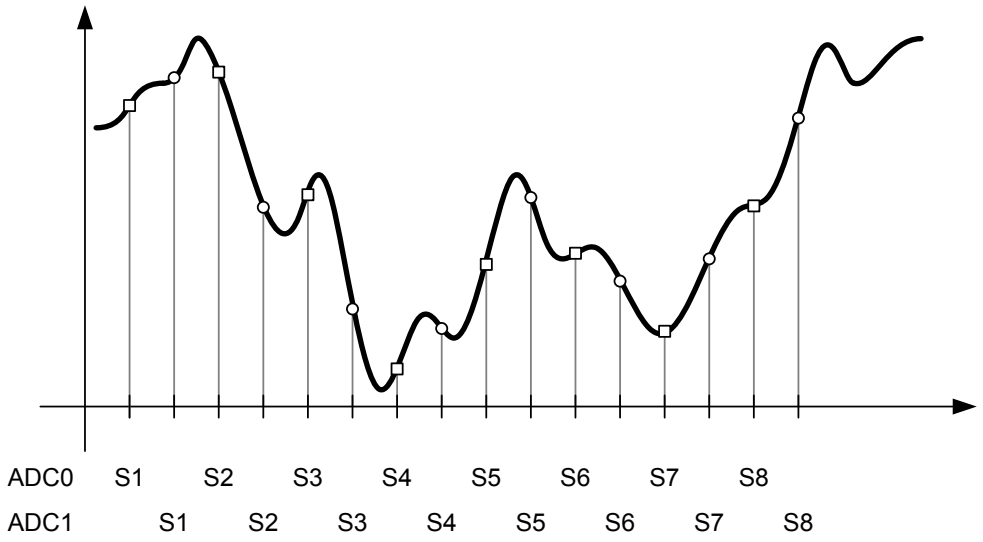
- Coincident continuous sampling of different signals. The sample sequence steps run coincidently in both converters. In this situation, the TSHn of matching sample steps of both ADC module sequencers must be the same and the PHASE field must be 0x0 in both ADC module **ADCSPC** registers. The TSHn field is found in the **ADC Sample Sequence n Sample and Hold Time (ADCSSTSHn)** register.
  - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
  - ADC Module 1, **ADCSPC** = 0x0, sampling AIN1

  **Note:** If two ADCs are configured to sample the same signal, a skew (phase lag) must be added to one of the ADC modules to prevent coincident sampling. Phase lag can be added by programming the PHASE field in the **ADCSPC** register.

- Skewed sampling of the same signal. The skew is determined by both the TSHn field in the **ADCSSTSHn** registers and the PHASE field in the **ADCSPC** register. For the fastest skewed sample rate, all TSHn fields must be programmed to 0x0. If TSHn=0x0 for all sequencers and the PHASE field of one ADC is 0x8, the configuration doubles the conversion bandwidth of a single input when software combines the results as shown in Figure 18-5 on page 1212.
  - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
  - ADC Module 1, **ADCSPC** = 0x8, sampling AIN0

  Note that it is not required that the TSHn fields be the same in a skewed sample. If an application has varying analog input resistance, then TSHn and PHASE may vary according to operational requirements.

**Figure 18-5. Skewed Sampling**




### 18.3.2.7 Module Clocking

The ADC digital block is clocked by the system clock and the ADC analog block is clocked from a separate conversion clock (ADC Clock). The ADC clock frequency can be up to 32 MHz to generate a conversion rate of 2 Msps. A 16 MHz ADC clock provides a 1 Msps sampling rate. There are three sources of the ADC clock:

- Divided PLL VCO. The PLL VCO frequency can be configured to generate up to a 32-MHz clock for a conversion rate of 2 Msps. The CS field in the **ADCCC** register must be programmed to 0x0 to select the PLL VCO and the CLKDIV field is used to set the appropriate clock divisor for the desired frequency.
- 16 MHz PIOSC. Using the PIOSC provides a conversion rate near 1 Msps. To use the PIOSC to clock the ADC, first power up the PLL and then enable the PIOSC in the CS bit field in the **ADCCC** register, then disable the PLL.
- MOSC. The MOSC clock source must be 16 MHz for a 1 Msps conversion rate and 32 MHz for a 2 Msps conversion rate.

The system clock must be at the same frequency or higher than the ADC clock. All ADC modules share the same clock source to facilitate the synchronization of data samples between conversion units, the selection and programming of which is provided by ADC0's **ADCCC** register. The ADC modules do not run at different conversion rates.

### 18.3.2.8 Busy Status

The BUSY bit of the **ADCACTSS** register is used to indicate when the ADC is busy with a current conversion. When there are no triggers pending which may start a new conversion in the immediate cycle or next few cycles, the BUSY bit reads as 0. Software must read the status of the BUSY bit as clear before disabling the ADC clock by writing to the **Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)** register.

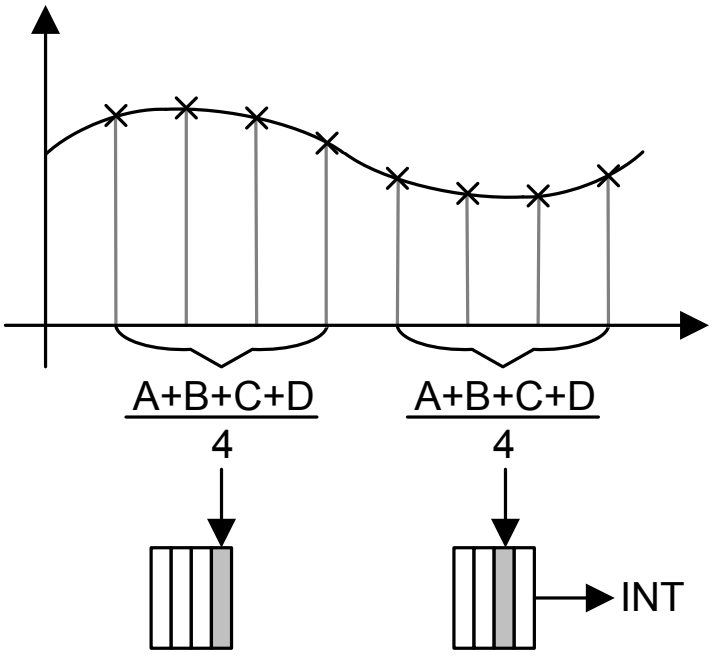
### 18.3.3 Hardware Sample Averaging Circuit

Higher precision results can be generated using the hardware averaging circuit, however, the improved results are at the cost of throughput. Up to 64 samples can be accumulated and averaged to form a single data entry in the sequencer FIFO. Throughput is decreased proportionally to the number of samples in the averaging calculation. For example, if the averaging circuit is configured to average 16 samples, the throughput is decreased by a factor of 16.

By default the averaging circuit is off, and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 1256). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

Figure 18-6 shows an example in which the **ADCSAC** register is set to 0x2 for 4x hardware oversampling and the `IE1` bit is set for the sample sequence, resulting in an interrupt after the second averaged value is stored in the FIFO.

**Figure 18-6. Sample Averaging Example**

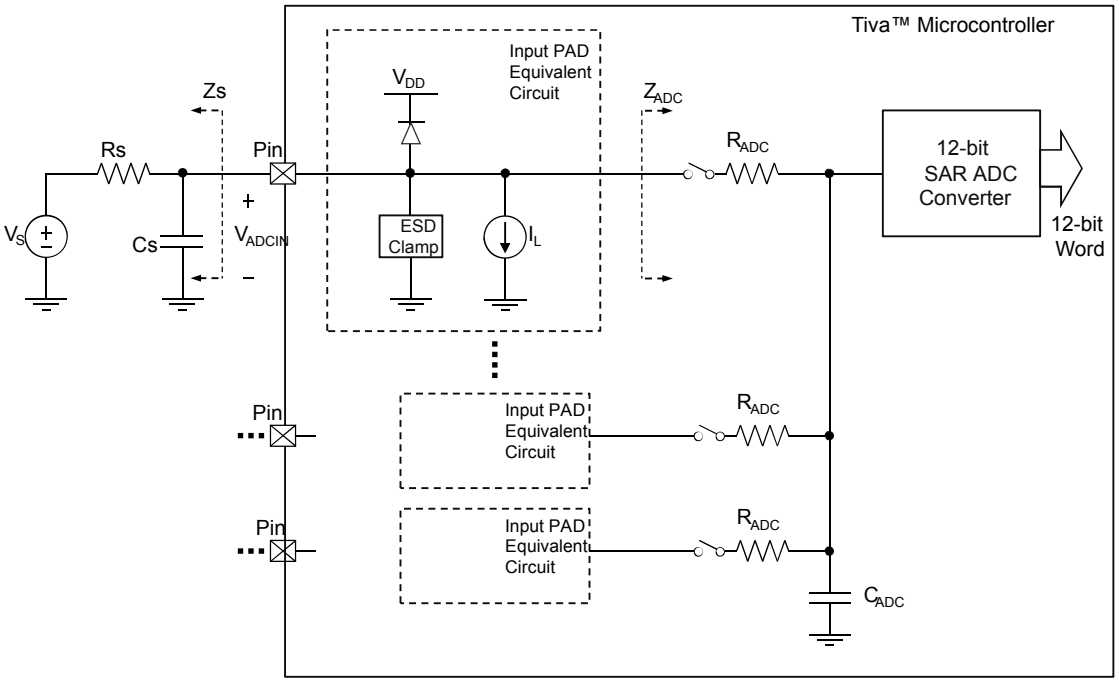



### 18.3.4 Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 12-bit, low-power, high-precision conversion value. The successive approximation uses a switched capacitor array to perform the dual functions of sampling and holding the signal as well as providing the 12-bit DAC operation.

Figure 18-7 shows the ADC input equivalency diagram; for parameter values, see “Analog-to-Digital Converter (ADC)” on page 2138.

**Figure 18-7. ADC Input Equivalency**

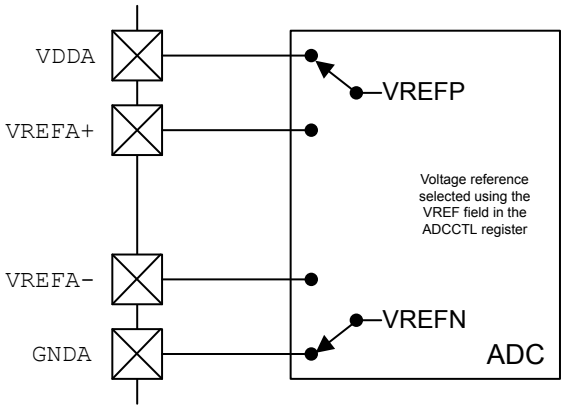



The ADC operates from both the 3.3-V analog and 1.2-V digital power supplies. The ADC clock can be configured to reduce power consumption when ADC conversions are not required (see "System Control" on page 251). The analog inputs are connected to the ADC through specially balanced input paths to minimize the distortion and cross-talk on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in "Analog-to-Digital Converter (ADC)" on page 2138.

### 18.3.4.1 Voltage Reference

The ADC uses internal signals VREFP and VREFN as references to produce a conversion value from the selected analog input. VREFP can be connected to either VREFA+ or VDDA and VREFN can be connected to either VREFA- or GNDA as configured by the VREF bit in the **ADC Control (ADCCTL)** register, as shown in Figure 18-8.

**Figure 18-8. ADC Voltage Reference**

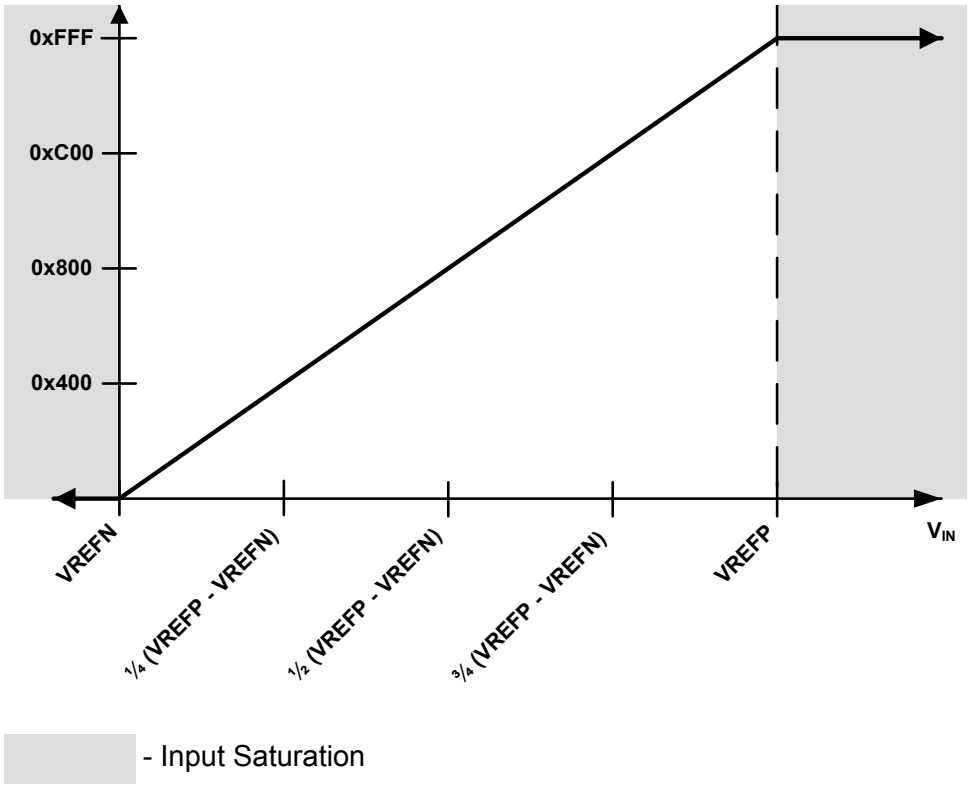


The range of this conversion value is from 0x000 to 0xFFF. In single-ended-input mode, the 0x000 value corresponds to the voltage level on VREFN; the 0xFFF value corresponds to the voltage level on VREFP. This configuration results in a resolution that can be calculated using the following equation:

```
mV per ADC code = (VREFP - VREFN) / 4096
```

While the analog input pads can handle voltages beyond this range, the analog input voltages must remain within the limits prescribed by Table 32-44 on page 2138 to produce accurate results. The $V_{REFA+}$ and $V_{REFA-}$ specifications define the useful range for the external voltage references on VREFA+ and VREFA-, see Table 32-44 on page 2138. Care must be taken to supply a reference voltage of acceptable quality. Figure 18-9 on page 1215 shows the ADC conversion function of the analog inputs.

**Figure 18-9. ADC Conversion Result**




## 18.3.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the Dn bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** register. Differential pair 0 samples analog inputs 0 and 1; differential

pair 1 samples analog inputs 2 and 3; and so on (see Table 18-6 on page 1216). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

**Table 18-6. Differential Sampling Pairs**

| Differential Pair | Analog Inputs |
|---|---|
| 0 | 0 and 1 |
| 1 | 2 and 3 |
| 2 | 4 and 5 |
| 3 | 6 and 7 |
| 4 | 8 and 9 |
| 5 | 10 and 11 |
| 6 | 12 and 13 |
| 7 | 14 and 15 |
| 8 | 16 and 17 |
| 9 | 18 and 19 |
| 10 | 20 and 21 |
| 11 | 22 and 23 |

The voltage sampled in differential mode is the difference between the odd and even channels:

- Input Positive Voltage: VIN+ = $V_{IN_EVEN}$ (even channel)
- Input Negative Voltage: VIN- = $V_{IN_ODD}$ (odd channel)

The input differential voltage is defined as: $VIN_D$ = VIN+ - VIN-, therefore:

- If $VIN_D$ = 0, then the conversion result = 0x800
- If $VIN_D$ > 0, then the conversion result > 0x800 (range is 0x800–0xFFF)
- If $VIN_D$ < 0, then the conversion result < 0x800 (range is 0–0x800)

When using differential sampling, the following definitions are relevant:

- Input Common Mode Voltage: $VIN_{CM}$ = (VIN+ + VIN-) / 2
- Reference Positive Voltage: VREFP
- Reference Negative Voltage: VREFN
- Reference Differential Voltage: $VREF_D$ = VREFP - VREFN
- Reference Common Mode Voltage: $VREF_{CM}$ = (VREFP + VREFN) / 2

The following conditions provide optimal results in differential mode:

- Both $V_{IN_EVEN}$ and $V_{IN_ODD}$ must be in the range of (VREFP to VREFN) for a valid conversion result

- The maximum possible differential input swing, or the maximum differential range, is: -$VREF_D$to +$VREF_D$, so the maximum peak-to-peak input differential signal is (+$VREF_D$ - -$VREF_D$) = 2 * $VREF_D$= 2 * (VREFP - VREFN)
- In order to take advantage of the maximum possible differential input swing, $VIN_{CM}$ should be very close to $VREF_{CM}$, see Table 32-44 on page 2138.

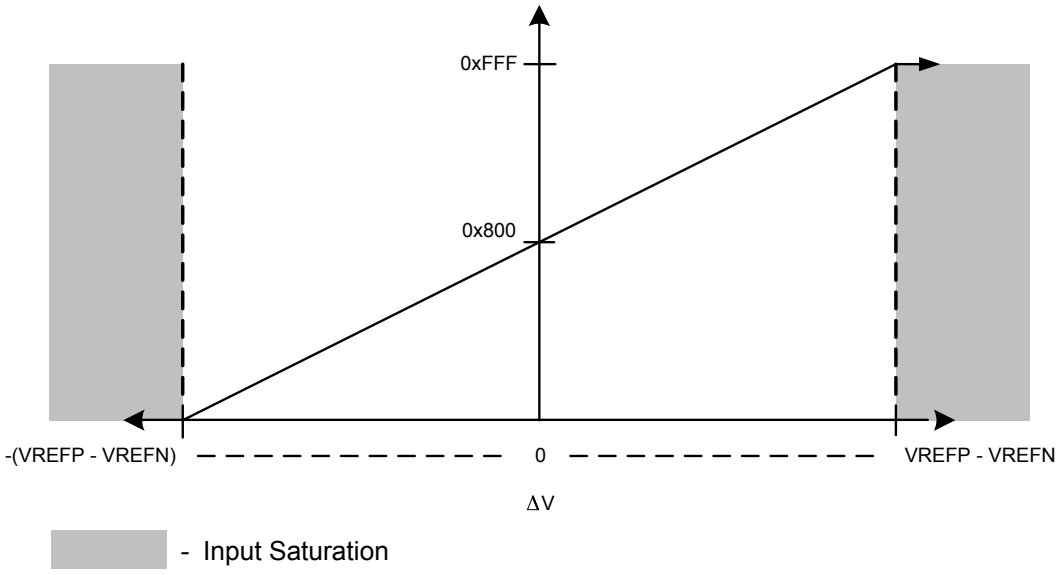
If $VIN_{CM}$ is not equal to $VREF_{CM}$, the differential input signal may clip at either maximum or minimum voltage, because either single ended input can never be larger than VREFP or smaller than VREFN, and it is not possible to achieve full swing. Thus any difference in common mode between the input voltage and the reference voltage limits the differential dynamic range of the ADC.

Because the maximum peak-to-peak differential signal voltage is 2 * (VREFP - VREFN), the ADC codes are interpreted as:

```
mV per ADC code = (2 *(VREFP - VREFN)) / 4096
```

Figure 18-10 shows how the differential voltage, ΔV, is represented in ADC codes.

**Figure 18-10. Differential Voltage Representation**




## 18.3.6 Internal Temperature Sensor

The temperature sensor serves two primary purposes: 1) to notify the system that internal temperature is too high or low for reliable operation and 2) to provide temperature measurements for calibration of the Hibernate module RTC trim value.

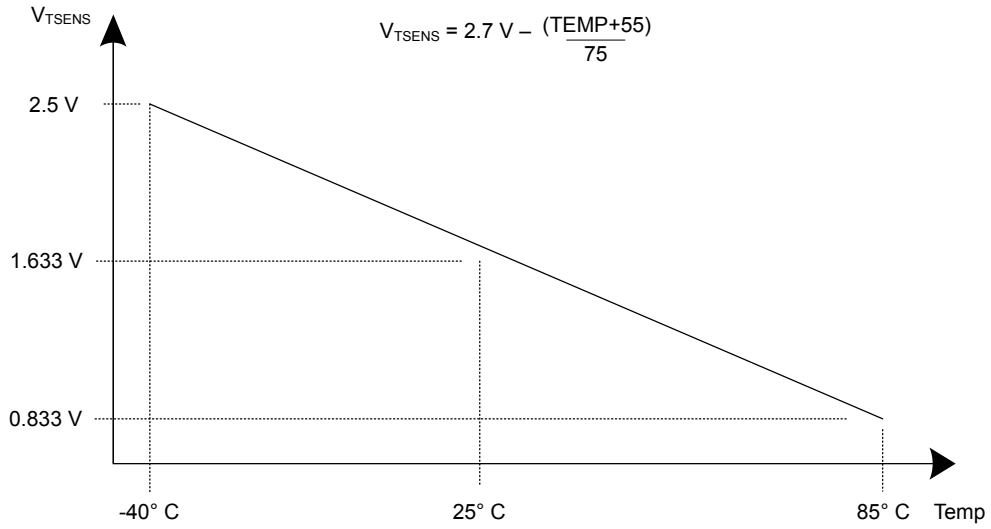
The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC. In addition, the temperature sensor has a second power-down input in the 3.3 V domain which provides control by the Hibernation module.

The internal temperature sensor converts a temperature measurement into a voltage. This voltage value, $V_{TSENS}$, is given by the following equation (where TEMP is the temperature in °C):

$V_{TSENS}$ = 2.7 - ((TEMP + 55) / 75)

This relation is shown in Figure 18-11 on page 1218.

**Figure 18-11. Internal Temperature Sensor Characteristic**




The temperature sensor reading can be sampled in a sample sequence by setting the TSn bit in the **ADCSSCTLn** register. The sample and hold width should be configured for at least 16 ADC clocks using the **ADCSSTSHn** register. The temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (TEMP in ℃) based on the ADC reading ($ADC_{CODE}$, given as an unsigned decimal number from 0 to 4095) and the maximum ADC voltage range (VREFP - VREFN):

TEMP = 147.5 - ((75 * (VREFP - VREFN) × $ADC_{CODE}$) / 4096)

## 18.3.7 Digital Comparator Unit

An ADC is commonly used to sample an external signal and to monitor its value to ensure that it remains in a given range. To automate this monitoring procedure and reduce the amount of processor overhead that is required, each module provides eight digital comparators.

Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the **ADC Digital Comparator Range (ADCDCCMPn)** registers. The ADC can be configured to generate an interrupt depending on whether the ADC is operating within the low, mid or high-band region configured in the ADCDCCMPn bit fields. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be additionally applied to the interrupt configuration.

### 18.3.7.1 Output Functions

ADC conversions can either be stored in the ADC Sample Sequence FIFOs or compared using the digital comparator resources as defined by the SnDCOP bits in the **ADC Sample Sequence n Operation (ADCSSOPn)** register. These selected ADC conversions are used by their respective

digital comparator to monitor the external signal. Each comparator has two possible output functions: processor interrupts and triggers.

Each function has its own state machine to track the monitored signal. Even though the interrupt and trigger functions can be enabled individually or both at the same time, the same conversion data is used by each function to determine if the right conditions have been met to assert the associated output.

#### *Interrupts*

The digital comparator interrupt function is enabled by setting the CIE bit in the **ADC Digital Comparator Control (ADCDCCTLn)** register. This bit enables the interrupt function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, and the DCONSSx bit is set in the **ADCIM** register, an interrupt is sent to the interrupt controller.

**Note:** For a 1 to 2 Msps rate, as the system clock frequency approaches the ADC clock frequency, it is recommended that the application use the µDMA to store conversion data from the FIFO to memory before processing rather than an interrupt-driven single data read. Using the µDMA to store multiple samples before interrupting the processor amortizes interrupt overhead across multiple transfers and prevents loss of sample data.

**Note:** Only a single DCONSSn bit should be set at any given time. Setting more than one of these bits results in the INRDC bit from the **ADCRIS** register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines. It is recommended that when interrupts are used, they are enabled on alternating samples or at the end of the sample sequence.

#### *Triggers*

The digital comparator trigger function is enabled by setting the CTE bit in the **ADCDCCTLn** register. This bit enables the trigger function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, the corresponding digital comparator trigger to the PWM module is asserted.

### 18.3.7.2 Operational Modes

Four operational modes are provided to support a broad range of applications and multiple possible signaling requirements: Always, Once, Hysteresis Always, and Hysteresis Once. The operational mode is selected using the CIM or CTM field in the **ADCDCCTLn** register.

#### *Always Mode*

In the Always operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria. The result is a string of assertions on the interrupt or trigger while the conversions are within the appropriate range.

#### *Once Mode*

In the Once operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria, and the previous ADC conversion value did not. The result is a single assertion of the interrupt or trigger when the conversions are within the appropriate range.

#### *Hysteresis-Always Mode*

The Hysteresis-Always operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Always mode, the associated interrupt or trigger

is asserted in the following cases: 1) the ADC conversion value meets its comparison criteria or 2) a previous ADC conversion value has met the comparison criteria, and the hysteresis condition has not been cleared by entering the opposite region. The result is a string of assertions on the interrupt or trigger that continue until the opposite region is entered.

#### *Hysteresis-Once Mode*

The Hysteresis-Once operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Once mode, the associated interrupt or trigger is asserted only when the ADC conversion value meets its comparison criteria, the hysteresis condition is clear, and the previous ADC conversion did not meet the comparison criteria. The result is a single assertion on the interrupt or trigger.

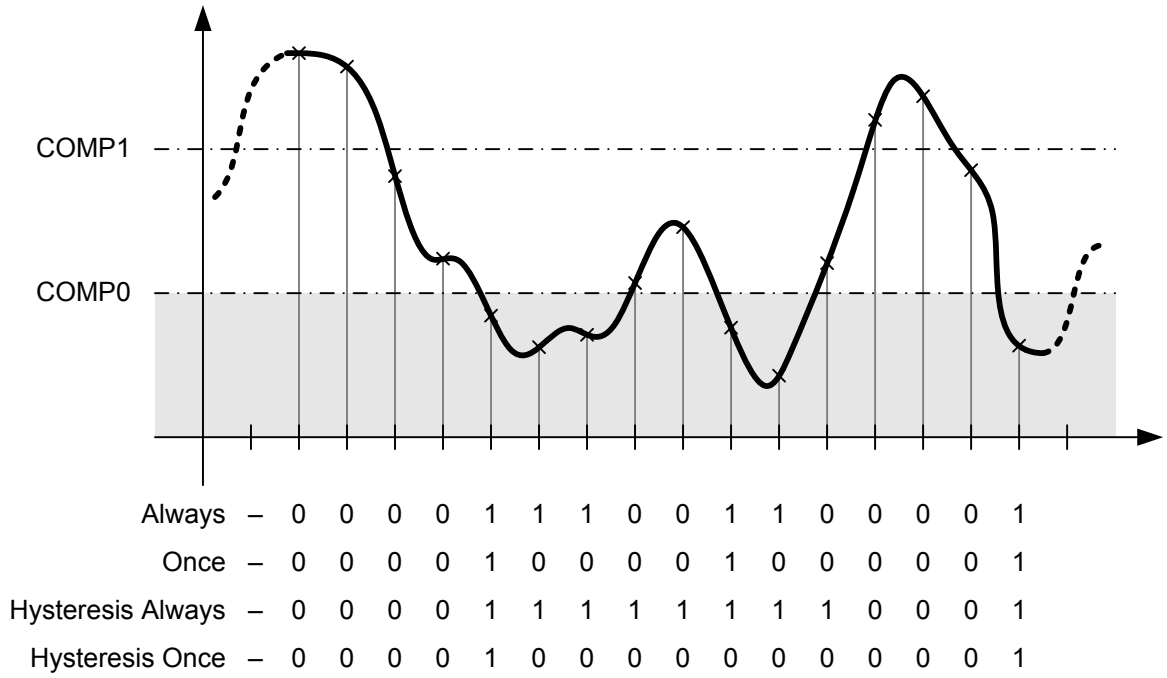
### 18.3.7.3 Function Ranges

The two comparison values, COMP0 and COMP1, in the **ADC Digital Comparator Range (ADCDCCMPn)** register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than COMP0), mid-band (greater than COMP0 but less than or equal to COMP1), and high-band (greater than or equal to COMP1) regions. COMP0 and COMP1 may be programmed to the same value, effectively creating two regions, but COMP1 must always be greater than or equal to the value of COMP0. A COMP1 value that is less than COMP0 generates unpredictable results.

#### *Low-Band Operation*

To operate in the low-band region, the CIC field or the CTC field in the **ADCDCCTLn** register must be programmed to 0x0. This setting causes interrupts or triggers to be generated in the low-band region as defined by the programmed operational mode. An example of the state of the interrupt/trigger signal in the low-band region for each of the operational modes is shown in Figure 18-12 on page 1221. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is deasserted and a "1" indicates that the signal is asserted.

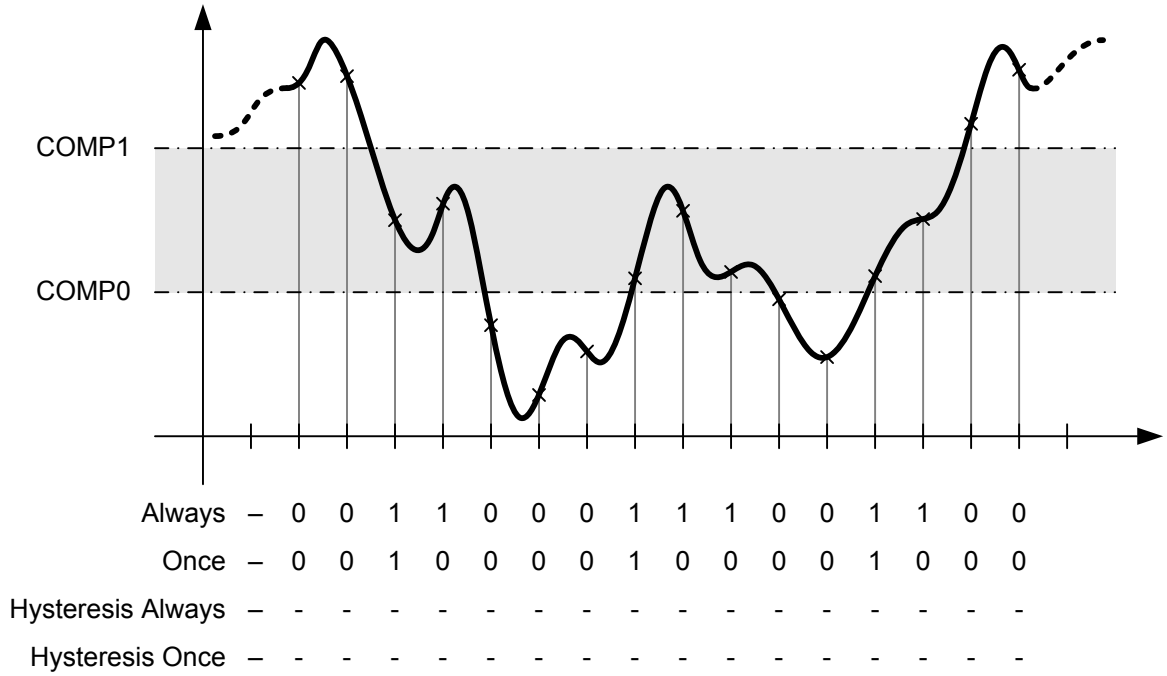
**Figure 18-12. Low-Band Operation (CIC=0x0 and/or CTC=0x0)**




### *Mid-Band Operation*

To operate in the mid-band region, the CIC field or the CTC field in the **ADCDCCTLn** register must be programmed to 0x1. This setting causes interrupts or triggers to be generated in the mid-band region according the operation mode. Only the Always and Once operational modes are available in the mid-band region. An example of the state of the interrupt/trigger signal in the mid-band region for each of the allowed operational modes is shown in Figure 18-13 on page 1222. Note that a "0" in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is deasserted and a "1" indicates that the signal is asserted.

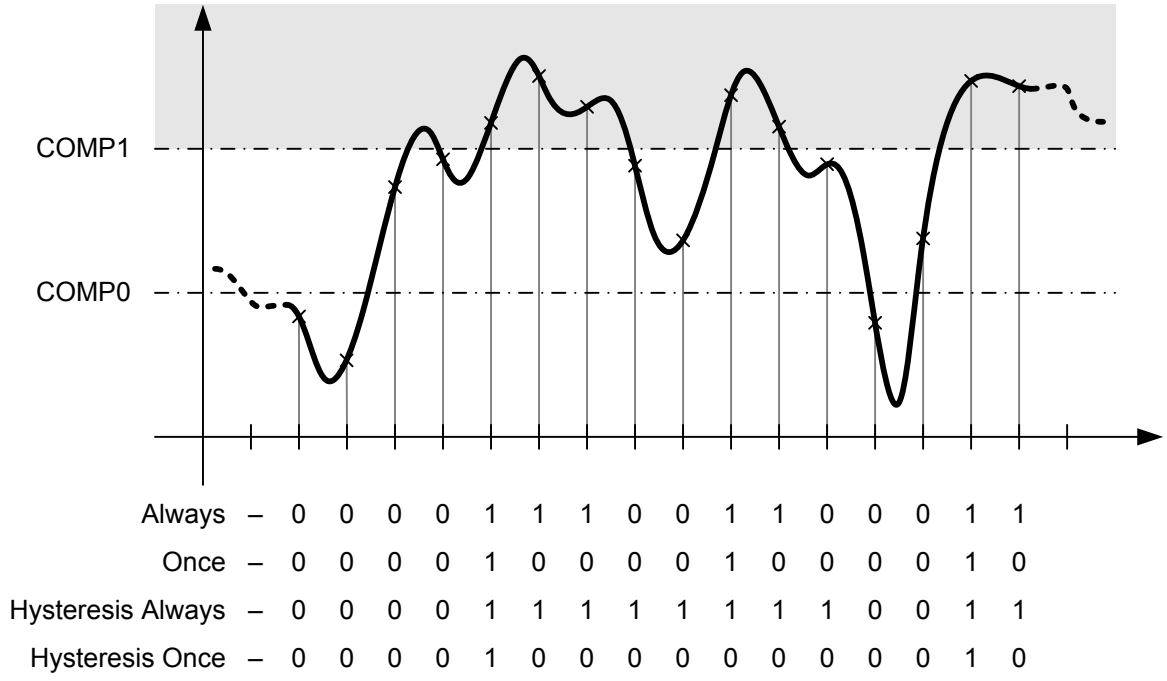
**Figure 18-13. Mid-Band Operation (CIC=0x1 and/or CTC=0x1)**




### *High-Band Operation*

To operate in the high-band region, the CIC field or the CTC field in the **ADCDCCTLn** register must be programmed to 0x3. This setting causes interrupts or triggers to be generated in the high-band region according the operation mode. An example of the state of the interrupt/trigger signal in the high-band region for each of the allowed operational modes is shown in Figure 18-14 on page 1223. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is deasserted and a "1" indicates that the signal is asserted.

**Figure 18-14. High-Band Operation (CIC=0x3 and/or CTC=0x3)**




## 18.4 Initialization and Configuration

### 18.4.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps: enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock using the **RCGCADC** register (see page 413).
2. Enable the clock to the appropriate GPIO modules via the **RCGCGPIO** register (see page 399). To find out which GPIO ports to enable, refer to “Signal Description” on page 1205.
3. Set the GPIO AFSEL bits for the ADC input pins (see page 803). To determine which GPIOs to configure, see Table 31-4 on page 2064.
4. Configure the AINx signals to be analog inputs by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register (see page 814).
5. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** register (see page 819) in the associated GPIO block.
6. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

### 18.4.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization because each sample sequencer is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by clearing the corresponding ASENn bit in the **ADCACTSS** register. Programming of the sample sequencers is allowed without having them enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.
2. Configure the trigger event for the sample sequencer in the **ADCEMUX** register.
3. When using a PWM generator as the trigger source, use the **ADC Trigger Source Select (ADCTSSEL)** register to specify in which PWM module the generator is located. The default register reset selects PWM module 0 for all generators.
4. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** and **ADCSSEMUXn** registers.
5. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the END bit is set. Failure to set the END bit causes unpredictable behavior.
6. If interrupts are to be used, set the corresponding MASK bit in the **ADCIM** register.
7. Enable the sample sequencer logic by setting the corresponding ASENn bit in the **ADCACTSS** register.

## 18.5 Register Map

Table 18-7 on page 1224 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to that ADC module's base address of:

- ADC0: 0x4003.8000
- ADC1: 0x4003.9000

Note that the ADC module clock must be enabled before the registers can be programmed (see page 413). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

**Table 18-7. ADC Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | ADCACTSS | RW | 0x0000.0000 | ADC Active Sample Sequencer | 1228 |
| 0x004 | ADCRIS | RO | 0x0000.0000 | ADC Raw Interrupt Status | 1230 |
| 0x008 | ADCIM | RW | 0x0000.0000 | ADC Interrupt Mask | 1233 |
| 0x00C | ADCISC | RW1C | 0x0000.0000 | ADC Interrupt Status and Clear | 1236 |
| 0x010 | ADCOSTAT | RW1C | 0x0000.0000 | ADC Overflow Status | 1240 |
| 0x014 | ADCEMUX | RW | 0x0000.0000 | ADC Event Multiplexer Select | 1242 |

**Table 18-7. ADC Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x018 | ADCUSTAT | RW1C | 0x0000.0000 | ADC Underflow Status | 1247 |
| 0x01C | ADCTSSEL | RW | 0x0000.0000 | ADC Trigger Source Select | 1248 |
| 0x020 | ADCSSPRI | RW | 0x0000.3210 | ADC Sample Sequencer Priority | 1250 |
| 0x024 | ADCSPC | RW | 0x0000.0000 | ADC Sample Phase Control | 1252 |
| 0x028 | ADCPSSI | RW | - | ADC Processor Sample Sequence Initiate | 1254 |
| 0x030 | ADCSAC | RW | 0x0000.0000 | ADC Sample Averaging Control | 1256 |
| 0x034 | ADCDCISC | RW1C | 0x0000.0000 | ADC Digital Comparator Interrupt Status and Clear | 1257 |
| 0x038 | ADCCTL | RW | 0x0000.0000 | ADC Control | 1259 |
| 0x040 | ADCSSMUX0 | RW | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 0 | 1260 |
| 0x044 | ADCSSCTL0 | RW | 0x0000.0000 | ADC Sample Sequence Control 0 | 1262 |
| 0x048 | ADCSSFIFO0 | RO | - | ADC Sample Sequence Result FIFO 0 | 1269 |
| 0x04C | ADCSSFSTAT0 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 0 Status | 1270 |
| 0x050 | ADCSSOP0 | RW | 0x0000.0000 | ADC Sample Sequence 0 Operation | 1272 |
| 0x054 | ADCSSDC0 | RW | 0x0000.0000 | ADC Sample Sequence 0 Digital Comparator Select | 1274 |
| 0x058 | ADCSSEMUX0 | RW | 0x0000.0000 | ADC Sample Sequence Extended Input Multiplexer Select 0 | 1276 |
| 0x05C | ADCSSTSH0 | RW | 0x0000.0000 | ADC Sample Sequence 0 Sample and Hold Time | 1278 |
| 0x060 | ADCSSMUX1 | RW | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 1 | 1280 |
| 0x064 | ADCSSCTL1 | RW | 0x0000.0000 | ADC Sample Sequence Control 1 | 1281 |
| 0x068 | ADCSSFIFO1 | RO | - | ADC Sample Sequence Result FIFO 1 | 1269 |
| 0x06C | ADCSSFSTAT1 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 1 Status | 1270 |
| 0x070 | ADCSSOP1 | RW | 0x0000.0000 | ADC Sample Sequence 1 Operation | 1285 |
| 0x074 | ADCSSDC1 | RW | 0x0000.0000 | ADC Sample Sequence 1 Digital Comparator Select | 1286 |
| 0x078 | ADCSSEMUX1 | RW | 0x0000.0000 | ADC Sample Sequence Extended Input Multiplexer Select 1 | 1288 |
| 0x07C | ADCSSTSH1 | RW | 0x0000.0000 | ADC Sample Sequence 1 Sample and Hold Time | 1290 |
| 0x080 | ADCSSMUX2 | RW | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 2 | 1280 |
| 0x084 | ADCSSCTL2 | RW | 0x0000.0000 | ADC Sample Sequence Control 2 | 1281 |
| 0x088 | ADCSSFIFO2 | RO | - | ADC Sample Sequence Result FIFO 2 | 1269 |
| 0x08C | ADCSSFSTAT2 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 2 Status | 1270 |
| 0x090 | ADCSSOP2 | RW | 0x0000.0000 | ADC Sample Sequence 2 Operation | 1285 |
| 0x094 | ADCSSDC2 | RW | 0x0000.0000 | ADC Sample Sequence 2 Digital Comparator Select | 1286 |

**Table 18-7. ADC Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x098 | ADCSSEMUX2 | RW | 0x0000.0000 | ADC Sample Sequence Extended Input Multiplexer Select 2 | 1288 |
| 0x09C | ADCSSTSH2 | RW | 0x0000.0000 | ADC Sample Sequence 2 Sample and Hold Time | 1290 |
| 0x0A0 | ADCSSMUX3 | RW | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 3 | 1292 |
| 0x0A4 | ADCSSCTL3 | RW | 0x0000.0000 | ADC Sample Sequence Control 3 | 1293 |
| 0x0A8 | ADCSSFIFO3 | RO | - | ADC Sample Sequence Result FIFO 3 | 1269 |
| 0x0AC | ADCSSFSTAT3 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 3 Status | 1270 |
| 0x0B0 | ADCSSOP3 | RW | 0x0000.0000 | ADC Sample Sequence 3 Operation | 1295 |
| 0x0B4 | ADCSSDC3 | RW | 0x0000.0000 | ADC Sample Sequence 3 Digital Comparator Select | 1296 |
| 0x0B8 | ADCSSEMUX3 | RW | 0x0000.0000 | ADC Sample Sequence Extended Input Multiplexer Select 3 | 1297 |
| 0x0BC | ADCSSTSH3 | RW | 0x0000.0000 | ADC Sample Sequence 3 Sample and Hold Time | 1298 |
| 0xD00 | ADCDCRIC | WO | 0x0000.0000 | ADC Digital Comparator Reset Initial Conditions | 1299 |
| 0xE00 | ADCDCCTL0 | RW | 0x0000.0000 | ADC Digital Comparator Control 0 | 1304 |
| 0xE04 | ADCDCCTL1 | RW | 0x0000.0000 | ADC Digital Comparator Control 1 | 1304 |
| 0xE08 | ADCDCCTL2 | RW | 0x0000.0000 | ADC Digital Comparator Control 2 | 1304 |
| 0xE0C | ADCDCCTL3 | RW | 0x0000.0000 | ADC Digital Comparator Control 3 | 1304 |
| 0xE10 | ADCDCCTL4 | RW | 0x0000.0000 | ADC Digital Comparator Control 4 | 1304 |
| 0xE14 | ADCDCCTL5 | RW | 0x0000.0000 | ADC Digital Comparator Control 5 | 1304 |
| 0xE18 | ADCDCCTL6 | RW | 0x0000.0000 | ADC Digital Comparator Control 6 | 1304 |
| 0xE1C | ADCDCCTL7 | RW | 0x0000.0000 | ADC Digital Comparator Control 7 | 1304 |
| 0xE40 | ADCDCCMP0 | RW | 0x0000.0000 | ADC Digital Comparator Range 0 | 1307 |
| 0xE44 | ADCDCCMP1 | RW | 0x0000.0000 | ADC Digital Comparator Range 1 | 1307 |
| 0xE48 | ADCDCCMP2 | RW | 0x0000.0000 | ADC Digital Comparator Range 2 | 1307 |
| 0xE4C | ADCDCCMP3 | RW | 0x0000.0000 | ADC Digital Comparator Range 3 | 1307 |
| 0xE50 | ADCDCCMP4 | RW | 0x0000.0000 | ADC Digital Comparator Range 4 | 1307 |
| 0xE54 | ADCDCCMP5 | RW | 0x0000.0000 | ADC Digital Comparator Range 5 | 1307 |
| 0xE58 | ADCDCCMP6 | RW | 0x0000.0000 | ADC Digital Comparator Range 6 | 1307 |
| 0xE5C | ADCDCCMP7 | RW | 0x0000.0000 | ADC Digital Comparator Range 7 | 1307 |
| 0xFC0 | ADCPP | RO | 0x01B0.2187 | ADC Peripheral Properties | 1308 |
| 0xFC4 | ADCPC | RW | 0x0000.0007 | ADC Peripheral Configuration | 1310 |
| 0xFC8 | ADCCC | RW | 0x0000.0001 | ADC Clock Configuration | 1311 |

## 18.6 Register Descriptions

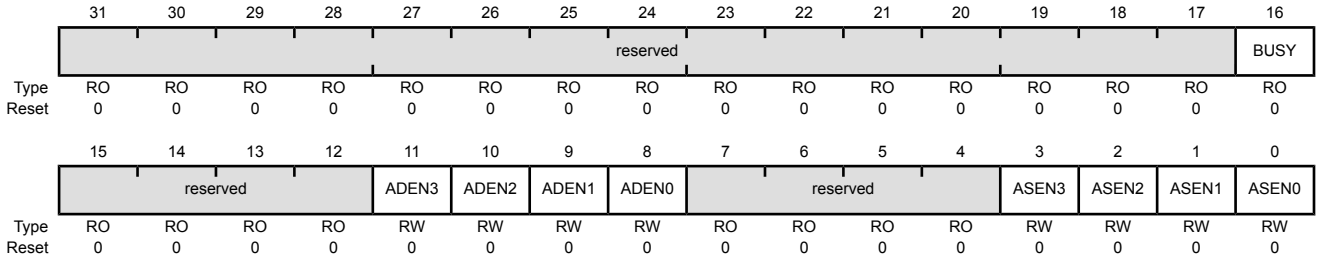
The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

## Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0 | 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. |
| 16 | BUSY | RO | 0 | ADC Busy<br><br>Value Description<br>0 ADC is idle<br>1 ADC is busy<br><br>**Note:** In order to use the BUSY bit, the **ADC Event Multiplexer Select (ADCEMUX)** register must be programmed such that no trigger is selected (bit field encoding is 0xE). The NEVER encoding in the **ADCEMUX** register allows the ADC to safely be put in Deep-Sleep mode. |
| 15:12 | reserved | RO | 0 | 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. |
| 11 | ADEN3 | RW | 0 | ADC SS3 DMA Enable<br><br>Value Description<br>0 DMA for Sample Sequencer 3 is disabled.<br>1 DMA for Sample Sequencer 3 is enabled. |
| 10 | ADEN2 | RW | 0 | ADC SS2 DMA Enable<br><br>Value Description<br>0 DMA for Sample Sequencer 2 is disabled.<br>1 DMA for Sample Sequencer 2 is enabled. |

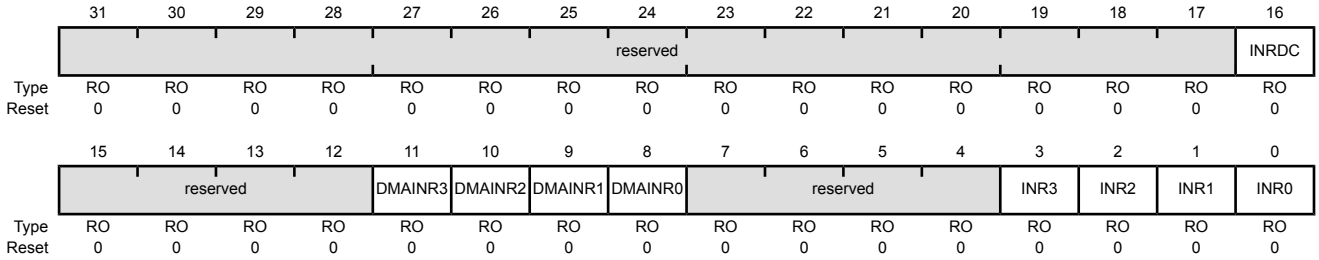
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | ADEN1 | RW | 0 | ADC SS1 DMA Enable<br>Value Description<br>0 DMA for Sample Sequencer 1 is disabled.<br>1 DMA for Sample Sequencer 1 is enabled. |
| 8 | ADEN0 | RW | 0 | ADC SS1 DMA Enable<br>Value Description<br>0 DMA for Sample Sequencer 1 is disabled.<br>1 DMA for Sample Sequencer 1 is enabled. |
| 7:4 | reserved | RO | 0 | 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. |
| 3 | ASEN3 | RW | 0 | ADC SS3 Enable<br>Value Description<br>0 Sample Sequencer 3 is disabled.<br>1 Sample Sequencer 3 is enabled. |
| 2 | ASEN2 | RW | 0 | ADC SS2 Enable<br>Value Description<br>0 Sample Sequencer 2 is disabled.<br>1 Sample Sequencer 2 is enabled. |
| 1 | ASEN1 | RW | 0 | ADC SS1 Enable<br>Value Description<br>0 Sample Sequencer 1 is disabled.<br>1 Sample Sequencer 1 is enabled. |
| 0 | ASEN0 | RW | 0 | ADC SS0 Enable<br>Value Description<br>0 Sample Sequencer 0 is disabled.<br>1 Sample Sequencer 0 is enabled. |

## Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without sending the interrupts to the interrupt controller.

ADC Raw Interrupt Status (ADCRIS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x004
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0x000 | 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. |
| 16 | INRDC | RO | 0 | Digital Comparator Raw Interrupt Status<br><br>Value Description<br>0 All bits in the **ADCDCISC** register are clear.<br>1 At least one bit in the **ADCDCISC** register is set, meaning that a digital comparator interrupt has occurred. |
| 15:12 | reserved | RO | 0 | 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. |
| 11 | DMAINR3 | RO | 0 | SS3 DMA Raw Interrupt Status<br><br>Value Description<br>0 The DMA interrupt has not occurred.<br>1 The sample sequence 3 DMA interrupt is asserted.<br><br>This bit is cleared by writing a 1 to the `DMAINR3` bit in the **ADCISC** register. |
| 10 | DMAINR2 | RO | 0 | SS2 DMA Raw Interrupt Status<br><br>Value Description<br>0 The DMA interrupt has not occurred.<br>1 The sample sequence 2 DMA interrupt is asserted.<br><br>This bit is cleared by writing a 1 to the `DMAINR2` bit in the **ADCISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | DMAINR1 | RO | 0 | SS1 DMA Raw Interrupt Status<br><br>Value Description<br>0 The DMA interrupt has not occurred.<br>1 The sample sequence 1 DMA interrupt is asserted.<br><br>This bit is cleared by writing a 1 to the DMAINR1 bit in the **ADCISC** register. |
| 8 | DMAINR0 | RO | 0 | SS0 DMA Raw Interrupt Status<br><br>Value Description<br>0 The DMA interrupt has not occurred.<br>1 The sample sequence 0 DMA interrupt is asserted.<br><br>This bit is cleared by writing a 1 to the DMAINR0 bit in the **ADCISC** register. |
| 7:4 | reserved | RO | 0 | 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. |
| 3 | INR3 | RO | 0 | SS3 Raw Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 A sample has completed conversion and the respective **ADCSSCTL3** IEn bit is set, enabling a raw interrupt.<br><br>This bit is cleared by writing a 1 to the IN3 bit in the **ADCISC** register. |
| 2 | INR2 | RO | 0 | SS2 Raw Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 A sample has completed conversion and the respective **ADCSSCTL2** IEn bit is set, enabling a raw interrupt.<br><br>This bit is cleared by writing a 1 to the IN2 bit in the **ADCISC** register. |
| 1 | INR1 | RO | 0 | SS1 Raw Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 A sample has completed conversion and the respective **ADCSSCTL1** IEn bit is set, enabling a raw interrupt.<br><br>This bit is cleared by writing a 1 to the IN1 bit in the **ADCISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INR0 | RO | 0 | SS0 Raw Interrupt Status |

| Value | Description |
|---|---|
| 0 | An interrupt has not occurred. |
| 1 | A sample has completed conversion and the respective **ADCSSCTL0** IEn bit is set, enabling a raw interrupt. |

This bit is cleared by writing a 1 to the IN0 bit in the **ADCISC** register.

## Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

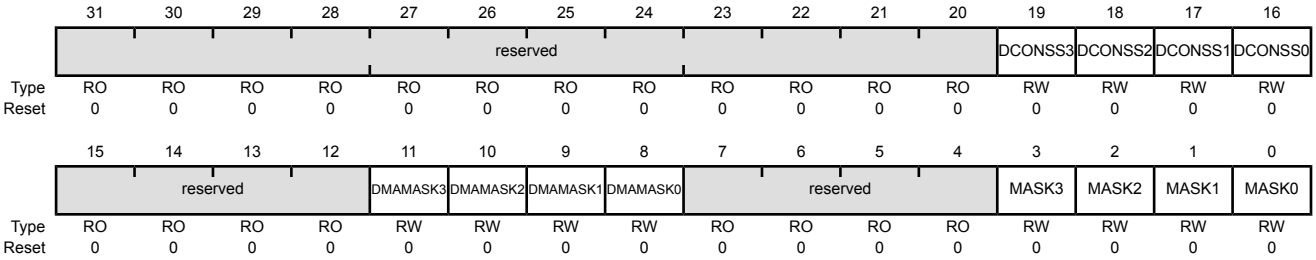
This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently.

**Note:** For a 1 to 2 Msps rate, as the system clock frequency approaches the ADC clock frequency, it is recommended that the application use the µDMA to store conversion data from the FIFO to memory before processing rather than an interrupt-driven single data read. Using the µDMA to store multiple samples before interrupting the processor amortizes interrupt overhead across multiple transfers and prevents loss of sample data.

**Note:** Only a single DCONSSn bit should be set at any given time. Setting more than one of these bits results in the INRDC bit from the **ADCRIS** register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines. It is recommended that when interrupts are used, they are enabled on alternating samples or at the end of the sample sequence.

ADC Interrupt Mask (ADCIM)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x008
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x000 | 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. |
| 19 | DCONSS3 | RW | 0 | Digital Comparator Interrupt on SS3<br><br>Value Description<br>0 The status of the digital comparators does not affect the SS3 interrupt status.<br>1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS3 interrupt line. |
| 18 | DCONSS2 | RW | 0 | Digital Comparator Interrupt on SS2<br><br>Value Description<br>0 The status of the digital comparators does not affect the SS2 interrupt status.<br>1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS2 interrupt line. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | DCONSS1 | RW | 0 | Digital Comparator Interrupt on SS1<br><br>Value Description<br>0 The status of the digital comparators does not affect the SS1 interrupt status.<br>1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS1 interrupt line. |
| 16 | DCONSS0 | RW | 0 | Digital Comparator Interrupt on SS0<br><br>Value Description<br>0 The status of the digital comparators does not affect the SS0 interrupt status.<br>1 The raw interrupt signal from the digital comparators (INRDC bit in the **ADCRIS** register) is sent to the interrupt controller on the SS0 interrupt line. |
| 15:12 | reserved | RO | 0 | 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. |
| 11 | DMAMASK3 | RW | 0 | SS3 DMA Interrupt Mask<br><br>Value Description<br>0 The status of Sample Sequencer 3 DMA does not affect the SS3 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 3 DMA (**ADCRIS** register DMAINR3 bit) is sent to the interrupt controller. |
| 10 | DMAMASK2 | RW | 0 | SS2 DMA Interrupt Mask<br><br>Value Description<br>0 The status of Sample Sequencer 2 DMA does not affect the SS2 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 2 DMA (**ADCRIS** register DMAINR2 bit) is sent to the interrupt controller. |
| 9 | DMAMASK1 | RW | 0 | SS1 DMA Interrupt Mask<br><br>Value Description<br>0 The status of Sample Sequencer 1 DMA does not affect the SS1 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 1 DMA (**ADCRIS** register DMAINR1 bit) is sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | DMAMASK0 | RW | 0 | SS0 DMA Interrupt Mask<br>Value Description<br>0 The status of Sample Sequencer 0 DMA does not affect the SS0 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 0 DMA (**ADCRIS** register DMAINR0 bit) is sent to the interrupt controller. |
| 7:4 | reserved | RO | 0 | 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. |
| 3 | MASK3 | RW | 0 | SS3 Interrupt Mask<br>Value Description<br>0 The status of Sample Sequencer 3 does not affect the SS3 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 3 (**ADCRIS** register INR3 bit) is sent to the interrupt controller. |
| 2 | MASK2 | RW | 0 | SS2 Interrupt Mask<br>Value Description<br>0 The status of Sample Sequencer 2 does not affect the SS2 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 2 (**ADCRIS** register INR2 bit) is sent to the interrupt controller. |
| 1 | MASK1 | RW | 0 | SS1 Interrupt Mask<br>Value Description<br>0 The status of Sample Sequencer 1 does not affect the SS1 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 1 (**ADCRIS** register INR1 bit) is sent to the interrupt controller. |
| 0 | MASK0 | RW | 0 | SS0 Interrupt Mask<br>Value Description<br>0 The status of Sample Sequencer 0 does not affect the SS0 interrupt status.<br>1 The raw interrupt signal from Sample Sequencer 0 (**ADCRIS** register INR0 bit) is sent to the interrupt controller. |

## Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective INR and MASK bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the **ADCDCISC** register. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence INRn bits are still cleared via the **ADCISC** register, even if the INn bit is not set.

ADC Interrupt Status and Clear (ADCISC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x00C
Type RW1C, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | DCINSS3 | DCINSS2 | DCINSS1 | DCINSS0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | DMAIN3 | DMAIN2 | DMAIN1 | DMAIN0 | reserved | | | | IN3 | IN2 | IN1 | IN0 |
| Type | RO | RO | RO | RO | RW1C | RW1C | RW1C | RW1C | RO | RO | RO | RO | RW1C | RW1C | RW1C | RW1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x000 | 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. |
| 19 | DCINSS3 | RO | 0 | Digital Comparator Interrupt Status on SS3<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INRDC bit in the **ADCRIS** register and the DCONSS3 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register. |
| 18 | DCINSS2 | RO | 0 | Digital Comparator Interrupt Status on SS2<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INRDC bit in the **ADCRIS** register and the DCONSS2 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | DCINSS1 | RO | 0 | Digital Comparator Interrupt Status on SS1<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INRDC bit in the **ADCRIS** register and the DCONSS1 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register. |
| 16 | DCINSS0 | RO | 0 | Digital Comparator Interrupt Status on SS0<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INRDC bit in the **ADCRIS** register and the DCONSS0 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register. |
| 15:12 | reserved | RO | 0 | 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. |
| 11 | DMAIN3 | RW1C | 0 | SS3 DMA Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the DMAINR3 bit in the **ADCRIS** register and the DMAMASK3 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR3 bit in the **ADCRIS** register. |
| 10 | DMAIN2 | RW1C | 0 | SS2 DMA Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the DMAINR2 bit in the **ADCRIS** register and the DMAMASK2 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR2 bit in the **ADCRIS** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | DMAIN1 | RW1C | 0 | SS1 DMA Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the DMAINR1 bit in the **ADCRIS** register and the DMAMASK1 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR1 bit in the **ADCRIS** register. |
| 8 | DMAIN0 | RW1C | 0 | SS0 DMA Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the DMAINR0 bit in the **ADCRIS** register and the DMAMASK0 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the DMAINR0 bit in the **ADCRIS** register. |
| 7:4 | reserved | RO | 0 | 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. |
| 3 | IN3 | RW1C | 0 | SS3 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INR3 bit in the **ADCRIS** register and the MASK3 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit in the **ADCRIS** register. |
| 2 | IN2 | RW1C | 0 | SS2 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INR2 bit in the **ADCRIS** register and the MASK2 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit in the **ADCRIS** register. |

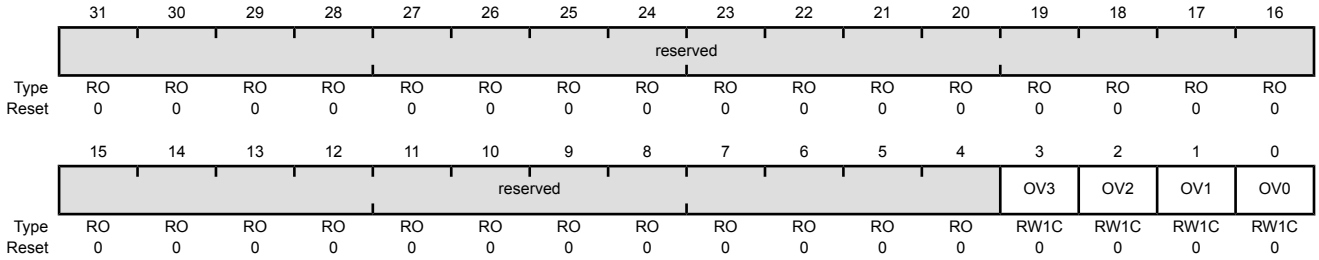
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | IN1 | RW1C | 0 | SS1 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INR1 bit in the **ADCRIS** register and the MASK1 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit in the **ADCRIS** register. |
| 0 | IN0 | RW1C | 0 | SS0 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 Both the INR0 bit in the **ADCRIS** register and the MASK0 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INR0 bit in the **ADCRIS** register. |

## Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADC Overflow Status (ADCOSTAT)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x010
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | OV3 | RW1C | 0 | SS3 FIFO Overflow<br><br>Value Description<br>0 The FIFO has not overflowed.<br>1 The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.<br><br>This bit is cleared by writing a 1. |
| 2 | OV2 | RW1C | 0 | SS2 FIFO Overflow<br><br>Value Description<br>0 The FIFO has not overflowed.<br>1 The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.<br><br>This bit is cleared by writing a 1. |
| 1 | OV1 | RW1C | 0 | SS1 FIFO Overflow<br><br>Value Description<br>0 The FIFO has not overflowed.<br>1 The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.<br><br>This bit is cleared by writing a 1. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | OV0 | RW1C | 0 | SS0 FIFO Overflow |

| Value | Description |
|---|---|
| 0 | The FIFO has not overflowed. |
| 1 | The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. |

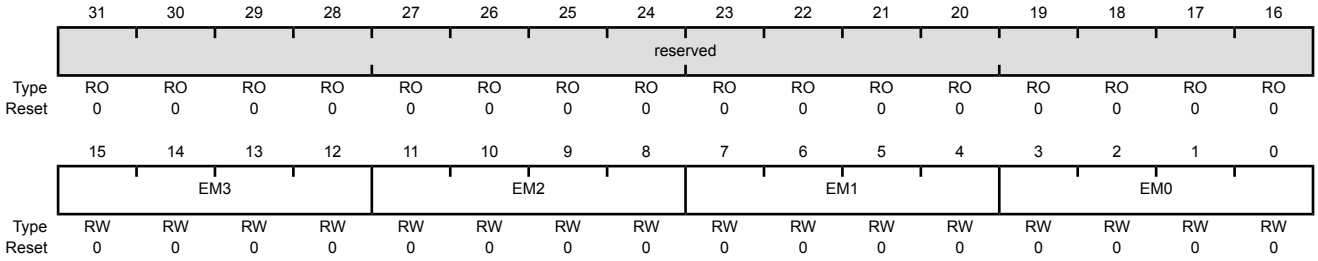
This bit is cleared by writing a 1.

## Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x014
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:12 | EM3 | RW | 0x0 | SS3 Trigger Select<br>This field selects the trigger source for Sample Sequencer 3.<br>The valid configurations for this field are: |

| Value | Event |
|---|---|
| 0x0 | Processor (default)<br>The trigger is initiated by setting the SSn bit in the **ADCPSSI** register. |
| 0x1 | Analog Comparator 0<br>This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1921). |
| 0x2 | Analog Comparator 1<br>This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1921). |
| 0x3 | Analog Comparator 2<br>This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1921). |
| 0x4 | External (GPIO Pins)<br>This trigger is connected to the GPIO interrupt for the corresponding GPIO (see "ADC Trigger Source" on page 782). |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 1136). |
| 0x6 | PWM generator 0<br>The PWM generator 0 trigger can be configured with the **PWM0 Interrupt and Trigger Enable (PWM0INTEN)** register (page 1969). |
| 0x7 | PWM generator 1<br>The PWM generator 1 trigger can be configured with the **PWM1INTEN** register (page 1969). |
| 0x8 | PWM generator 2<br>The PWM generator 2 trigger can be configured with the **PWM2INTEN** register (page 1969). |
| 0x9 | PWM generator 3<br>The PWM generator 3 trigger can be configured with the **PWM3INTEN** register (page 1969). |
| 0xA-0xD | reserved |
| 0xE | Never Trigger (No triggers are allowed to the ADC digital interface) |
| 0xF | Always (continuously sample) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:8 | EM2 | RW | 0x0 | SS2 Trigger Select<br>This field selects the trigger source for Sample Sequencer 2.<br>The valid configurations for this field are: |

| Value | Event |
|---|---|
| 0x0 | Processor (default)<br>The trigger is initiated by setting the SSn bit in the **ADCPSSI** register. |
| 0x1 | Analog Comparator 0<br>This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1921). |
| 0x2 | Analog Comparator 1<br>This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1921). |
| 0x3 | Analog Comparator 2<br>This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1921). |
| 0x4 | External (GPIO Pins)<br>This trigger is connected to the GPIO interrupt for the corresponding GPIO (see “ADC Trigger Source” on page 782). |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 1136). |
| 0x6 | PWM generator 0<br>The PWM generator 0 trigger can be configured with the **PWM0 Interrupt and Trigger Enable (PWM0INTEN)** register (page 1969). |
| 0x7 | PWM generator 1<br>The PWM generator 1 trigger can be configured with the **PWM1INTEN** register (page 1969). |
| 0x8 | PWM generator 2<br>The PWM generator 2 trigger can be configured with the **PWM2INTEN** register (page 1969). |
| 0x9 | PWM generator 3<br>The PWM generator 3 trigger can be configured with the **PWM3INTEN** register (page 1969). |
| 0xA-0xD | reserved |
| 0xE | Never Trigger (No triggers are allowed to the ADC digital interface) |
| 0xF | Always (continuously sample) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:4 | EM1 | RW | 0x0 | SS1 Trigger Select<br>This field selects the trigger source for Sample Sequencer 1.<br>The valid configurations for this field are: |

| Value | Event |
|---|---|
| 0x0 | Processor (default)<br>The trigger is initiated by setting the SSn bit in the **ADCPSSI** register. |
| 0x1 | Analog Comparator 0<br>This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1921). |
| 0x2 | Analog Comparator 1<br>This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1921). |
| 0x3 | Analog Comparator 2<br>This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1921). |
| 0x4 | External (GPIO Pins)<br>This trigger is connected to the GPIO interrupt for the corresponding GPIO (see "ADC Trigger Source" on page 782). |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 1136). |
| 0x6 | PWM generator 0<br>The PWM generator 0 trigger can be configured with the **PWM0 Interrupt and Trigger Enable (PWM0INTEN)** register (page 1969). |
| 0x7 | PWM generator 1<br>The PWM generator 1 trigger can be configured with the **PWM1INTEN** register (page 1969). |
| 0x8 | PWM generator 2<br>The PWM generator 2 trigger can be configured with the **PWM2INTEN** register (page 1969). |
| 0x9 | PWM generator 3<br>The PWM generator 3 trigger can be configured with the **PWM3INTEN** register (page 1969). |
| 0xA-0xD | reserved |
| 0xE | Never Trigger (No triggers are allowed to the ADC digital interface) |
| 0xF | Always (continuously sample) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | EM0 | RW | 0x0 | SS0 Trigger Select<br>This field selects the trigger source for Sample Sequencer 0<br>The valid configurations for this field are: |

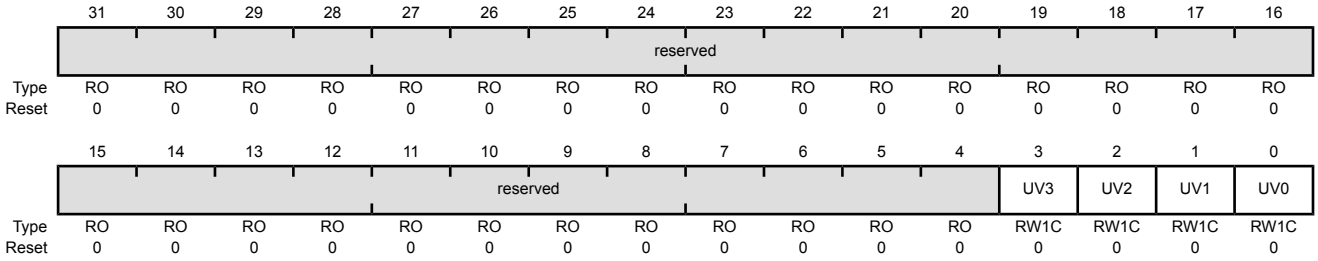
| Value | Event |
|---|---|
| 0x0 | Processor (default)<br>The trigger is initiated by setting the SSn bit in the **ADCPSSI** register. |
| 0x1 | Analog Comparator 0<br>This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1921). |
| 0x2 | Analog Comparator 1<br>This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1921). |
| 0x3 | Analog Comparator 2<br>This trigger is configured by the **Analog Comparator Control 2 (ACCTL2)** register (page 1921). |
| 0x4 | External (GPIO Pins)<br>This trigger is connected to the GPIO interrupt for the corresponding GPIO (see “ADC Trigger Source” on page 782). |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 1136). |
| 0x6 | PWM generator 0<br>The PWM generator 0 trigger can be configured with the **PWM0 Interrupt and Trigger Enable (PWM0INTEN)** register (page 1969). |
| 0x7 | PWM generator 1<br>The PWM generator 1 trigger can be configured with the **PWM1INTEN** register (page 1969). |
| 0x8 | PWM generator 2<br>The PWM generator 2 trigger can be configured with the **PWM2INTEN** register (page 1969). |
| 0x9 | PWM generator 3<br>The PWM generator 3 trigger can be configured with the **PWM3INTEN** register (page 1969). |
| 0xA-0xD | reserved |
| 0xE | Never Trigger (No triggers are allowed to the ADC digital interface) |
| 0xF | Always (continuously sample) |

## Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

ADC Underflow Status (ADCUSTAT)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x018
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | UV3 | RW1C | 0 | SS3 FIFO Underflow<br>The valid configurations for this field are shown below. This bit is cleared by writing a 1.<br>Value Description<br>0 The FIFO has not underflowed.<br>1 The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. |
| 2 | UV2 | RW1C | 0 | SS2 FIFO Underflow<br>The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1. |
| 1 | UV1 | RW1C | 0 | SS1 FIFO Underflow<br>The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1. |
| 0 | UV0 | RW1C | 0 | SS0 FIFO Underflow<br>The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1. |

## Register 8: ADC Trigger Source Select (ADCTSSEL), offset 0x01C

If a PWM Generator n is selected as a trigger source through the EMn bit field in the **ADC Event Multiplexer Select (ADCEMUX)** register, the **ADCTSSEL** register is programmed to identify in which PWM module instance the generator creating the trigger is located. The register resets to 0x0000.0000, which selects PWM module 0 for all generators. Note that field PS3 selects the PWM module that maps to Generator 3; PS2 selects the PWM module that maps to Generator 2, and so on.

ADC Trigger Source Select (ADCTSSEL)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x01C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | PS3 | | reserved | | | | | | PS2 | | reserved | | | |
| Type | RO | RO | RW | RW | RO | RO | RO | RO | RO | RO | RW | RW | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | PS1 | | reserved | | | | | | PS0 | | reserved | | | |
| Type | RO | RO | RW | RW | RO | RO | RO | RO | RO | RO | RW | RW | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:30 | reserved | RO | 0x0 | 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. |
| 29:28 | PS3 | RW | 0x0 | Generator 3 PWM Module Trigger Select<br>This field selects in which PWM module the generator 3 trigger is located.<br><br>Value Description<br>0x0 Use Generator 3 (and its trigger) in PWM module 0<br>0x1-0x3 reserved |
| 27:22 | reserved | RO | 0x0 | 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. |
| 21:20 | PS2 | RW | 0x0 | Generator 2 PWM Module Trigger Select<br>This field selects in which PWM module the Generator 2 trigger is located.<br><br>Value Description<br>0x0 Use Generator 2 (and its trigger) in PWM module 0<br>0x1-0x3 reserved |
| 19:14 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13:12 | PS1 | RW | 0x0 | Generator 1 PWM Module Trigger Select<br>This field selects in which PWM module the Generator 1 trigger is located.<br>Value: 0x0 — Use Generator 1 (and its trigger) in PWM module 0<br>Value: 0x1-0x3 — reserved |
| 11:6 | reserved | RO | 0x0 | 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 | PS0 | RW | 0x0 | Generator 0 PWM Module Trigger Select<br>This field selects in which PWM module the Generator 0 trigger is located.<br>Value: 0x0 — Use Generator 0 (and its trigger) in PWM module 0<br>Value: 0x1-0x3 — reserved |
| 3:0 | reserved | RO | 0x0 | 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. |

## Register 9: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADC Sample Sequencer Priority (ADCSSPRI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x020
Type RW, reset 0x0000.3210




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000.0 | 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. |
| 13:12 | SS3 | RW | 0x3 | SS3 Priority<br>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |
| 11:10 | reserved | RO | 0x0 | 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. |
| 9:8 | SS2 | RW | 0x2 | SS2 Priority<br>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |
| 7:6 | reserved | RO | 0x0 | 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 | SS1 | RW | 0x1 | SS1 Priority<br>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |
| 3:2 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | SS0 | RW | 0x0 | SS0 Priority<br>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |

## Register 10: ADC Sample Phase Control (ADCSPC), offset 0x024

The **ADC Sample Phase Control (ADCSPC)** register is used to insert a delay in ADC module sampling. This feature can be used with the SYNCWAIT and GSYNC bit in the **ADCPSSI** register to provide concurrent sampling of two different signals by two different ADC modules or skewed sampling of two ADC modules to increase the effective sampling rate. For concurrent sampling, the PHASE field of each ADC module must be the same and the sample and hold times (TSHn) for the matching sample steps of each ADC must be the same. For example, both ADC0 and ADC1 would program PHASE = 0x0 in the **ADCSPC** register and might both have the following configuration for their **ADCSSTSH0** register:

- TSH7=0x4
- TSH6=0x2
- TSH5=0x2
- TSH4=0x8
- TSH3=0x6
- TSH2=0x2
- TSH1=0x4
- TSH0=0x2

For skewed sampling with a consistent phase lag, the TSHn field in the **ADCSSTSHn** register must be the same for all sample steps of an ADC and for both ADC Modules. The desired lag can be calculated by adding the sample and hold time (TSHn) to the twelve clock conversion time to determine the total number of clocks in a sample period. For example to create a 180.0° phase lag, the PHASE of the lagging ADC is calculated as:

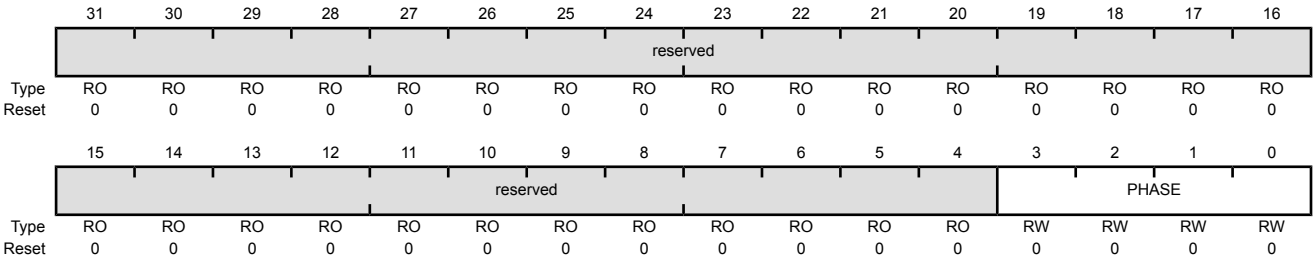
PHASE = (TSHn+ 12)/2, where TSHn is in ADC_Clocks

For situations where a predictable phase lag is not required, sample and hold times (TSHn) of ADC modules can vary.

**Note:** Care should be taken when the PHASE field is non-zero, as the resulting delay in sampling the AINx input may result in undesirable system consequences. The time from ADC trigger to sample is increased and could make the response time longer than anticipated. The added latency could have ramifications in the system design. Designers should carefully consider the impact of this delay.

## ADC Sample Phase Control (ADCSPC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x024
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | PHASE | RW | 0x0 | Phase Lag<br>This field selects the sample phase lag from the standard sample time. |

| Value | Description |
|---|---|
| 0x0 | The ADC samples are concurrent. |
| 0x1 | The ADC sample lags by 1 ADC clock |
| 0x2 | The ADC sample lags by 2 ADC clocks |
| 0x3 | The ADC sample lags by 3 ADC clocks |
| 0x4 | The ADC sample lags by 4 clocks |
| 0x5 | The ADC sample lags by 5 clocks |
| 0x6 | The ADC sample lags by 6 clocks |
| 0x7 | The ADC sample lags by 7 clocks |
| 0x8 | The ADC sample lags by 8 clocks |
| 0x9 | The ADC sample lags by 9 clocks |
| 0xA | The ADC sample lags by 10 clocks |
| 0xB | The ADC sample lags by 11 clocks |
| 0xC | The ADC sample lags by 12 clocks |
| 0xD | The ADC sample lags by 13 clocks |
| 0xE | The ADC sample lags by 14 clocks |
| 0xF | The ADC sample lags by 15 clocks |

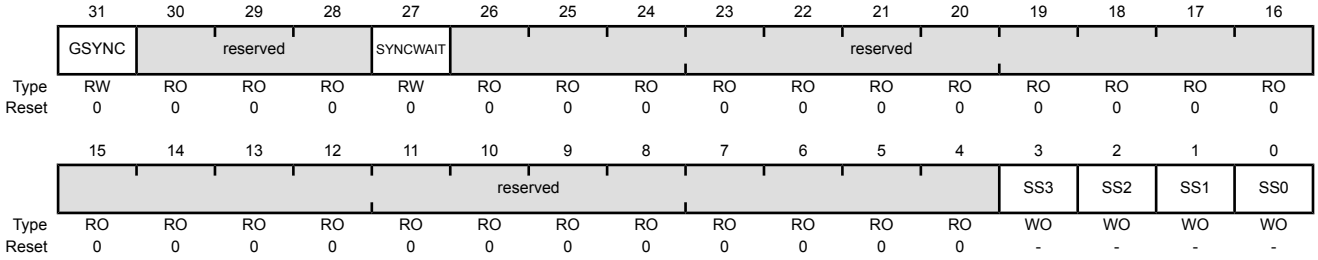
## Register 11: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

This register also provides a means to configure and then initiate concurrent sampling on all ADC modules. To do this, the first ADC module should be configured. The **ADCPSSI** register for that module should then be written. The appropriate SS bits should be set along with the SYNCWAIT bit. Additional ADC modules should then be configured following the same procedure. Once the final ADC module is configured, its **ADCPSSI** register should be written with the appropriate SS bits set along with the GSYNC bit. All of the ADC modules then begin concurrent sampling according to their configuration.

ADC Processor Sample Sequence Initiate (ADCPSSI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x028
Type RW, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GSYNC | RW | 0 | Global Synchronize<br>Value Description<br>0 This bit is cleared once sampling has been initiated.<br>1 This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SSn bit and the SYNCWAIT bit starts sampling once this bit is written. |
| 30:28 | reserved | RO | 0x0 | 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. |
| 27 | SYNCWAIT | RW | 0 | Synchronize Wait<br>Value Description<br>0 Sampling begins when a sample sequence has been initiated.<br>1 This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set. |
| 26:4 | reserved | RO | 0x0000.0 | 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. |

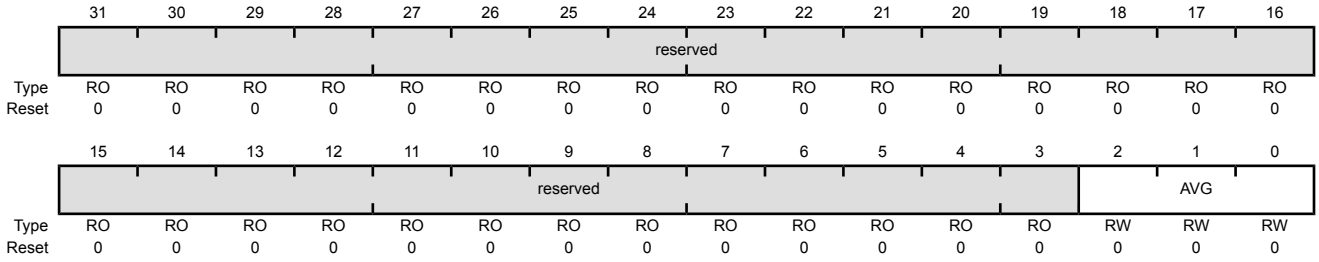
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | SS3 | WO | - | SS3 Initiate<br><br>Value Description<br>0 No effect.<br>1 Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the **ADCACTSS** register.<br><br>Only a write by software is valid; a read of this register returns no meaningful data. |
| 2 | SS2 | WO | - | SS2 Initiate<br><br>Value Description<br>0 No effect.<br>1 Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the **ADCACTSS** register.<br><br>Only a write by software is valid; a read of this register returns no meaningful data. |
| 1 | SS1 | WO | - | SS1 Initiate<br><br>Value Description<br>0 No effect.<br>1 Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the **ADCACTSS** register.<br><br>Only a write by software is valid; a read of this register returns no meaningful data. |
| 0 | SS0 | WO | - | SS0 Initiate<br><br>Value Description<br>0 No effect.<br>1 Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the **ADCACTSS** register.<br><br>Only a write by software is valid; a read of this register returns no meaningful data. |

## Register 12: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from $2^{AVG}$ consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG=7 provides unpredictable results.

ADC Sample Averaging Control (ADCSAC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x030
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | 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. |
| 2:0 | AVG | RW | 0x0 | Hardware Averaging Control<br>Specifies the amount of hardware averaging that will be applied to ADC samples. The AVG field can be any value between 0 and 6. Entering a value of 7 creates unpredictable results. |

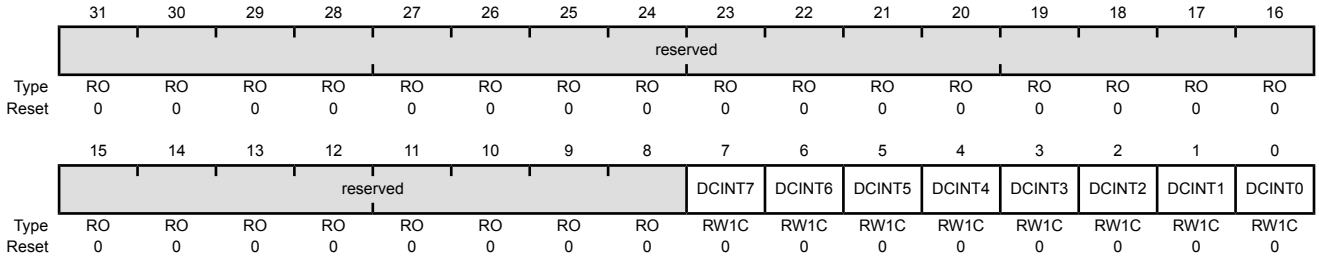
| Value | Description |
|---|---|
| 0x0 | No hardware oversampling |
| 0x1 | 2x hardware oversampling |
| 0x2 | 4x hardware oversampling |
| 0x3 | 8x hardware oversampling |
| 0x4 | 16x hardware oversampling |
| 0x5 | 32x hardware oversampling |
| 0x6 | 64x hardware oversampling |
| 0x7 | reserved |

## Register 13: ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034

This register provides status and acknowledgement of digital comparator interrupts. One bit is provided for each comparator.

ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x034
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | DCINT7 | RW1C | 0 | Digital Comparator 7 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 7 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |
| 6 | DCINT6 | RW1C | 0 | Digital Comparator 6 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 6 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |
| 5 | DCINT5 | RW1C | 0 | Digital Comparator 5 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 5 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |

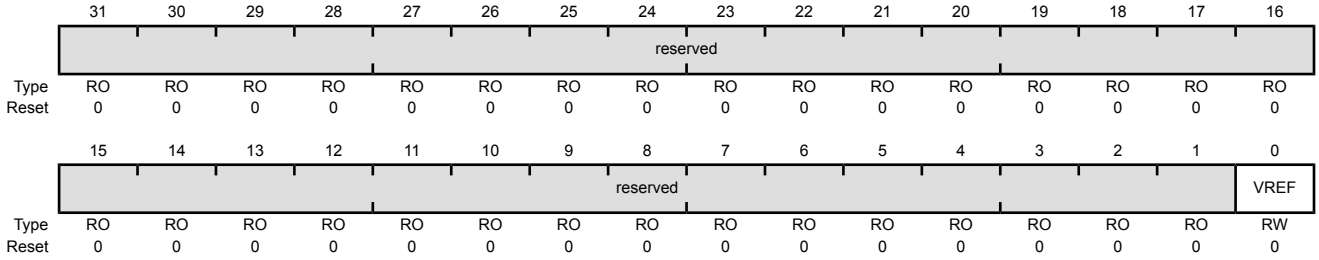
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | DCINT4 | RW1C | 0 | Digital Comparator 4 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 4 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |
| 3 | DCINT3 | RW1C | 0 | Digital Comparator 3 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 3 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |
| 2 | DCINT2 | RW1C | 0 | Digital Comparator 2 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 2 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |
| 1 | DCINT1 | RW1C | 0 | Digital Comparator 1 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 1 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |
| 0 | DCINT0 | RW1C | 0 | Digital Comparator 0 Interrupt Status and Clear<br><br>Value Description<br>0 No interrupt.<br>1 Digital Comparator 0 has generated an interrupt.<br><br>This bit is cleared by writing a 1. |

## Register 14: ADC Control (ADCCTL), offset 0x038

This register configures the voltage reference. The voltage references for the conversion can be VREFA+ and VREFA- or VDDA and GNDA. Note that values set in this register apply to all ADC modules, it is not possible to set one module to use internal references and another to use external references.

ADC Control (ADCCTL)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x038
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | VREF | RW | 0x0 | Voltage Reference Select |

| Value | Description |
|---|---|
| 0x0 | VDDA and GNDA are the voltage references for all ADC modules. |
| 0x1 | The external VREFA+ and VREFA- inputs are the voltage references for all ADC modules. |

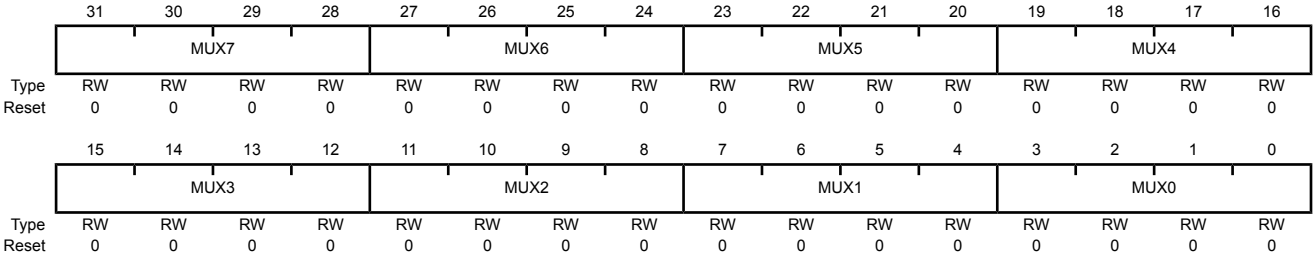
## Register 15: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register, along with the **ADCSSEMUX0** register, defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. If the corresponding EMUXn bit in the **ADCSSEMUX0** register is set, the MUXn field in this register selects from AIN[23:16]. When the corresponding EMUXn bit is clear, the MUXn field selects from AIN[15:0]. This register is 32 bits wide and contains information for eight possible samples.

**Note:** Channels AIN[31:24] do not exist on this microcontroller. Configuring MUXn to be 0x8-0xF when the corresponding EMUXn bit is set results in undefined behavior.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x040
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | MUX7 | RW | 0x0 | 8th Sample Input Select<br>The MUX7 field is used during the eighth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 0x1 when EMUX7 is clear indicates the input is AIN1. A value of 0x1 when EMUX7 is set indicates the input is AIN17.<br>If differential sampling is enabled (the D7 bit in the **ADCSSCTL0** register is set), this field must be set to the pair number "i", where the paired inputs are "2i and 2i+1". |
| 27:24 | MUX6 | RW | 0x0 | 7th Sample Input Select<br>The MUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |
| 23:20 | MUX5 | RW | 0x0 | 6th Sample Input Select<br>The MUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |
| 19:16 | MUX4 | RW | 0x0 | 5th Sample Input Select<br>The MUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:12 | MUX3 | RW | 0x0 | 4th Sample Input Select<br>The MUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |
| 11:8 | MUX2 | RW | 0x0 | 3rd Sample Input Select<br>The MUX2 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |
| 7:4 | MUX1 | RW | 0x0 | 2nd Sample Input Select<br>The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |
| 3:0 | MUX0 | RW | 0x0 | 1st Sample Input Select<br>The MUX0 field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |

## Register 16: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, eighth sample, or any sample in between. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Control 0 (ADCSSCTL0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x044
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TS7 | IE7 | END7 | D7 | TS6 | IE6 | END6 | D6 | TS5 | IE5 | END5 | D5 | TS4 | IE4 | END4 | D4 |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TS3 | IE3 | END3 | D3 | TS2 | IE2 | END2 | D2 | TS1 | IE1 | END1 | D1 | TS0 | IE0 | END0 | D0 |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | TS7 | RW | 0 | 8th Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the eighth sample of the sample sequence.<br>1 The temperature sensor is read during the eighth sample of the sample sequence. |
| 30 | IE7 | RW | 0 | 8th Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the eighth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 29 | END7 | RW | 0 | 8th Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The eighth sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 28 | D7 | RW | 0 | 8th Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS7 bit is set. |
| 27 | TS6 | RW | 0 | 7th Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the seventh sample of the sample sequence.<br>1 The temperature sensor is read during the seventh sample of the sample sequence. |
| 26 | IE6 | RW | 0 | 7th Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the seventh sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 25 | END6 | RW | 0 | 7th Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The seventh sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 24 | D6 | RW | 0 | 7th Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS6 bit is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23 | TS5 | RW | 0 | 6th Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the sixth sample of the sample sequence.<br>1 The temperature sensor is read during the sixth sample of the sample sequence. |
| 22 | IE5 | RW | 0 | 6th Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the sixth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 21 | END5 | RW | 0 | 6th Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The sixth sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 20 | D5 | RW | 0 | 6th Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS5 bit is set. |
| 19 | TS4 | RW | 0 | 5th Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the fifth sample of the sample sequence.<br>1 The temperature sensor is read during the fifth sample of the sample sequence. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 18 | IE4 | RW | 0 | 5th Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the fifth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 17 | END4 | RW | 0 | 5th Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The fifth sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 16 | D4 | RW | 0 | 5th Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS4 bit is set. |
| 15 | TS3 | RW | 0 | 4th Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the fourth sample of the sample sequence.<br>1 The temperature sensor is read during the fourth sample of the sample sequence. |
| 14 | IE3 | RW | 0 | 4th Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | END3 | RW | 0 | 4th Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The fourth sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 12 | D3 | RW | 0 | 4th Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS3 bit is set. |
| 11 | TS2 | RW | 0 | 3rd Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the third sample of the sample sequence.<br>1 The temperature sensor is read during the third sample of the sample sequence. |
| 10 | IE2 | RW | 0 | 3rd Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the third sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 9 | END2 | RW | 0 | 3rd Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The third sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | D2 | RW | 0 | 3rd Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS2 bit is set. |
| 7 | TS1 | RW | 0 | 2nd Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the second sample of the sample sequence.<br>1 The temperature sensor is read during the second sample of the sample sequence. |
| 6 | IE1 | RW | 0 | 2nd Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the second sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 5 | END1 | RW | 0 | 2nd Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The second sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 4 | D1 | RW | 0 | 2nd Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS1 bit is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | TS0 | RW | 0 | 1st Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the first sample of the sample sequence.<br>1 The temperature sensor is read during the first sample of the sample sequence. |
| 2 | IE0 | RW | 0 | 1st Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the first sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 1 | END0 | RW | 0 | 1st Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The first sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 0 | D0 | RW | 0 | 1st Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set. |

## Register 17: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

## Register 18: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

## Register 19: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

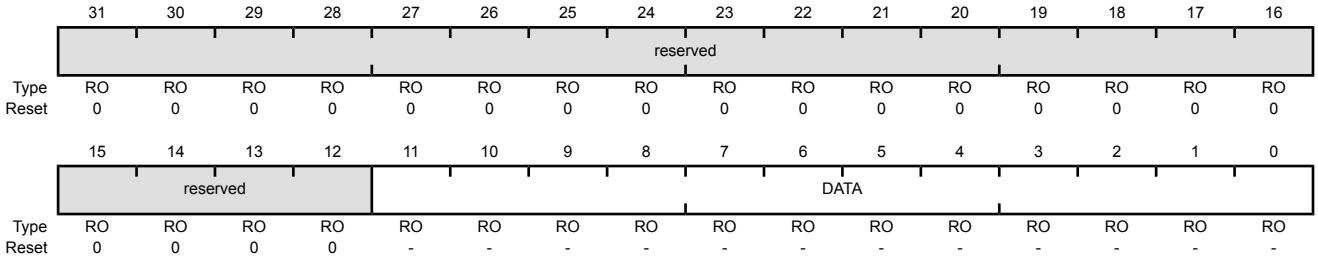
## Register 20: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

**Important:** This register is read-sensitive. See the register description for details.

This register contains the conversion results for samples collected with the sample sequencer (the **ADCSSFIFO0** register is used for Sample Sequencer 0, **ADCSSFIFO1** for Sequencer 1, **ADCSSFIFO2** for Sequencer 2, and **ADCSSFIFO3** for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

ADC Sample Sequence Result FIFO n (ADCSSFIFOn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x048
Type RO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |
| 11:0 | DATA | RO | - | Conversion Result Data |

## Register 21: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

## Register 22: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

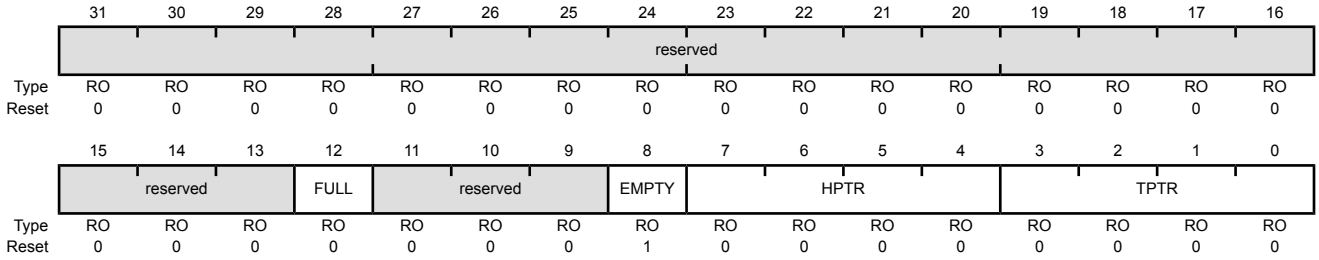
## Register 23: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

## Register 24: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the sample sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO with the head and tail pointers both pointing to index 0. The **ADCSSFSTAT0** register provides status on FIFO0, which has 8 entries; **ADCSSFSTAT1** on FIFO1, which has 4 entries; **ADCSSFSTAT2** on FIFO2, which has 4 entries; and **ADCSSFSTAT3** on FIFO3 which has a single entry.

ADC Sample Sequence FIFO n Status (ADCSSFSTATn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x04C
Type RO, reset 0x0000.0100




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:13 | reserved | RO | 0x0000.0 | 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. |
| 12 | FULL | RO | 0 | FIFO Full<br>Value Description<br>0 The FIFO is not currently full.<br>1 The FIFO is currently full. |
| 11:9 | reserved | RO | 0x0 | 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. |
| 8 | EMPTY | RO | 1 | FIFO Empty<br>Value Description<br>0 The FIFO is not currently empty.<br>1 The FIFO is currently empty. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:4 | HPTR | RO | 0x0 | FIFO Head Pointer<br>This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.<br>Valid values are 0x0-0x7 for FIFO0; 0x0-0x3 for FIFO1 and FIFO2; and 0x0 for FIFO3. |
| 3:0 | TPTR | RO | 0x0 | FIFO Tail Pointer<br>This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.<br>Valid values are 0x0-0x7 for FIFO0; 0x0-0x3 for FIFO1 and FIFO2; and 0x0 for FIFO3. |

## Register 25: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

ADC Sample Sequence 0 Operation (ADCSSOP0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x050
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0x0 | 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. |
| 28 | S7DCOP | RW | 0 | Sample 7 Digital Comparator Operation<br>Value Description<br>0 The eighth sample is saved in Sample Sequence FIFO0.<br>1 The eighth sample is sent to the digital comparator unit specified by the S7DCSEL bit in the **ADCSSDC0** register, and the value is not written to the FIFO. |
| 27:25 | reserved | RO | 0x0 | 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. |
| 24 | S6DCOP | RW | 0 | Sample 6 Digital Comparator Operation<br>Same definition as S7DC0P but used during the seventh sample. |
| 23:21 | reserved | RO | 0x0 | 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. |
| 20 | S5DCOP | RW | 0 | Sample 5 Digital Comparator Operation<br>Same definition as S7DC0P but used during the sixth sample. |
| 19:17 | reserved | RO | 0x0 | 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. |
| 16 | S4DCOP | RW | 0 | Sample 4 Digital Comparator Operation<br>Same definition as S7DC0P but used during the fifth sample. |
| 15:13 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | S3DCOP | RW | 0 | Sample 3 Digital Comparator Operation<br>Same definition as S7DCOP but used during the fourth sample. |
| 11:9 | reserved | RO | 0x0 | 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. |
| 8 | S2DCOP | RW | 0 | Sample 2 Digital Comparator Operation<br>Same definition as S7DCOP but used during the third sample. |
| 7:5 | reserved | RO | 0x0 | 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. |
| 4 | S1DCOP | RW | 0 | Sample 1 Digital Comparator Operation<br>Same definition as S7DCOP but used during the second sample. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | S0DCOP | RW | 0 | Sample 0 Digital Comparator Operation<br>Same definition as S7DCOP but used during the first sample. |

## Register 26: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding SnDC0P bit in the **ADCSSOP0** register is set.

ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x054
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S7DCSEL | | | | S6DCSEL | | | | S5DCSEL | | | | S4DCSEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | S7DCSEL | RW | 0x0 | Sample 7 Digital Comparator Select<br>When the S7DC0P bit in the **ADCSSOP0** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer 0.<br>**Note:** Values not listed are reserved.<br>Value Description<br>0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCDCCTL0**)<br>0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCDCCTL1**)<br>0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCDCCTL2**)<br>0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCDCCTL3**)<br>0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCDCCTL4**)<br>0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCDCCTL5**)<br>0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCDCCTL6**)<br>0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCDCCTL7**) |
| 27:24 | S6DCSEL | RW | 0x0 | Sample 6 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the seventh sample. |
| 23:20 | S5DCSEL | RW | 0x0 | Sample 5 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the sixth sample. |
| 19:16 | S4DCSEL | RW | 0x0 | Sample 4 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the fifth sample. |
| 15:12 | S3DCSEL | RW | 0x0 | Sample 3 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the fourth sample. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:8 | S2DCSEL | RW | 0x0 | Sample 2 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the third sample. |
| 7:4 | S1DCSEL | RW | 0x0 | Sample 1 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the second sample. |
| 3:0 | S0DCSEL | RW | 0x0 | Sample 0 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the first sample. |

## Register 27: ADC Sample Sequence Extended Input Multiplexer Select 0 (ADCSSEMUX0), offset 0x058

This register, along with the **ADCSSMUX0** register, defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. If a bit in this register is set, the corresponding MUXn field in the **ADCSSMUX0** register selects from AIN[23:16]. When a bit in this register is clear, the corresponding MUXn field selects from AIN[15:0]. This register is 32 bits wide and contains information for eight possible samples.

Note that this register is not used when the differential channel designation is used (the Dn bit is set in the **ADCSSCTL0** register) because the **ADCSSMUX0** register can select all the available pairs.

ADC Sample Sequence Extended Input Multiplexer Select 0 (ADCSSEMUX0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x058
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | EMUX7 | reserved | | | EMUX6 | reserved | | | EMUX5 | reserved | | | EMUX4 |
| Type | RO | RO | RO | RW | RO | RO | RO | RW | RO | RO | RO | RW | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | EMUX3 | reserved | | | EMUX2 | reserved | | | EMUX1 | reserved | | | EMUX0 |
| Type | RO | RO | RO | RW | RO | RO | RO | RW | RO | RO | RO | RW | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0x0 | 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. |
| 28 | EMUX7 | RW | 0x0 | 8th Sample Input Select (Upper Bit)<br>The EMUX7 field is used during the eighth sample of a sequence executed with the sample sequencer.<br><br>Value Description<br>0 The eighth sample input is selected from AIN[15:0] using the **ADCSSMUX0** register. For example, if the MUX7 field is 0x0, AIN0 is selected.<br>1 The eighth sample input is selected from AIN[23:16] using the **ADCSSMUX0** register. For example, if the MUX7 field is 0x0, AIN16 is selected. |
| 27:25 | reserved | RO | 0x0 | 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. |
| 24 | EMUX6 | RW | 0x0 | 7th Sample Input Select (Upper Bit)<br>The EMUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7. |
| 23:21 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 20 | EMUX5 | RW | 0x0 | 6th Sample Input Select (Upper Bit)<br>The EMUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7. |
| 19:17 | reserved | RO | 0x0 | 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. |
| 16 | EMUX4 | RW | 0x0 | 5th Sample Input Select (Upper Bit)<br>The EMUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7. |
| 15:13 | reserved | RO | 0x0 | 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. |
| 12 | EMUX3 | RW | 0x0 | 4th Sample Input Select (Upper Bit)<br>The EMUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7. |
| 11:9 | reserved | RO | 0x0 | 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. |
| 8 | EMUX2 | RW | 0x0 | 3rd Sample Input Select (Upper Bit)<br>The EMUX2 field is used during the third sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7. |
| 7:5 | reserved | RO | 0x0 | 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. |
| 4 | EMUX1 | RW | 0x0 | 2th Sample Input Select (Upper Bit)<br>The EMUX1 field is used during the second sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | EMUX0 | RW | 0x0 | 1st Sample Input Select (Upper Bit)<br>The EMUX0 field is used during the first sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX7. |

## Register 28: ADC Sample Sequence 0 Sample and Hold Time (ADCSSTSH0), offset 0x05C

This register controls the sample period size for each sample of sequencer 0. Each sample and hold period select specifies the time allocated to the sample and hold circuit as shown by the encodings in Table 18-3 on page 1209.

**Note:** If sampling the internal temperature sensor, the sample and hold width should be at least 16 ADC clocks (TSHn = 0x4).

**Table 18-8. Sample and Hold Width in ADC Clocks**

| TSHn Encoding | $N_{SH}$ |
|---|---|
| 0x0 | 4 |
| 0x1 | reserved |
| 0x2 | 8 |
| 0x3 | reserved |
| 0x4 | 16 |
| 0x5 | reserved |
| 0x6 | 32 |
| 0x7 | reserved |
| 0x8 | 64 |
| 0x9 | reserved |
| 0xA | 128 |
| 0xB | reserved |
| 0xC | 256 |
| 0xD-0xF | reserved |

ADC Sample Sequence 0 Sample and Hold Time (ADCSSTSH0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x05C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSH7 | | | | TSH6 | | | | TSH5 | | | | TSH4 | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSH3 | | | | TSH2 | | | | TSH1 | | | | TSH0 | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | TSH7 | RW | 0x0 | 8th Sample and Hold Period Select<br>The TSH7 field is used during the eighth sample of a sequence executed with the sample sequencer. |
| 27:24 | TSH6 | RW | 0x0 | 7th Sample and Hold Period Select<br>The TSH6 field is used during the seventh sample of a sequence executed with the sample sequencer. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23:20 | TSH5 | RW | 0x0 | 6th Sample and Hold Period Select<br>The TSH5 field is used during the sixth sample of a sequence executed with the sample sequencer. |
| 19:16 | TSH4 | RW | 0x0 | 5th Sample and Hold Period Select<br>The TSH4 field is used during the fifth sample of a sequence executed with the sample sequencer. |
| 15:12 | TSH3 | RW | 0x0 | 4th Sample and Hold Period Select<br>The TSH3 field is used during the fourth sample of a sequence executed with the sample sequencer. |
| 11:8 | TSH2 | RW | 0x0 | 3rd Sample and Hold Period Select<br>The TSH2 field is used during the third sample of a sequence executed with the sample sequencer. |
| 7:4 | TSH1 | RW | 0x0 | 2nd Sample and Hold Period Select<br>The TSH1 field is used during the second sample of a sequence executed with the sample sequencer. |
| 3:0 | TSH0 | RW | 0x0 | 1st Sample and Hold Period Select<br>The TSH0 field is used during the first sample of a sequence executed with the sample sequencer. |

## Register 29: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

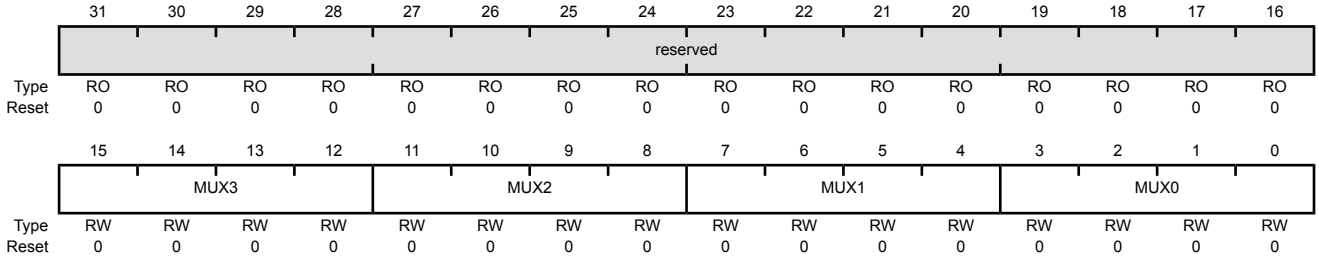
## Register 30: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register, along with the **ADCSSEMUX1** or **ADCSSEMUX2** register, defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. If the corresponding EMUXn bit in the **ADCSSEMUX1** or **ADCSSEMUX2** register is set, the MUXn field in this register selects from AIN[23:16]. When the corresponding EMUXn bit is clear, the MUXn field selects from AIN[15:0]. These registers are 16 bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 1260 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

**Note:** Channels AIN[31:24] do not exist on this microcontroller. Configuring MUXn to be 0x8-0xF when the corresponding EMUXn bit is set results in undefined behavior.

ADC Sample Sequence Input Multiplexer Select n (ADCSSMUXn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x060
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:12 | MUX3 | RW | 0x0 | 4th Sample Input Select |
| 11:8 | MUX2 | RW | 0x0 | 3rd Sample Input Select |
| 7:4 | MUX1 | RW | 0x0 | 2nd Sample Input Select |
| 3:0 | MUX0 | RW | 0x0 | 1st Sample Input Select |

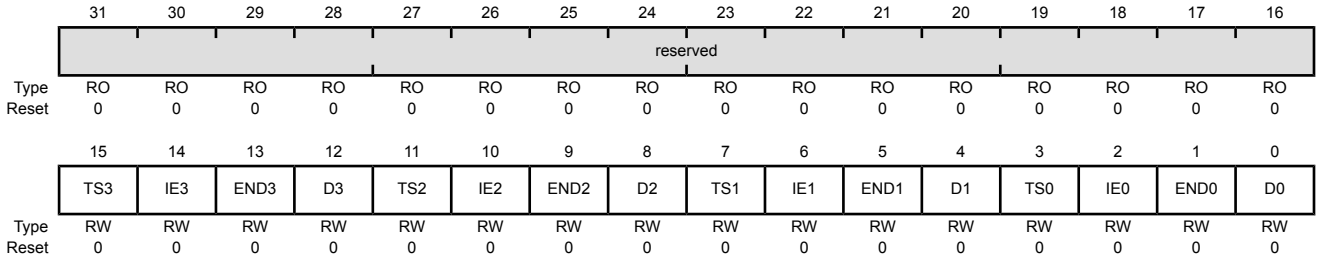
## Register 31: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

## Register 32: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 1262 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control n (ADCSSCTLn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x064
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15 | TS3 | RW | 0 | 4th Sample Temp Sensor Select<br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the fourth sample of the sample sequence.<br>1 The temperature sensor is read during the fourth sample of the sample sequence. |
| 14 | IE3 | RW | 0 | 4th Sample Interrupt Enable<br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the fourth sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br>It is legal to have multiple samples within a sequence generate interrupts. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | END3 | RW | 0 | 4th Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The fourth sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 12 | D3 | RW | 0 | 4th Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS3 bit is set. |
| 11 | TS2 | RW | 0 | 3rd Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the third sample of the sample sequence.<br>1 The temperature sensor is read during the third sample of the sample sequence. |
| 10 | IE2 | RW | 0 | 3rd Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the third sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 9 | END2 | RW | 0 | 3rd Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The third sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | D2 | RW | 0 | 3rd Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS2 bit is set. |
| 7 | TS1 | RW | 0 | 2nd Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the second sample of the sample sequence.<br>1 The temperature sensor is read during the second sample of the sample sequence. |
| 6 | IE1 | RW | 0 | 2nd Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the second sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 5 | END1 | RW | 0 | 2nd Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The second sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 4 | D1 | RW | 0 | 2nd Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS1 bit is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | TS0 | RW | 0 | 1st Sample Temp Sensor Select<br><br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the first sample of the sample sequence.<br>1 The temperature sensor is read during the first sample of the sample sequence. |
| 2 | IE0 | RW | 0 | 1st Sample Interrupt Enable<br><br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of the first sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br><br>It is legal to have multiple samples within a sequence generate interrupts. |
| 1 | END0 | RW | 0 | 1st Sample is End of Sequence<br><br>Value Description<br>0 Another sample in the sequence is the final sample.<br>1 The first sample is the last sample of the sequence.<br><br>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero. |
| 0 | D0 | RW | 0 | 1st Sample Differential Input Select<br><br>Value Description<br>0 The analog inputs are not differentially sampled.<br>1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".<br><br>Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set. |

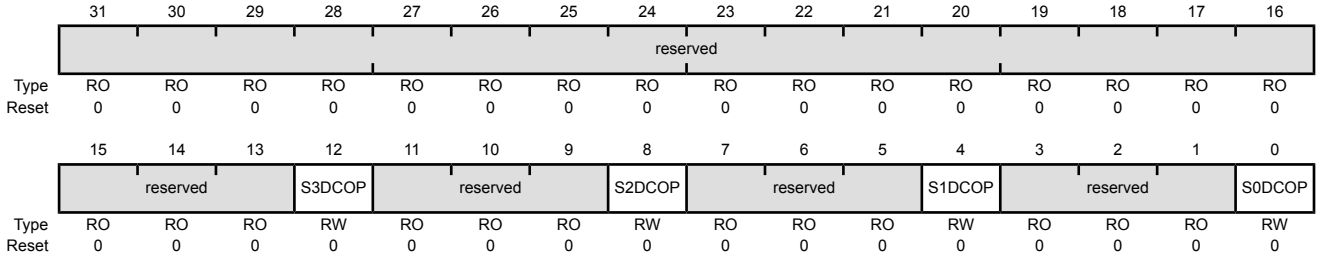
## Register 33: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070

## Register 34: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090

This register determines whether the sample from the given conversion on Sample Sequence n is saved in the Sample Sequence n FIFO or sent to the digital comparator unit. The **ADCSSOP1** register controls Sample Sequencer 1 and the **ADCSSOP2** register controls Sample Sequencer 2.

ADC Sample Sequence n Operation (ADCSSOPn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x070
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:13 | reserved | RO | 0x0000.0 | 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. |
| 12 | S3DCOP | RW | 0 | Sample 3 Digital Comparator Operation<br>Value Description<br>0 The fourth sample is saved in Sample Sequence FIFOn.<br>1 The fourth sample is sent to the digital comparator unit specified by the S3DCSEL bit in the **ADCSSDC0n** register, and the value is not written to the FIFO. |
| 11:9 | reserved | RO | 0x0 | 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. |
| 8 | S2DCOP | RW | 0 | Sample 2 Digital Comparator Operation<br>Same definition as S3DCOP but used during the third sample. |
| 7:5 | reserved | RO | 0x0 | 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. |
| 4 | S1DCOP | RW | 0 | Sample 1 Digital Comparator Operation<br>Same definition as S3DCOP but used during the second sample. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | S0DCOP | RW | 0 | Sample 0 Digital Comparator Operation<br>Same definition as S3DCOP but used during the first sample. |

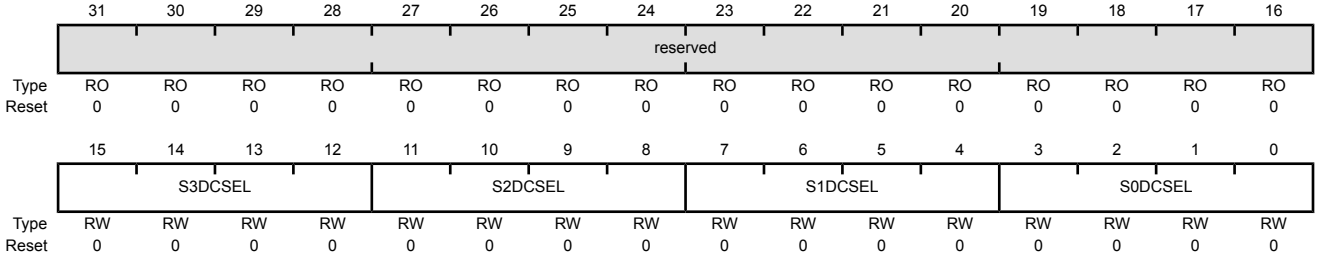
## Register 35: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074

## Register 36: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094

These registers determine which digital comparator receives the sample from the given conversion on Sample Sequence n if the corresponding SnDC0P bit in the **ADCSSOPn** register is set. The **ADCSSDC1** register controls the selection for Sample Sequencer 1 and the **ADCSSDC2** register controls the selection for Sample Sequencer 2.

ADC Sample Sequence n Digital Comparator Select (ADCSSDCn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x074
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:12 | S3DCSEL | RW | 0x0 | Sample 3 Digital Comparator Select<br>When the S3DC0P bit in the **ADCSSOPn** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer n.<br>**Note:** Values not listed are reserved.<br>Value Description<br>0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**)<br>0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**)<br>0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**)<br>0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**)<br>0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**)<br>0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**)<br>0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**)<br>0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**) |
| 11:8 | S2DCSEL | RW | 0x0 | Sample 2 Digital Comparator Select<br>This field has the same encodings as S3DCSEL but is used during the third sample. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:4 | S1DCSEL | RW | 0x0 | Sample 1 Digital Comparator Select<br>This field has the same encodings as S3DCSEL but is used during the second sample. |
| 3:0 | S0DCSEL | RW | 0x0 | Sample 0 Digital Comparator Select<br>This field has the same encodings as S3DCSEL but is used during the first sample. |

## Register 37: ADC Sample Sequence Extended Input Multiplexer Select 1 (ADCSSEMUX1), offset 0x078

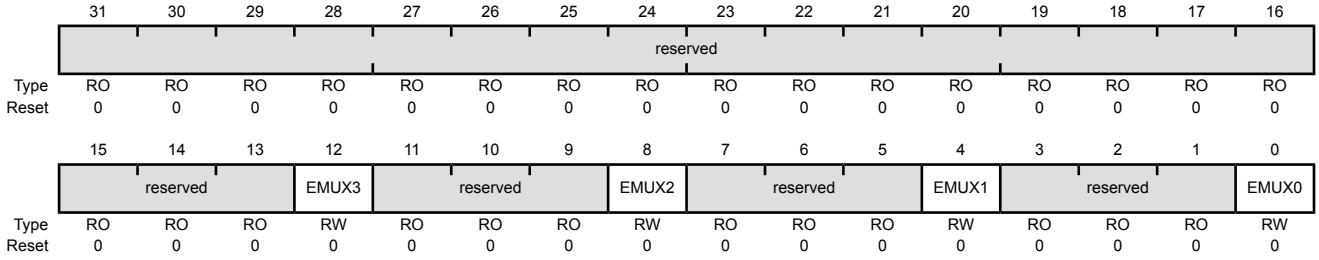
## Register 38: ADC Sample Sequence Extended Input Multiplexer Select 2 (ADCSSEMUX2), offset 0x098

This register, along with the **ADCSSMUX1** or **ADCSSMUX2** register, defines the analog input configuration for each sample in a sequence executed with either Sample Sequencer 1 or 2. If a bit in this register is set, the corresponding MUXn field in the **ADCSSMUX1** or **ADCSSMUX2** register selects from AIN[23:16]. When a bit in this register is clear, the corresponding MUXn field selects from AIN[15:0]. This register is 16 bits wide and contains information for four possible samples. The **ADCSSEMUX1** register controls Sample Sequencer 1 and the **ADCSSEMUX2** register controls Sample Sequencer 2.

Note that this register is not used when the differential channel designation is used (the Dn bit is set in the **ADCSSCTL1** or **ADCSSCTL2** register) because the **ADCSSMUX1** or **ADCSSMUX2** register can select all the available pairs.

ADC Sample Sequence Extended Input Multiplexer Select n (ADCSSEMUXn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x078
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:13 | reserved | RO | 0x0000 | 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. |
| 12 | EMUX3 | RW | 0x0 | 4th Sample Input Select (Upper Bit)<br>The EMUX3 field is used during the fourth sample of a sequence executed with the sample sequencer.<br><br>Value Description<br>0 The fourth sample input is selected from AIN[15:0] using the **ADCSSMUX1** or **ADCSSMUX2** register. For example, if the MUX3 field is 0x0, AIN0 is selected.<br>1 The fourth sample input is selected from AIN[23:16] using the **ADCSSMUX1** or **ADCSSMUX2** register. For example, if the MUX3 field is 0x0, AIN16 is selected. |
| 11:9 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | EMUX2 | RW | 0x0 | 3rd Sample Input Select (Upper Bit)<br>The EMUX2 field is used during the third sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX3. |
| 7:5 | reserved | RO | 0x0 | 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. |
| 4 | EMUX1 | RW | 0x0 | 2th Sample Input Select (Upper Bit)<br>The EMUX1 field is used during the second sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX3. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | EMUX0 | RW | 0x0 | 1st Sample Input Select (Upper Bit)<br>The EMUX0 field is used during the first sample of a sequence executed with the sample sequencer. This bit has the same description as EMUX3. |

## Register 39: ADC Sample Sequence 1 Sample and Hold Time (ADCSSTSH1), offset 0x07C

## Register 40: ADC Sample Sequence 2 Sample and Hold Time (ADCSSTSH2), offset 0x09C

These registers control the sample period size for each sample step of sequencer 1 and sequencer 2. Each sample and hold period select specifies the time allocated to the sample and hold circuit as shown by the encodings in Table 18-3 on page 1209.

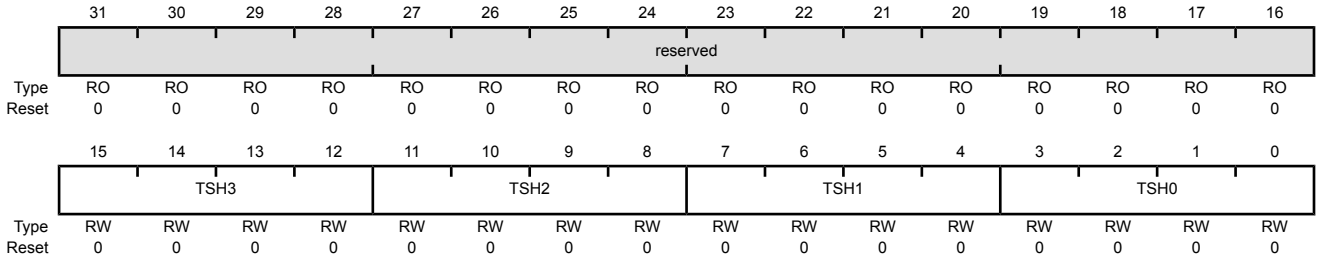
**Note:** If sampling the internal temperature sensor, the sample and hold width should be at least 16 ADC clocks (TSHn = 0x4).

**Table 18-9. Sample and Hold Width in ADC Clocks**

| TSHn Encoding | $N_{SH}$ |
|---|---|
| 0x0 | 4 |
| 0x1 | reserved |
| 0x2 | 8 |
| 0x3 | reserved |
| 0x4 | 16 |
| 0x5 | reserved |
| 0x6 | 32 |
| 0x7 | reserved |
| 0x8 | 64 |
| 0x9 | reserved |
| 0xA | 128 |
| 0xB | reserved |
| 0xC | 256 |
| 0xD-0xF | reserved |

ADC Sample Sequence n Sample and Hold Time (ADCSSTSHn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x07C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |

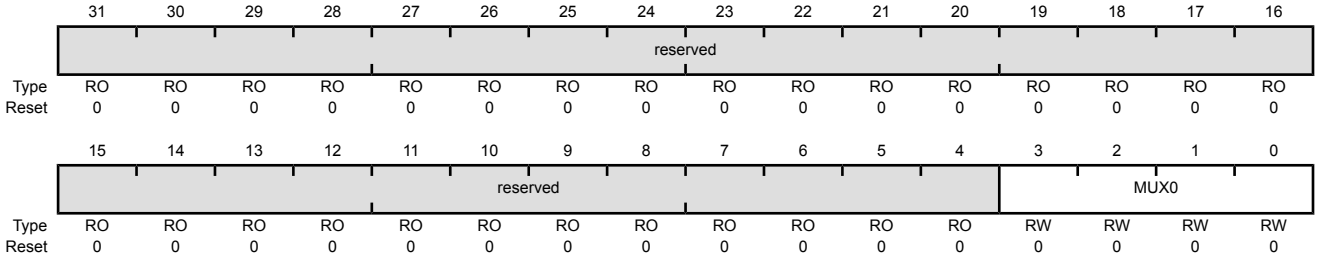
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:12 | TSH3 | RW | 0x0 | 4th Sample and Hold Period Select<br>The TSH3 field is used during the fourth sample of a sequence executed with the sample sequencer. |
| 11:8 | TSH2 | RW | 0x0 | 3rd Sample and Hold Period Select<br>The TSH2 field is used during the third sample of a sequence executed with the sample sequencer. |
| 7:4 | TSH1 | RW | 0x0 | 2nd Sample and Hold Period Select<br>The TSH1 field is used during the second sample of a sequence executed with the sample sequencer. |
| 3:0 | TSH0 | RW | 0x0 | 1st Sample and Hold Period Select<br>The TSH0 field is used during the first sample of a sequence executed with the sample sequencer. |

## Register 41: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register, along with the **ADCSSEMUX3** register, defines the analog input configuration for the sample in a sequence executed with Sample Sequencer 3. If the EMUX0 bit in the **ADCSSEMUX3** register is set, the MUX0 field in this register selects from AIN[23:16]. When the EMUX0 bit is clear, the MUX0 field selects from AIN[15:0]. This register is four bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 1260 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A0
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | MUX0 | RW | 0 | 1st Sample Input Select |

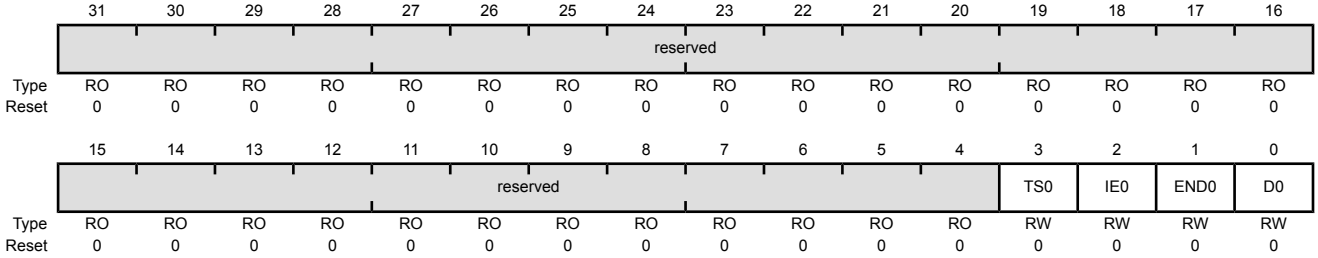
## Register 42: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 1262 for detailed bit descriptions.

**Note:** When configuring a sample sequence in this register, the END0 bit must be set.

ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A4
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | TS0 | RW | 0 | 1st Sample Temp Sensor Select<br>Value Description<br>0 The input pin specified by the **ADCSSMUXn** register is read during the first sample of the sample sequence.<br>1 The temperature sensor is read during the first sample of the sample sequence. |
| 2 | IE0 | RW | 0 | Sample Interrupt Enable<br>Value Description<br>0 The raw interrupt is not asserted to the interrupt controller.<br>1 The raw interrupt signal (INR0 bit) is asserted at the end of this sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller.<br>It is legal to have multiple samples within a sequence generate interrupts. |
| 1 | END0 | RW | 0 | End of Sequence<br>This bit must be set before initiating a single sample sequence.<br>Value Description<br>0 Sampling and conversion continues.<br>1 This is the end of sequence. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | D0 | RW | 0 | Sample Differential Input Select |

| Value | Description |
|---|---|
| 0 | The analog inputs are not differentially sampled. |
| 1 | The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". |

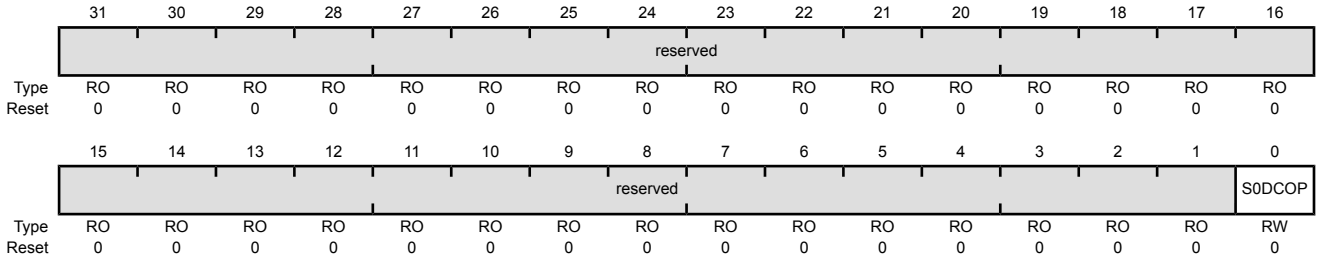
Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set.

## Register 43: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0

This register determines whether the sample from the given conversion on Sample Sequence 3 is saved in the Sample Sequence 3 FIFO or sent to the digital comparator unit.

ADC Sample Sequence 3 Operation (ADCSSOP3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0B0
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | S0DCOP | RW | 0 | Sample 0 Digital Comparator Operation |

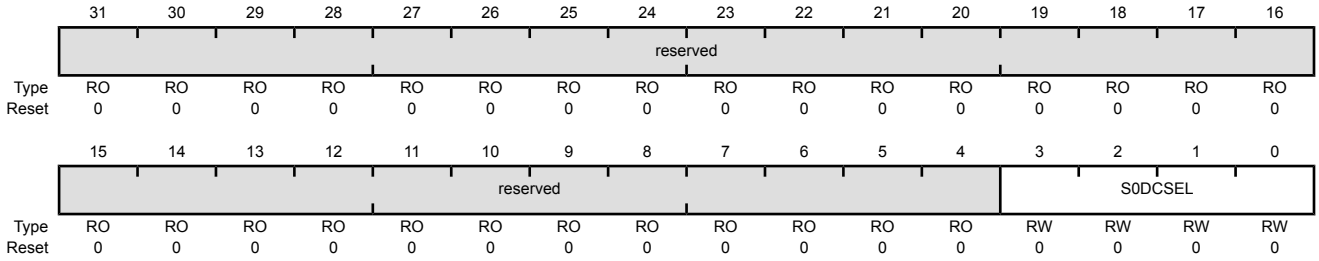
| Value | Description |
|---|---|
| 0 | The sample is saved in Sample Sequence FIFO3. |
| 1 | The sample is sent to the digital comparator unit specified by the S0DCSEL bit in the **ADCSSDC03** register, and the value is not written to the FIFO. |

## Register 44: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 3 if the corresponding SnDCOP bit in the **ADCSSOP3** register is set.

ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0B4
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | S0DCSEL | RW | 0x0 | Sample 0 Digital Comparator Select<br>When the S0DCOP bit in the **ADCSSOP3** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the sample from Sample Sequencer 3.<br>**Note:** Values not listed are reserved.<br>Value Description<br>0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**)<br>0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**)<br>0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**)<br>0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**)<br>0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**)<br>0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**)<br>0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**)<br>0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**) |

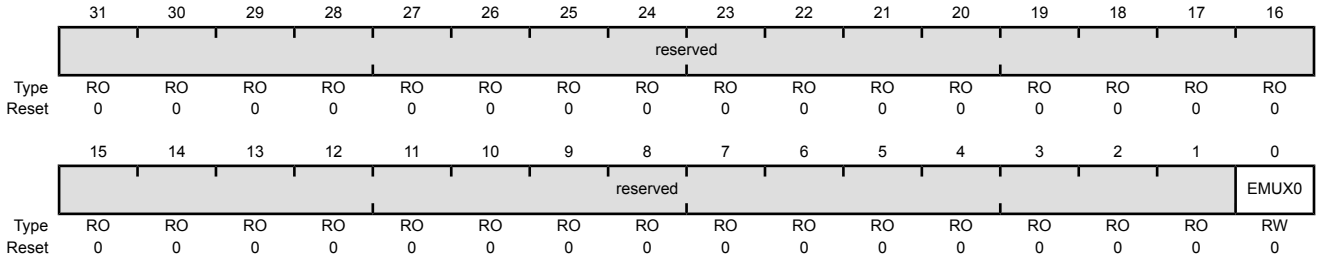
## Register 45: ADC Sample Sequence Extended Input Multiplexer Select 3 (ADCSSEMUX3), offset 0x0B8

This register, along with the **ADCSSMUX3** register, defines the analog input configuration for the sample in a sequence executed with Sample Sequencer 3. If EMUX0 is set, the MUX0 field in the **ADCSSMUX3** register selects from AIN[23:16]. When EMUX0 is clear, the MUX0 field selects from AIN[15:0]. This register is 1 bit wide and contains information for one possible sample.

Note that this register is not used when the differential channel designation is used (the Dn bit is set in the **ADCSSCTL3** register) because the **ADCSSMUX3** register can select all the available pairs.

ADC Sample Sequence Extended Input Multiplexer Select 3 (ADCSSEMUX3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0B8
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | EMUX0 | RW | 0x0 | 1st Sample Input Select (Upper Bit)<br>The EMUX0 field is used during the only sample of a sequence executed with the sample sequencer.<br><br>Value Description<br>0 The sample input is selected from AIN[15:0] using the **ADCSSMUX3** register. For example, if the MUX0 field is 0x0, AIN0 is selected.<br>1 The sample input is selected from AIN[23:16] using the **ADCSSMUX3** register. For example, if the MUX0 field is 0x0, AIN16 is selected. |

## Register 46: ADC Sample Sequence 3 Sample and Hold Time (ADCSSTSH3), offset 0x0BC

This register controls the sample period size for the sample in sequencer 3. The sample and hold period select specifies the time allocated to the sample and hold circuit as shown by the encodings in Table 18-3 on page 1209

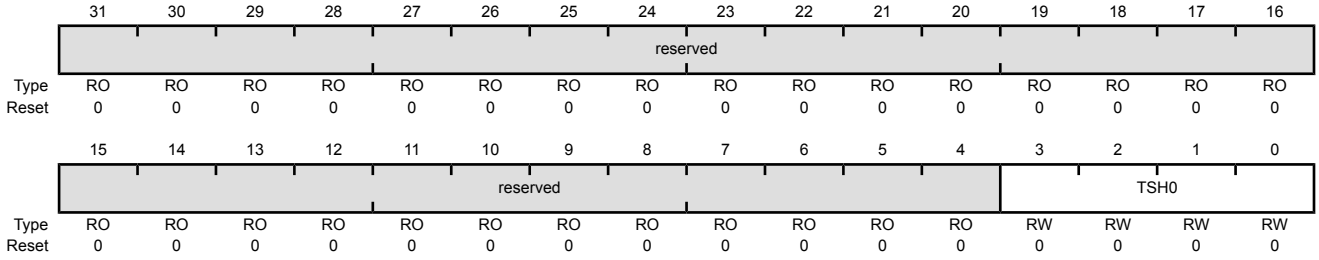
**Note:** If sampling the internal temperature sensor, the sample and hold width should be at least 16 ADC clocks (TSHn = 0x4).

**Table 18-10. Sample and Hold Width in ADC Clocks**

| TSHn Encoding | $N_{SH}$ |
|---|---|
| 0x0 | 4 |
| 0x1 | reserved |
| 0x2 | 8 |
| 0x3 | reserved |
| 0x4 | 16 |
| 0x5 | reserved |
| 0x6 | 32 |
| 0x7 | reserved |
| 0x8 | 64 |
| 0x9 | reserved |
| 0xA | 128 |
| 0xB | reserved |
| 0xC | 256 |
| 0xD-0xF | reserved |

ADC Sample Sequence 3 Sample and Hold Time (ADCSSTSH3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0BC
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0 | 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. |
| 3:0 | TSH0 | RW | 0x0 | 1st Sample and Hold Period Select<br>The TSH0 field is used during the first sample of a sequence executed with the sample sequencer. |

## Register 47: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00

This register provides the ability to reset any of the digital comparator interrupt or trigger functions back to their initial conditions. Resetting these functions ensures that the data that is being used by the interrupt and trigger functions in the digital comparator unit is not stale.

ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xD00
Type WO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | DCTRIG7 | DCTRIG6 | DCTRIG5 | DCTRIG4 | DCTRIG3 | DCTRIG2 | DCTRIG1 | DCTRIG0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | DCINT7 | DCINT6 | DCINT5 | DCINT4 | DCINT3 | DCINT2 | DCINT1 | DCINT0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x00 | 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. |
| 23 | DCTRIG7 | WO | 0 | Digital Comparator Trigger 7<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 7 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br><br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. After setting this bit, software should wait until the bit clears before continuing. |
| 22 | DCTRIG6 | WO | 0 | Digital Comparator Trigger 6<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 6 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br><br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 21 | DCTRIG5 | WO | 0 | Digital Comparator Trigger 5<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 5 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 20 | DCTRIG4 | WO | 0 | Digital Comparator Trigger 4<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 4 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 19 | DCTRIG3 | WO | 0 | Digital Comparator Trigger 3<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 3 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 18 | DCTRIG2 | WO | 0 | Digital Comparator Trigger 2<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 2 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | DCTRIG1 | WO | 0 | Digital Comparator Trigger 1<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 1 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 16 | DCTRIG0 | WO | 0 | Digital Comparator Trigger 0<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 0 trigger unit to its initial conditions.<br><br>When the trigger has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 15:8 | reserved | RO | 0x00 | 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. |
| 7 | DCINT7 | WO | 0 | Digital Comparator Interrupt 7<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 7 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 6 | DCINT6 | WO | 0 | Digital Comparator Interrupt 6<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 6 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | DCINT5 | WO | 0 | Digital Comparator Interrupt 5<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 5 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 4 | DCINT4 | WO | 0 | Digital Comparator Interrupt 4<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 4 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 3 | DCINT3 | WO | 0 | Digital Comparator Interrupt 3<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 3 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 2 | DCINT2 | WO | 0 | Digital Comparator Interrupt 2<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 2 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | DCINT1 | WO | 0 | Digital Comparator Interrupt 1<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 1 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br><br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |
| 0 | DCINT0 | WO | 0 | Digital Comparator Interrupt 0<br><br>Value Description<br>0 No effect.<br>1 Resets the Digital Comparator 0 interrupt unit to its initial conditions.<br><br>When the interrupt has been cleared, this bit is automatically cleared.<br><br>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used. |

## Register 48: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00

## Register 49: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04

## Register 50: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08

## Register 51: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C

## Register 52: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10

## Register 53: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14

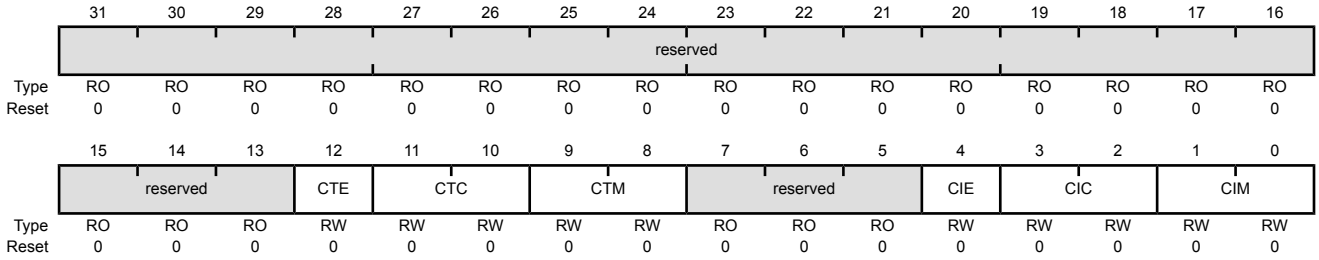
## Register 54: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18

## Register 55: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C

This register provides the comparison encodings that generate an interrupt and/or PWM trigger. See “Interrupt/ADC-Trigger Selector” on page 1931 for more information on using the ADC digital comparators to trigger a PWM generator.

ADC Digital Comparator Control n (ADCDCCTLn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE00
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:13 | reserved | RO | 0x0000.0 | 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. |
| 12 | CTE | RW | 0 | Comparison Trigger Enable |

| Value | Description |
|---|---|
| 0 | Disables the trigger function state machine. ADC conversion data is ignored by the trigger function. |
| 1 | Enables the trigger function state machine. The ADC conversion data is used to determine if a trigger should be generated according to the programming of the CTC and CTM fields. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:10 | CTC | RW | 0x0 | Comparison Trigger Condition<br>This field specifies the operational region in which a trigger is generated when the ADC conversion data is compared against the values of COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the **ADCDCCMPx** registers.<br>Value Description<br>0x0 Low Band<br>ADC Data < COMP0 ≤ COMP1<br>0x1 Mid Band<br>COMP0 < ADC Data ≤ COMP1<br>0x2 reserved<br>0x3 High Band<br>COMP0 ≤ COMP1 ≤ ADC Data |
| 9:8 | CTM | RW | 0x0 | Comparison Trigger Mode<br>This field specifies the mode by which the trigger comparison is made.<br>Value Description<br>0x0 Always<br>This mode generates a trigger every time the ADC conversion data falls within the selected operational region.<br>0x1 Once<br>This mode generates a trigger the first time that the ADC conversion data enters the selected operational region.<br>0x2 Hysteresis Always<br>This mode generates a trigger when the ADC conversion data falls within the selected operational region and continues to generate the trigger until the hysteresis condition is cleared by entering the opposite operational region.<br>Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.<br>0x3 Hysteresis Once<br>This mode generates a trigger the first time that the ADC conversion data falls within the selected operational region. No additional triggers are generated until the hysteresis condition is cleared by entering the opposite operational region.<br>Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3. |
| 7:5 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | CIE | RW | 0 | Comparison Interrupt Enable<br><br>Value Description<br>0 Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.<br>1 Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the CIC and CIM fields. |
| 3:2 | CIC | RW | 0x0 | Comparison Interrupt Condition<br><br>This field specifies the operational region in which an interrupt is generated when the ADC conversion data is compared against the values of COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the **ADCDCCMPx** registers.<br><br>Value Description<br>0x0 Low Band<br>ADC Data < COMP0 ≤ COMP1<br>0x1 Mid Band<br>COMP0 ≤ ADC Data < COMP1<br>0x2 reserved<br>0x3 High Band<br>COMP0 < COMP1 ≤ ADC Data |
| 1:0 | CIM | RW | 0x0 | Comparison Interrupt Mode<br><br>This field specifies the mode by which the interrupt comparison is made.<br><br>Value Description<br>0x0 Always<br>This mode generates an interrupt every time the ADC conversion data falls within the selected operational region.<br>0x1 Once<br>This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region.<br>0x2 Hysteresis Always<br>This mode generates an interrupt when the ADC conversion data falls within the selected operational region and continues to generate the interrupt until the hysteresis condition is cleared by entering the opposite operational region.<br>Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.<br>0x3 Hysteresis Once<br>This mode generates an interrupt the first time that the ADC conversion data falls within the selected operational region. No additional interrupts are generated until the hysteresis condition is cleared by entering the opposite operational region.<br>Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3. |

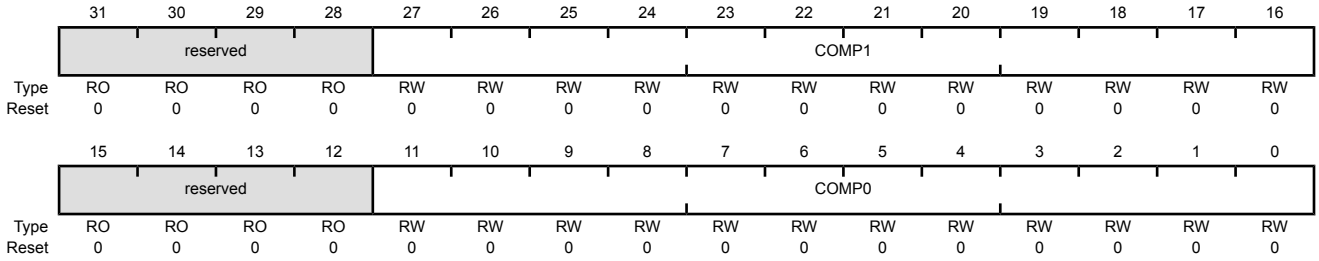
## Register 56: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40
## Register 57: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44
## Register 58: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48
## Register 59: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C
## Register 60: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50
## Register 61: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54
## Register 62: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58
## Register 63: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C

This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region.

**Note:** The value in the COMP1 field must be greater than or equal to the value in the COMP0 field or unexpected results can occur.

ADC Digital Comparator Range n (ADCDCCMPn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE40
Type RW, reset 0x0000.0000



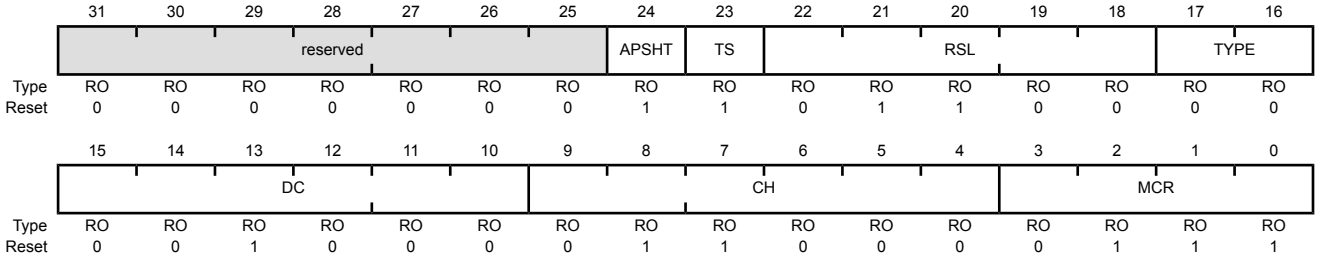

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 27:16 | COMP1 | RW | 0x000 | Compare 1<br>The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the high-band region.<br>Note that the value of COMP1 must be greater than or equal to the value of COMP0. |
| 15:12 | reserved | RO | 0x0 | 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. |
| 11:0 | COMP0 | RW | 0x000 | Compare 0<br>The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region. |

## Register 64: ADC Peripheral Properties (ADCPP), offset 0xFC0

The **ADCPP** register provides information regarding the properties of the ADC module.

ADC Peripheral Properties (ADCPP)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xFC0
Type RO, reset 0x01B0.2187




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:25 | reserved | RO | 0x00 | 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. |
| 24 | APSHT | RO | 0x1 | Application-Programmable Sample-and-Hold Time<br>This bit indicates the ADC has the capability of allowing the application to adjust the sample and hold window period. |
| 23 | TS | RO | 0x1 | Temperature Sensor<br>Value Description<br>0 The ADC module does not have a temperature sensor.<br>1 The ADC module has a temperature sensor.<br>This field provides the similar information as the legacy **DC1** register TEMPSNS bit. |
| 22:18 | RSL | RO | 0xC | Resolution<br>This field specifies the maximum number of binary bits used to represent the converted sample. The field is encoded as a binary value, in the range of 0 to 32 bits. |
| 17:16 | TYPE | RO | 0x0 | ADC Architecture<br>Value Description<br>0x0 SAR<br>0x1 - 0x3 Reserved |
| 15:10 | DC | RO | 0x8 | Digital Comparator Count<br>This field specifies the number of ADC digital comparators available to the converter. The field is encoded as a binary value, in the range of 0 to 63.<br>This field provides similar information to the legacy **DC9** register ADCnDCn bits. |

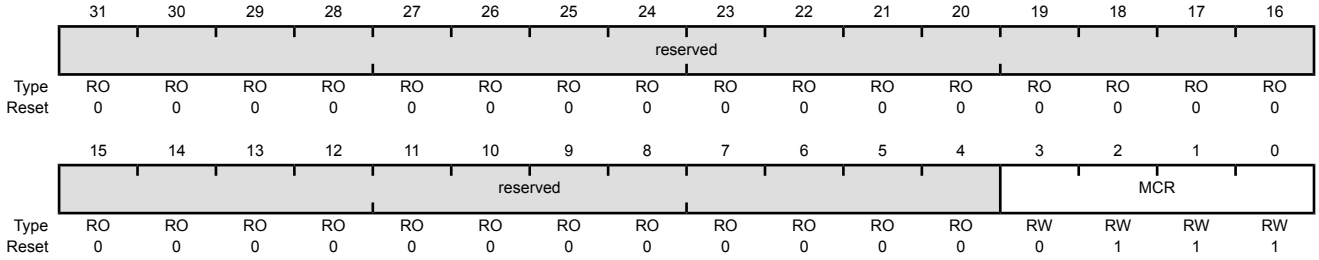
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9:4 | CH | RO | 0x18 | ADC Channel Count<br>This field specifies the number of ADC input channels available to the converter. This field is encoded as a binary value, in the range of 0 to 63.<br>This field provides similar information to the legacy **DC3** and **DC8** register ADCnAINn bits. |
| 3:0 | MCR | RO | 0x7 | Maximum Conversion Rate<br>This field specifies the maximum value that may be programmed into the **ADCPC** register's CR field.<br>Value Description<br>0x0-0x6 Reserved<br>0x7 Full conversion rate ($F_{CONV}$) as defined by $T_{ADC}$ and $N_{SH}$.<br>0x8 - 0xF Reserved |

## Register 65: ADC Peripheral Configuration (ADCPC), offset 0xFC4

The **ADCPC** register provides information regarding the configuration of the peripheral.

ADC Peripheral Configuration (ADCPC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xFC4
Type RW, reset 0x0000.0007




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.0000 | 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. |
| 3:0 | MCR | RW | 0x7 | Conversion Rate<br>This field specifies the relative sample rate of the ADC and is used in run, sleep, and deep-sleep modes. It allows the application to reduce the rate at which conversions are generated relative to the maximum conversion rate. |

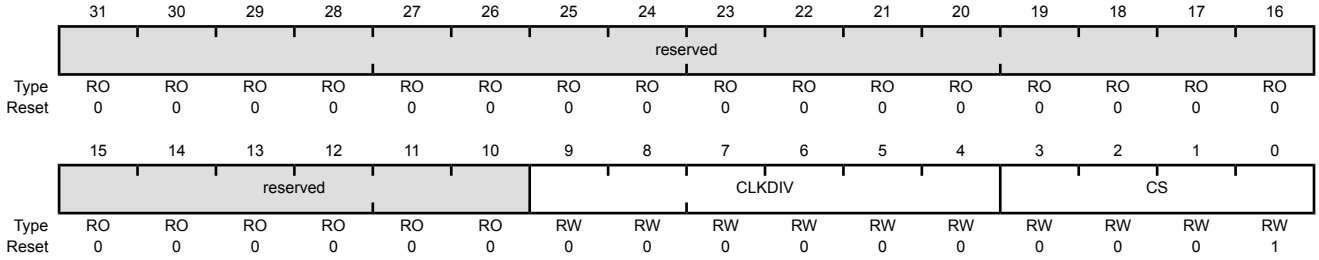
| Value | Description |
|---|---|
| 0x0 | Reserved |
| 0x1 | Eighth conversion rate. After a conversion completes, the logic pauses for 112 $T_{ADC}$ periods before starting the next conversion. |
| 0x2 | Reserved |
| 0x3 | Quarter conversion rate. After a conversion completes, the logic pauses for 48 $T_{ADC}$ periods before starting the next conversion. |
| 0x4 | Reserved |
| 0x5 | Half conversion rate. After a conversion completes, the logic pauses for 16 $T_{ADC}$ periods before starting the next conversion. |
| 0x6 | Reserved |
| 0x7 | Full conversion rate ($F_{CONV}$) as defined by $T_{ADC}$ and $N_{SH}$. |
| 0x8 - 0xF | Reserved |

## Register 66: ADC Clock Configuration (ADCCC), offset 0xFC8

The **ADCCC** register controls the clock source for the ADC module.

ADC Clock Configuration (ADCCC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xFC8
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9:4 | CLKDIV | RW | 0x0 | PLL VCO Clock Divisor<br>Value Description<br>0x0 /1<br>0x1 /2<br>0x2 /3<br>0xN /(N + 1) |
| 3:0 | CS | RW | 0x1 | ADC Clock Source<br>Value Description<br>0x0 PLL VCO divided by CLKDIV.<br>0x1 Alternate clock source as defined by **ALTCLKCFG** register in System Control Module.<br>0x2 MOSC<br>0x2 - 0xF Reserved |

# 19 Universal Asynchronous Receivers/Transmitters (UARTs)

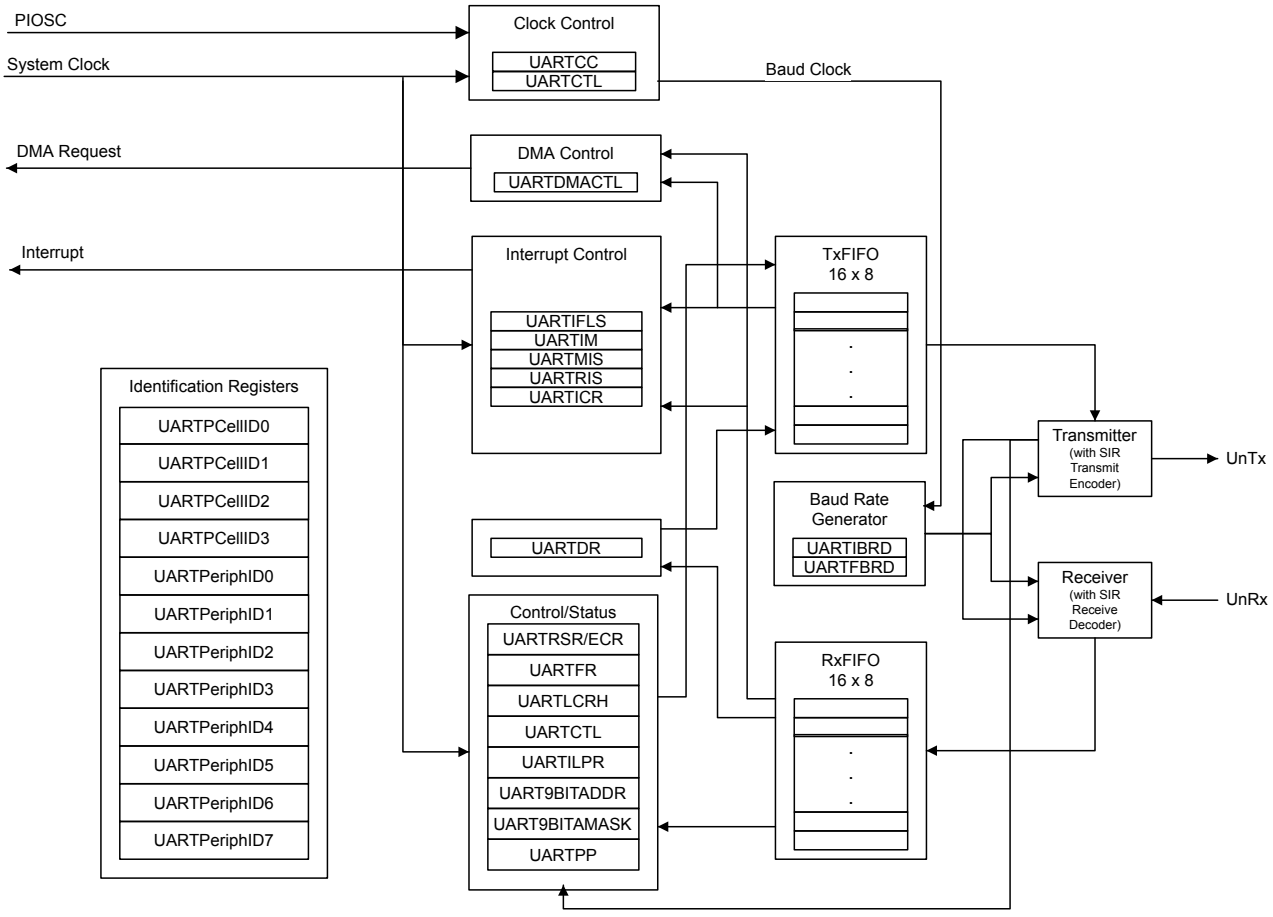
The TM4C129XKCZAD controller includes eight Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- Programmable baud-rate generator allowing speeds up to 7.5 Mbps for regular speed (divide by 16) and 15 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing 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/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23 µs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Modem functionality available on the following UARTs:
  - UART0 (modem flow control and modem status)
  - UART1 (modem flow control and modem status)
  - UART2 (modem flow control)
  - UART3 (modem flow control)
  - UART4 (modem flow control)
- EIA-485 9-bit support

- 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 asserted when there is space in the FIFO; burst request asserted at programmed FIFO level
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate baud clock

## 19.1 Block Diagram

**Figure 19-1. UART Module Block Diagram**




## 19.2 Signal Description

The following table lists the external signals of the UART module and describes the function of each. The UART signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin

placements for these UART signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the UART function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the UART signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 772.

**Table 19-1. UART Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| U0CTS | C6<br>A7<br>K17<br>R2<br>M18 | PB4 (1)<br>PE6 (1)<br>PG4 (1)<br>PH1 (1)<br>PM4 (1) | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| U0DCD | R1<br>G15<br>C12 | PH2 (1)<br>PM5 (1)<br>PP3 (2) | I | TTL | UART module 0 Data Carrier Detect modem status input signal. |
| U0DSR | T1<br>N19<br>D8 | PH3 (1)<br>PM6 (1)<br>PP4 (2) | I | TTL | UART module 0 Data Set Ready modem output control line. |
| U0DTR | R3<br>B13 | PH4 (1)<br>PP2 (1) | O | TTL | UART module 0 Data Terminal Ready modem status input signal. |
| U0RI | T2<br>W16<br>N18 | PH5 (1)<br>PK7 (1)<br>PM7 (1) | I | TTL | UART module 0 Ring Indicator modem status input signal. |
| U0RTS | B6<br>B7<br>K15<br>P4 | PB5 (1)<br>PE7 (1)<br>PG5 (1)<br>PH0 (1) | O | TTL | UART module 0 Request to Send modem flow control output signal. |
| U0Rx | V3 | PA0 (1) | I | TTL | UART module 0 receive. |
| U0Tx | W3 | PA1 (1) | O | TTL | UART module 0 transmit. |
| U1CTS | B11<br>C12 | PN1 (1)<br>PP3 (1) | I | TTL | UART module 1 Clear To Send modem flow control input signal. |
| U1DCD | G1<br>A11<br>B8 | PE2 (1)<br>PN2 (1)<br>PP6 (1) | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| U1DSR | H2<br>B10<br>B14 | PE1 (1)<br>PN3 (1)<br>PS2 (1) | I | TTL | UART module 1 Data Set Ready modem output control line. |
| U1DTR | G2<br>A10<br>U15 | PE3 (1)<br>PN4 (1)<br>PQ6 (1) | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| U1RI | A5<br>B9<br>M3 | PE4 (1)<br>PN5 (1)<br>PQ7 (1) | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| U1RTS | H3<br>C10<br>U12 | PE0 (1)<br>PN0 (1)<br>PN7 (1) | O | TTL | UART module 1 Request to Send modem flow control output line. |
| U1Rx | A16<br>A13<br>P2 | PB0 (1)<br>PQ4 (1)<br>PR5 (1) | I | TTL | UART module 1 receive. |
| U1Tx | B16<br>W12<br>W9 | PB1 (1)<br>PQ5 (1)<br>PR6 (1) | O | TTL | UART module 1 transmit. |

**Table 19-1. UART Signals (212BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| U2CTS | B2<br>F16<br>B10 | PD7 (1)<br>PJ3 (1)<br>PN3 (2) | I | TTL | UART module 2 Clear To Send modem flow control input signal. |
| U2RTS | B3<br>H17<br>A11 | PD6 (1)<br>PJ2 (1)<br>PN2 (2) | O | TTL | UART module 2 Request to Send modem flow control output line. |
| U2Rx | V5<br>A4 | PA6 (1)<br>PD4 (1) | I | TTL | UART module 2 receive. |
| U2Tx | R7<br>B4 | PA7 (1)<br>PD5 (1) | O | TTL | UART module 2 transmit. |
| U3CTS | E17<br>B9<br>B12 | PJ5 (1)<br>PN5 (2)<br>PP5 (1) | I | TTL | UART module 3 Clear To Send modem flow control input signal. |
| U3RTS | F18<br>A10<br>D8 | PJ4 (1)<br>PN4 (2)<br>PP4 (1) | O | TTL | UART module 3 Request to Send modem flow control output line. |
| U3Rx | V4<br>C8 | PA4 (1)<br>PJ0 (1) | I | TTL | UART module 3 receive. |
| U3Tx | W4<br>E7 | PA5 (1)<br>PJ1 (1) | O | TTL | UART module 3 transmit. |
| U4CTS | K5<br>K2<br>U12 | PJ7 (1)<br>PK3 (1)<br>PN7 (2) | I | TTL | UART module 4 Clear To Send modem flow control input signal. |
| U4RTS | N1<br>K1<br>T12 | PJ6 (1)<br>PK2 (1)<br>PN6 (2) | O | TTL | UART module 4 Request to Send modem flow control output line. |
| U4Rx | T6<br>J1<br>N4 | PA2 (1)<br>PK0 (1)<br>PR1 (1) | I | TTL | UART module 4 receive. |
| U4Tx | U5<br>J2<br>N5 | PA3 (1)<br>PK1 (1)<br>PR0 (1) | O | TTL | UART module 4 transmit. |
| U5Rx | L2<br>U2 | PC6 (1)<br>PH6 (1) | I | TTL | UART module 5 receive. |
| U5Tx | K3<br>V2 | PC7 (1)<br>PH7 (1) | O | TTL | UART module 5 transmit. |
| U6Rx | D6 | PP0 (1) | I | TTL | UART module 6 receive. |
| U6Tx | D7 | PP1 (1) | O | TTL | UART module 6 transmit. |
| U7Rx | M2<br>U2 | PC4 (1)<br>PH6 (2) | I | TTL | UART module 7 receive. |
| U7Tx | M1<br>V2 | PC5 (1)<br>PH7 (2) | O | TTL | UART module 7 transmit. |

## 19.3 Functional Description

Each TM4C129XKCZAD UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and/or receive via the TXE and RXE bits of the **UART Control (UARTCTL)** register (see page 1340). Transmit and receive are both enabled out of reset. Before any

control registers are programmed, the UART must be disabled by clearing the UARTEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

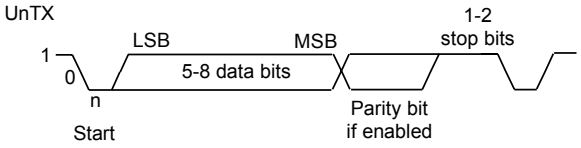
The UART module also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

### 19.3.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit 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. See Figure 19-2 on page 1316 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

**Figure 19-2. UART Character Frame**




### 19.3.2 Baud-Rate Generation

The baud-rate divisor 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 baud-rate divisor allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 1336) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 1337). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.)

```
BRD = BRDI + BRDF = UARTSysClk / (ClkDiv * Baud Rate)
```

where UARTSysClk is the system clock connected to the UART, and ClkDiv is either 16 (if HSE in **UARTCTL** is clear) or 8 (if HSE is set). By default, this will be the main system clock described in “Clock Control” on page 241. Alternatively, the UART may be clocked from the internal precision oscillator (PIOSC), independent of the system clock selection. This will allow the UART clock to be programmed independently of the system clock PLL settings. See the **UARTCC** register for more details.

The 6-bit fractional number (that is to be loaded into the DIVFRAC bit field in the **UARTFBRD** register) 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:

```
UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)
```

The UART generates an internal baud-rate reference clock at 8x or 16x the baud-rate (referred to as Baud8 and Baud16, depending on the setting of the HSE bit (bit 5) in **UARTCTL**). This reference clock is divided by 8 or 16 to generate the transmit clock, and is used for error detection during

receive operations. Note that the state of the HSE bit has no effect on clock generation in ISO 7816 smart card mode (when the SMART bit in the **UARTCTL** register is set).

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 1338), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write
- **UARTFBRD** write and **UARTLCRH** write

### 19.3.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the **UART Flag (UARTFR)** register (see page 1332) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The BUSY bit is negated only when the transmit FIFO is empty, and the last character has been 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 UnRx signal is continuously 1), and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of Baud16 or fourth cycle of Baud8 depending on the setting of the HSE bit (bit 5) in **UARTCTL** (described in “Transmit/Receive Logic” on page 1316).

The start bit is valid and recognized if the UnRx signal is still low on the eighth cycle of Baud16 (HSE clear) or the fourth cycle of Baud 8 (HSE set), otherwise it is ignored. After a valid start bit is detected, successive data bits are sampled on every 16th cycle of Baud16 or 8th cycle of Baud8 (that is, one bit period later) according to the programmed length of the data characters and value of the HSE bit in **UARTCTL**. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if the UnRx signal is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

### 19.3.4 Serial IR (SIR)

The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream and a half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output and decoded input to the UART. When enabled, the SIR block uses the UnTx and UnRx pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before

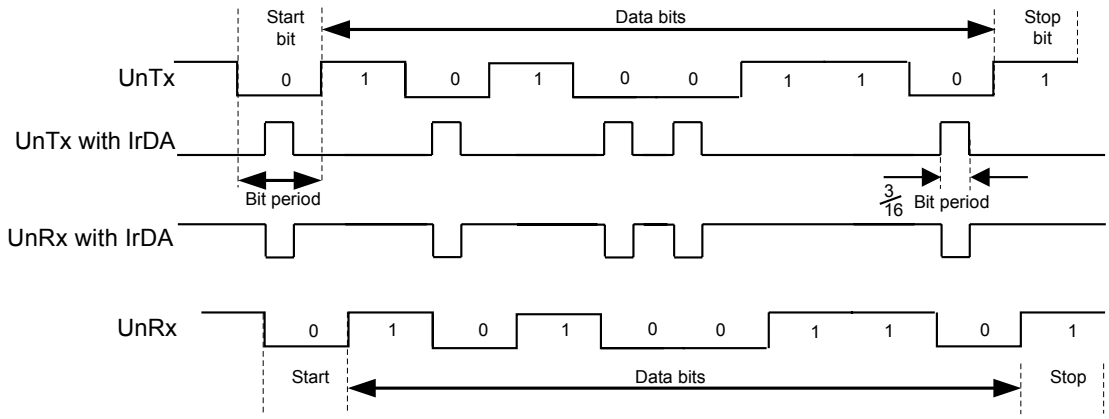
data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW and driving the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 µs, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCTL** register (see page 1340).

Whether the device is in normal or low-power IrDA mode, a start bit is deemed valid if the decoder is still Low, one period of IrLPBaud16 after the Low was first detected. This enables a normal-mode UART to receive data from a low-power mode UART that can transmit pulses as small as 1.41 µs. Thus, for both low-power and normal mode operation, the ILPDVSR field in the **UARTILPR** register must be programmed such that 1.42 MHz < $F_{IrLPBaud16}$ < 2.12 MHz, resulting in a low-power pulse duration of 1.41–2.11 µs (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 µs are accepted as valid pulses.

Figure 19-3 on page 1318 shows the UART transmit and receive signals, with and without IrDA modulation.

**Figure 19-3. IrDA Data Modulation**




In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10-ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency or receiver setup time.

### 19.3.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (SMART) of the **UARTCTL** register is set, the UnTx signal is used as a bit clock, and the UnRx signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design. The maximum clock rate in this mode is system clock / 16.

When using ISO 7816 mode, the **UARTLCRH** register must be set to transmit 8-bit words (WLEN bits 6:5 configured to 0x3) with EVEN parity (PEN set and EPS set). In this mode, the UART automatically uses 2 stop bits, and the STP2 bit of the **UARTLCRH** register is ignored.

If a parity error is detected during transmission, UnRx is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

### 19.3.6 Modem Handshake Support

This section describes how to configure and use the modem flow control and status signals for a UART when connected as a DTE (data terminal equipment) or as a DCE (data communications equipment). In general, a modem is a DCE and a computing device that connects to a modem is the DTE.

#### 19.3.6.1 Signaling

The status signals provided by a UART differ based on whether the UART is used as a DTE or DCE. When used as a DTE, the modem flow control and status signals are defined as:

- $\overline{\text{UnCTS}}$ is Clear To Send
- $\overline{\text{UnDSR}}$ is Data Set Ready
- $\overline{\text{UnDCD}}$ is Data Carrier Detect
- $\overline{\text{UnRI}}$ is Ring Indicator
- $\overline{\text{UnRTS}}$ is Request To Send
- $\overline{\text{UnDTR}}$ is Data Terminal Ready

When used as a DCE, the modem flow control and status signals are defined as:

- $\overline{\text{UnCTS}}$ is Request To Send
- $\overline{\text{UnDSR}}$ is Data Terminal Ready
- $\overline{\text{UnRTS}}$ is Clear To Send
- $\overline{\text{UnDTR}}$ is Data Set Ready

Note that the support for DCE functions Data Carrier Detect and Ring Indicator are not provided. If these signals are required, their function can be emulated by using a general-purpose I/O signal and providing software support.

### 19.3.6.2 Flow Control

Flow control can be accomplished by either hardware or software. The following sections describe the different methods.

#### *Hardware Flow Control (RTS/CTS)*

Hardware flow control between two devices is accomplished by connecting the $\overline{\text{UnRTS}}$ output to the Clear-To-Send input on the receiving device, and connecting the Request-To-Send output on the receiving device to the $\overline{\text{UnCTS}}$ input.

The $\overline{\text{UnCTS}}$ input controls the transmitter. The transmitter may only transmit data when the $\overline{\text{UnCTS}}$ input is asserted. The $\overline{\text{UnRTS}}$ output signal indicates the state of the receive FIFO. $\overline{\text{UnCTS}}$ remains asserted until the preprogrammed watermark level is reached, indicating that the Receive FIFO has no space to store additional characters.

The **UARTCTL** register bits 15 (CTSEN) and 14 (RTSEN) specify the flow control mode as shown in Table 19-2 on page 1320.

**Table 19-2. 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 |

Note that when RTSEN is 1, software cannot modify the $\overline{\text{UnRTS}}$ output value through the **UARTCTL** register Request to Send (RTS) bit, and the status of the RTS bit should be ignored.

#### *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 may be generated for the $\overline{\text{UnDSR}}$, $\overline{\text{UnDCD}}$, $\overline{\text{UnCTS}}$, and $\overline{\text{UnRI}}$ signals using bits 3:0 of the **UARTIM** register, respectively. The raw and masked interrupt status may be checked using the **UARTRIS** and **UARTMIS** register. These interrupts may be cleared using the **UARTICR** register.

## 19.3.7 9-Bit UART Mode

The UART provides a 9-bit mode that is enabled with the 9BITEN bit in the **UART9BITADDR** register. This feature is useful in a multi-drop configuration of the UART where a single master connected to multiple slaves can communicate with a particular slave through its address or set of addresses along with a qualifier for an address byte. All the slaves check for the address qualifier in the place of the parity bit and, if set, then compare the byte received with the preprogrammed address. If the address matches, then it receives or sends further data. If the address does not match, it drops the address byte and any subsequent data bytes. If the UART is in 9-bit mode, then the receiver operates with no parity mode. The address can be predefined to match with the received byte and it can be configured with the **UART9BITADDR** register. The matching can be extended to a set of addresses using the address mask in the **UART9BITAMASK** register. By default, the **UART9BITAMASK** is 0xFF, meaning that only the specified address is matched.

When not finding a match, the rest of the data bytes with the 9th bit cleared are dropped. If a match is found, then an interrupt is generated to the NVIC for further action. The subsequent data bytes with the cleared 9th bit are stored in the FIFO. Software can mask this interrupt in case µDMA and/or FIFO operations are enabled for this instance and processor intervention is not required. All the

send transactions with 9-bit mode are data bytes and the 9th bit is cleared. Software can override the 9th bit to be set (to indicate address) by overriding the parity settings to sticky parity with odd parity enabled for a particular byte. To match the transmission time with correct parity settings, the address byte can be transmitted as a single then a burst transfer. The Transmit FIFO does not hold the address/data bit, hence software should take care of enabling the address bit appropriately.

### 19.3.8 FIFO Operation

The UART has two 16x8 FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 1327). Read operations of the **UARTDR** register return a 12-bit value consisting of 8 data bits and 4 error flags while write operations place 8-bit data in the transmit FIFO.

Out of reset, both FIFOs are disabled and act as 1-byte-deep holding registers. The FIFOs are enabled by setting the FEN bit in **UARTLCRH** (page 1338).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 1332) and the **UART Receive Status (UARTRSR)** register. Hardware monitors empty, full and overrun conditions. The **UARTFR** register contains empty and full flags (TXFE, TXFF, RXFE, and RXFF bits), and the **UARTRSR** register shows overrun status via 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 is controlled via the **UART Interrupt FIFO Level Select (UARTIFLS)** register (see page 1344). 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 4 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the ½ mark.

### 19.3.9 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error
- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the TXIFLSEL bit in the **UARTIFLS** register is met, or if the EOT bit in **UARTCTL** is set, when the last bit of all transmitted data leaves the serializer)
- Receive (when condition defined in the RXIFLSEL bit in the **UARTIFLS** register is met)

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 by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 1354).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 1346) by setting the corresponding IM bits. If interrupts are not used, the raw interrupt status is visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 1350).

**Note:** For receive timeout, the RTIM bit in the **UARTIM** register must be set to see the RTMIS and RTRIS status in the **UARTMIS** and **UARTRIS** registers.

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by writing a 1 to the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 1358).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period when the HSE bit is clear or over a 64-bit period when the HSE bit is set. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

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, the RXRIS bit is set. 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 by writing a 1 to the RXIC bit.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the RXRIS bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt by writing a 1 to the RXIC bit.

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO progresses through the programmed trigger level, the TXRIS bit is set. The transmit interrupt is based on a transition through level, therefore the FIFO must be written past the programmed trigger level otherwise no further transmit interrupts will be generated. 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 by writing a 1 to the TXIC bit.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXRIS bit is set. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt by writing a 1 to the TXIC bit.

### 19.3.10 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the LBE bit in the **UARTCTL** register (see page 1340). In loopback mode, data transmitted on the UnTx output is received on the UnRx input. Note that the LBE bit should be set before the UART is enabled.

### 19.3.11 DMA Operation

The UART provides an interface to the µDMA controller with separate channels for transmit and receive. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART asserts a DMA 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 receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level configured in the **UARTIFLS** register. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the µDMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, set the RXDMAE bit of the **DMA Control (UARTDMACTL)** register. To enable DMA operation for the transmit channel, set the TXDMAE bit of the **UARTDMACTL** register. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the DMAERR bit of the **UARTDMACR** register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

When the µDMA is finished transferring data to the TX FIFO or from the RX FIFO, a dma_done signal is sent to the UART to indicate completion. The dma_done status is indicated through the DMATXRIS and DMARXIS bits of the **UARTRIS** register. An interrupt can be generated from these status bits by setting the DMATXIM and/or DMARXIM bits in the **UARTIM** register.

**Note:** The DMATXRIS bit can be used to indicate the µDMA's completion of data transfer to the TX FIFO. To indicate transfer completion from the UART's serializer, the end-of-transmission bit (EOT bit) should be enabled in the **UARTCTL** register. An interrupt can be generated on an end-of-transmission completion by setting the EOTIM bit of the **UARTIM** register.

See “Micro Direct Memory Access (µDMA)” on page 708 for more details about programming the µDMA controller.

## 19.4 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

1. Enable the UART module using the **RCGCUART** register (see page 405).
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register (see page 399). To find out which GPIO port to enable, refer to Table 31-5 on page 2081.
3. Set the GPIO AFSEL bits for the appropriate pins (see page 803). To determine which GPIOs to configure, see Table 31-4 on page 2064.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected (see page 805 and page 813).
5. Configure the PMCn fields in the **GPIOPCTL** register to assign the UART signals to the appropriate pins (see page 820 and Table 31-5 on page 2081).

To use the UART, the peripheral clock must be enabled by setting the appropriate bit in the **RCGCUART** register (page 405). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGCGPIO** register (page 399) in the System Control module. To find out which GPIO port to enable, refer to Table 31-5 on page 2081.

This section discusses the steps that are required to use a UART module. For this example, the UART clock is assumed to be 20 MHz, and the desired UART configuration is:

- 115200 baud rate
- 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 baud-rate divisor (BRD), because the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in "Baud-Rate Generation" on page 1316, the BRD can be calculated:

```
BRD = 20,000,000 / (16 * 115,200) = 10.8507
```

which means that the DIVINT field of the **UARTIBRD** register (see page 1336) should be set to 10 decimal or 0xA. The value to be loaded into the **UARTFBRD** register (see page 1337) is calculated by the equation:

```
UARTFBRD[DIVFRAC] = integer(0.8507 * 64 + 0.5) = 54
```

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the UARTEN bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.
4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of 0x0000.0060).
5. Configure the UART clock source by writing to the **UARTCC** register.
6. Optionally, configure the µDMA channel (see "Micro Direct Memory Access (µDMA)" on page 708) and enable the DMA option(s) in the **UARTDMACTL** register.
7. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

## 19.5 Register Map

Table 19-3 on page 1325 lists the UART registers. The offset listed is a hexadecimal increment to the register's address, relative to that UART's base address:

- UART0: 0x4000.C000
- UART1: 0x4000.D000
- UART2: 0x4000.E000
- UART3: 0x4000.F000
- UART4: 0x4001.0000
- UART5: 0x4001.1000
- UART6: 0x4001.2000
- UART7: 0x4001.3000

The UART module clock must be enabled before the registers can be programmed (see page 405). There must be a delay of 3 system clocks after the UART module clock is enabled before any UART module registers are accessed.

The UART must be disabled (see the UARTEN bit in the **UARTCTL** register on page 1340) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

**Note:** Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)

- UART1 (modem flow control and modem status)
- UART2 (modem flow control)
- UART3 (modem flow control)
- UART4 (modem flow control)

**Table 19-3. UART Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | UARTDR | RW | 0x0000.0000 | UART Data | 1327 |
| 0x004 | UARTRSR/UARTECR | RW | 0x0000.0000 | UART Receive Status/Error Clear | 1329 |
| 0x018 | UARTFR | RO | 0x0000.0090 | UART Flag | 1332 |
| 0x020 | UARTILPR | RW | 0x0000.0000 | UART IrDA Low-Power Register | 1335 |
| 0x024 | UARTIBRD | RW | 0x0000.0000 | UART Integer Baud-Rate Divisor | 1336 |
| 0x028 | UARTFBRD | RW | 0x0000.0000 | UART Fractional Baud-Rate Divisor | 1337 |
| 0x02C | UARTLCRH | RW | 0x0000.0000 | UART Line Control | 1338 |
| 0x030 | UARTCTL | RW | 0x0000.0300 | UART Control | 1340 |
| 0x034 | UARTIFLS | RW | 0x0000.0012 | UART Interrupt FIFO Level Select | 1344 |
| 0x038 | UARTIM | RW | 0x0000.0000 | UART Interrupt Mask | 1346 |
| 0x03C | UARTRIS | RO | 0x0000.0000 | UART Raw Interrupt Status | 1350 |
| 0x040 | UARTMIS | RO | 0x0000.0000 | UART Masked Interrupt Status | 1354 |
| 0x044 | UARTICR | W1C | 0x0000.0000 | UART Interrupt Clear | 1358 |
| 0x048 | UARTDMACTL | RW | 0x0000.0000 | UART DMA Control | 1360 |
| 0x0A4 | UART9BITADDR | RW | 0x0000.0000 | UART 9-Bit Self Address | 1361 |
| 0x0A8 | UART9BITAMASK | RW | 0x0000.00FF | UART 9-Bit Self Address Mask | 1362 |
| 0xFC0 | UARTPP | RO | 0x0000.000F | UART Peripheral Properties | 1363 |
| 0xFC8 | UARTCC | RW | 0x0000.0000 | UART Clock Configuration | 1365 |
| 0xFD0 | UARTPeriphID4 | RO | 0x0000.0060 | UART Peripheral Identification 4 | 1366 |
| 0xFD4 | UARTPeriphID5 | RO | 0x0000.0000 | UART Peripheral Identification 5 | 1367 |
| 0xFD8 | UARTPeriphID6 | RO | 0x0000.0000 | UART Peripheral Identification 6 | 1368 |
| 0xFDC | UARTPeriphID7 | RO | 0x0000.0000 | UART Peripheral Identification 7 | 1369 |
| 0xFE0 | UARTPeriphID0 | RO | 0x0000.0011 | UART Peripheral Identification 0 | 1370 |
| 0xFE4 | UARTPeriphID1 | RO | 0x0000.0000 | UART Peripheral Identification 1 | 1371 |
| 0xFE8 | UARTPeriphID2 | RO | 0x0000.0018 | UART Peripheral Identification 2 | 1372 |
| 0xFEC | UARTPeriphID3 | RO | 0x0000.0001 | UART Peripheral Identification 3 | 1373 |

**Table 19-3. UART Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xFF0 | UARTPCellID0 | RO | 0x0000.000D | UART PrimeCell Identification 0 | 1374 |
| 0xFF4 | UARTPCellID1 | RO | 0x0000.00F0 | UART PrimeCell Identification 1 | 1375 |
| 0xFF8 | UARTPCellID2 | RO | 0x0000.0005 | UART PrimeCell Identification 2 | 1376 |
| 0xFFC | UARTPCellID3 | RO | 0x0000.00B1 | UART PrimeCell Identification 3 | 1377 |

## 19.6 Register Descriptions

The remainder of this section lists and describes the UART registers, in numerical order by address offset.

## Register 1: UART Data (UARTDR), offset 0x000

**Important:** This register is read-sensitive. See the register description for details.

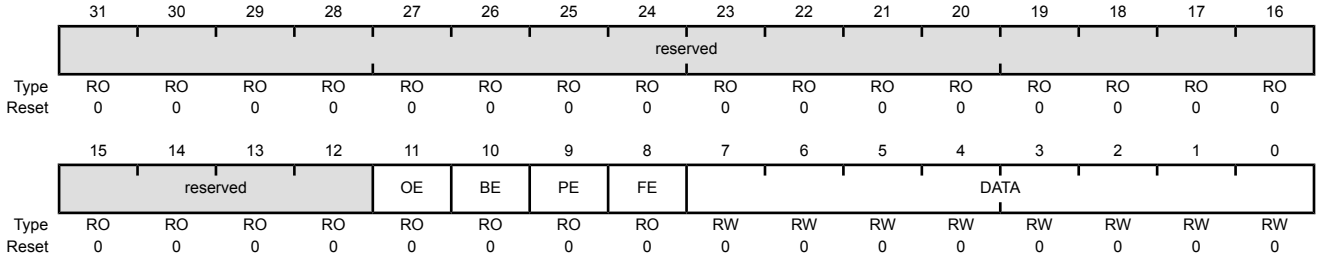
This register is the data register (the interface to the FIFOs).

For transmitted data, if the FIFO is enabled, data written to this location is pushed onto the transmit FIFO. If the FIFO is disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If the FIFO is disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |
| 11 | OE | RO | 0 | UART Overrun Error<br>Value Description<br>0 No data has been lost due to a FIFO overrun.<br>1 New data was received when the FIFO was full, resulting in data loss. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | BE | RO | 0 | UART Break Error<br><br>Value Description<br>0 No break condition has occurred<br>1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).<br><br>In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state), and the next valid start bit is received. |
| 9 | PE | RO | 0 | UART Parity Error<br><br>Value Description<br>0 No parity error has occurred<br>1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register.<br><br>In FIFO mode, this error is associated with the character at the top of the FIFO. |
| 8 | FE | RO | 0 | UART Framing Error<br><br>Value Description<br>0 No framing error has occurred<br>1 The received character does not have a valid stop bit (a valid stop bit is 1). |
| 7:0 | DATA | RW | 0x00 | Data Transmitted or Received<br>Data that is to be transmitted via the UART is written to this field.<br>When read, this field contains the data that was received by the UART. |

## Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

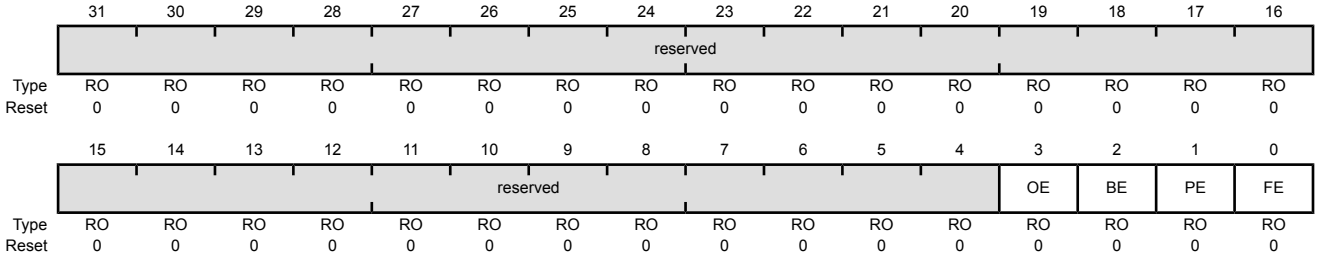
The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared on reset.

### Read-Only Status Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x004
Type RO, reset 0x0000.0000



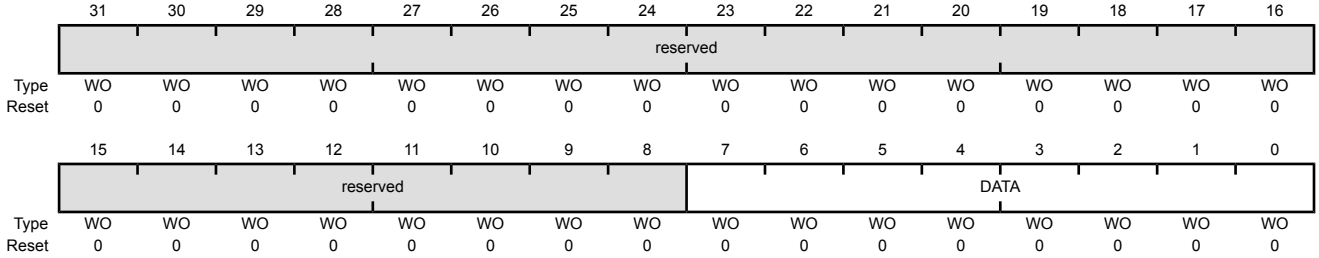

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | OE | RO | 0 | UART Overrun Error<br><br>Value Description<br>0 No data has been lost due to a FIFO overrun.<br>1 New data was received when the FIFO was full, resulting in data loss.<br><br>This bit is cleared by a write to **UARTECR**.<br><br>The FIFO contents remain valid because no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must read the data in order to empty the FIFO. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | BE | RO | 0 | UART Break Error<br><br>Value Description<br>0 No break condition has occurred<br>1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).<br><br>This bit is cleared to 0 by a write to **UARTECR**.<br><br>In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received. |
| 1 | PE | RO | 0 | UART Parity Error<br><br>Value Description<br>0 No parity error has occurred<br>1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register.<br><br>This bit is cleared to 0 by a write to **UARTECR**. |
| 0 | FE | RO | 0 | UART Framing Error<br><br>Value Description<br>0 No framing error has occurred<br>1 The received character does not have a valid stop bit (a valid stop bit is 1).<br><br>This bit is cleared to 0 by a write to **UARTECR**.<br><br>In FIFO mode, this error is associated with the character at the top of the FIFO. |

## Write-Only Error Clear Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x004
Type WO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | WO | 0x0000.00 | 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. |
| 7:0 | DATA | WO | 0x00 | Error Clear<br>A write to this register of any data clears the framing, parity, break, and overrun flags. |

## Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the TXFF, RXFF, and BUSY bits are 0, and TXFE and RXFE bits are 1.

**Note:** Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)
- UART1 (modem flow control and modem status)
- UART2 (modem flow control)
- UART3 (modem flow control)
- UART4 (modem flow control)

UART Flag (UARTFR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x018
Type RO, reset 0x0000.0090

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | RI | TXFE | RXFF | TXFF | RXFE | BUSY | DCD | DSR | CTS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | 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. |
| 8 | RI | RO | 0 | Ring Indicator<br>Value Description<br>0 The UnRI signal is not asserted.<br>1 The UnRI signal is asserted. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | TXFE | RO | 1 | UART Transmit FIFO Empty<br>The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register.<br><br>Value Description<br>0 The transmitter has data to transmit.<br>1 If the FIFO is disabled (FEN is 0), the transmit holding register is empty.<br>If the FIFO is enabled (FEN is 1), the transmit FIFO is empty. |
| 6 | RXFF | RO | 0 | UART Receive FIFO Full<br>The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register.<br><br>Value Description<br>0 The receiver can receive data.<br>1 If the FIFO is disabled (FEN is 0), the receive holding register is full.<br>If the FIFO is enabled (FEN is 1), the receive FIFO is full. |
| 5 | TXFF | RO | 0 | UART Transmit FIFO Full<br>The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register.<br><br>Value Description<br>0 The transmitter is not full.<br>1 If the FIFO is disabled (FEN is 0), the transmit holding register is full.<br>If the FIFO is enabled (FEN is 1), the transmit FIFO is full. |
| 4 | RXFE | RO | 1 | UART Receive FIFO Empty<br>The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register.<br><br>Value Description<br>0 The receiver is not empty.<br>1 If the FIFO is disabled (FEN is 0), the receive holding register is empty.<br>If the FIFO is enabled (FEN is 1), the receive FIFO is empty. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | BUSY | RO | 0 | UART Busy<br><br>Value Description<br>0 The UART is not busy.<br>1 The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register.<br><br>This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled). |
| 2 | DCD | RO | 0 | Data Carrier Detect<br><br>Value Description<br>0 The UnDCD signal is not asserted.<br>1 The UnDCD signal is asserted. |
| 1 | DSR | RO | 0 | Data Set Ready<br><br>Value Description<br>0 The UnDSR signal is not asserted.<br>1 The UnDSR signal is asserted. |
| 0 | CTS | RO | 0 | Clear To Send<br><br>Value Description<br>0 The U1CTS signal is not asserted.<br>1 The UnCTS signal is asserted. |

## Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register stores the 8-bit low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared when reset.

The internal IrLPBaud16 clock is generated by dividing down SysClk according to the low-power divisor value written to **UARTILPR**. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the IrLPBaud16 clock. The low-power divisor value is calculated as follows:

ILPDVSR = SysClk / $F_{IrLPBaud16}$

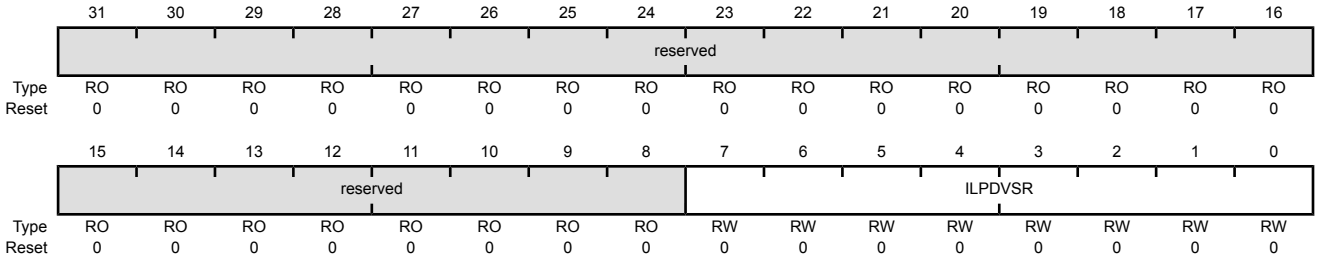
where $F_{IrLPBaud16}$ is nominally 1.8432 MHz.

Because the IrLPBaud16 clock is used to sample transmitted data irrespective of mode, the ILPDVSR field must be programmed in both low power and normal mode,such that 1.42 MHz < $F_{IrLPBaud16}$ < 2.12 MHz, resulting in a low-power pulse duration of 1.41–2.11 µs (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 µs are accepted as valid pulses.

**Note:** Zero is an illegal value. Programming a zero value results in no IrLPBaud16 pulses being generated.

UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x020
Type RW, reset 0x0000.0000



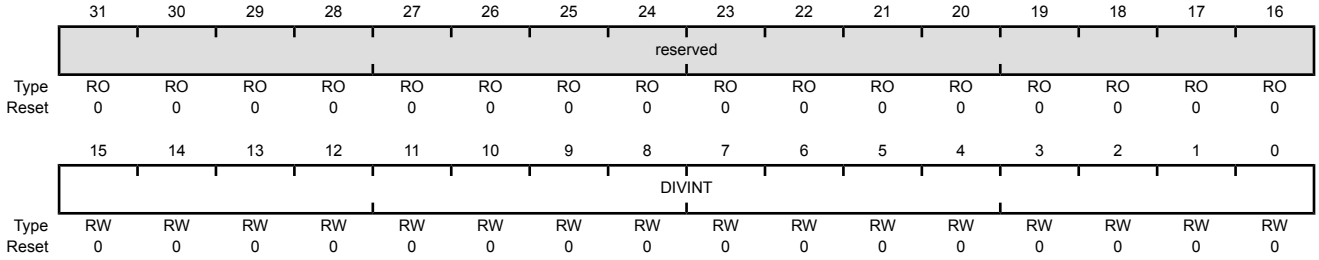

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | ILPDVSR | RW | 0x00 | IrDA Low-Power Divisor<br>This field contains the 8-bit low-power divisor value. |

## Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 1316 for configuration details.

UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x024
Type RW, reset 0x0000.0000



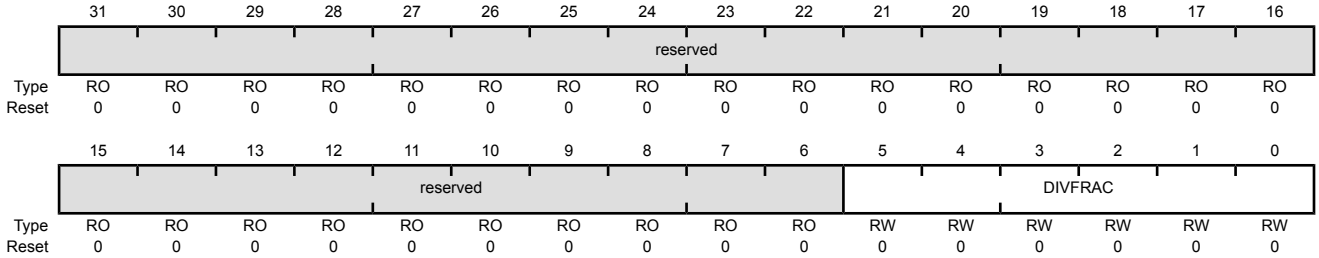

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | DIVINT | RW | 0x0000 | Integer Baud-Rate Divisor |

## Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 1316 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x028
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.000 | 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. |
| 5:0 | DIVFRAC | RW | 0x0 | Fractional Baud-Rate Divisor |

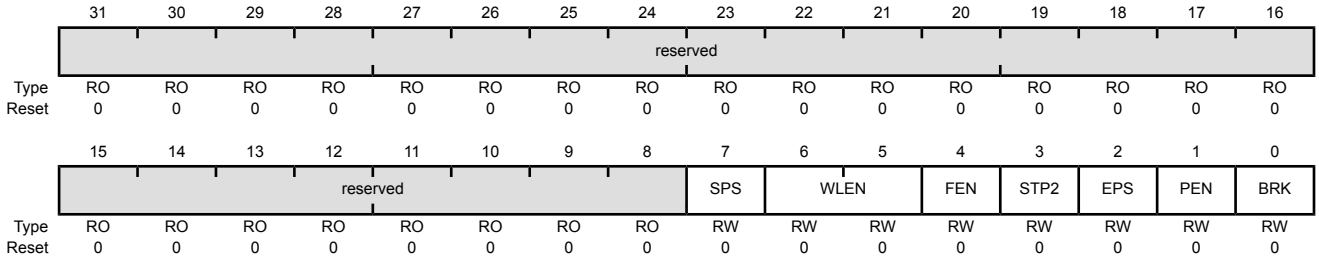
## Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x02C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | SPS | RW | 0 | UART Stick Parity Select<br>When bits 1, 2, and 7 of **UARTLCRH** are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1.<br>When this bit is cleared, stick parity is disabled. |
| 6:5 | WLEN | RW | 0x0 | UART Word Length<br>The bits indicate the number of data bits transmitted or received in a frame as follows:<br>Value Description<br>0x0 5 bits (default)<br>0x1 6 bits<br>0x2 7 bits<br>0x3 8 bits |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | FEN | RW | 0 | UART Enable FIFOs<br><br>Value Description<br>0 The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.<br>1 The transmit and receive FIFO buffers are enabled (FIFO mode). |
| 3 | STP2 | RW | 0 | UART Two Stop Bits Select<br><br>Value Description<br>0 One stop bit is transmitted at the end of a frame.<br>1 Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.<br>When in 7816 smartcard mode (the SMART bit is set in the **UARTCTL** register), the number of stop bits is forced to 2. |
| 2 | EPS | RW | 0 | UART Even Parity Select<br><br>Value Description<br>0 Odd parity is performed, which checks for an odd number of 1s.<br>1 Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits.<br><br>This bit has no effect when parity is disabled by the PEN bit. |
| 1 | PEN | RW | 0 | UART Parity Enable<br><br>Value Description<br>0 Parity is disabled and no parity bit is added to the data frame.<br>1 Parity checking and generation is enabled. |
| 0 | BRK | RW | 0 | UART Send Break<br><br>Value Description<br>0 Normal use.<br>1 A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods). |

## Register 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set.

To enable the UART module, the UARTEN bit must be set. If software requires a configuration change in the module, the UARTEN bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

**Note:** Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)
- UART1 (modem flow control and modem status)
- UART2 (modem flow control)
- UART3 (modem flow control)
- UART4 (modem flow control)

**Note:** The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by clearing bit 4 (FEN) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x030
Type RW, reset 0x0000.0300

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CTSEN | RTSEN | reserved | | RTS | DTR | RXE | TXE | LBE | reserved | HSE | EOT | SMART | SIRLP | SIREN | UARTEN |
| Type | RW | RW | RO | RO | RW | RW | RW | RW | RW | RO | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15 | CTSEN | RW | 0 | Enable Clear To Send<br>Value Description<br>0 CTS hardware flow control is disabled.<br>1 CTS hardware flow control is enabled. Data is only transmitted when the UnCTS signal is asserted. |
| 14 | RTSEN | RW | 0 | Enable Request to Send<br>Value Description<br>0 RTS hardware flow control is disabled.<br>1 RTS hardware flow control is enabled. Data is only requested (by asserting UnRTS) when the receive FIFO has available entries. |
| 13:12 | reserved | RO | 0 | 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. |
| 11 | RTS | RW | 0 | Request to Send<br>When RTSEN is clear, the status of this bit is reflected on the U1RTS signal. If RTSEN is set, this bit is ignored on a write and should be ignored on read. |
| 10 | DTR | RW | 0 | Data Terminal Ready<br>This bit sets the state of the UnDTR output. |
| 9 | RXE | RW | 1 | UART Receive Enable<br>Value Description<br>0 The receive section of the UART is disabled.<br>1 The receive section of the UART is enabled.<br>If the UART is disabled in the middle of a receive, it completes the current character before stopping.<br>**Note:** To enable reception, the UARTEN bit must also be set. |
| 8 | TXE | RW | 1 | UART Transmit Enable<br>Value Description<br>0 The transmit section of the UART is disabled.<br>1 The transmit section of the UART is enabled.<br>If the UART is disabled in the middle of a transmission, it completes the current character before stopping.<br>**Note:** To enable transmission, the UARTEN bit must also be set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | LBE | RW | 0 | UART Loop Back Enable<br><br>Value Description<br>0 Normal operation.<br>1 The UnTx path is fed through the UnRx path. |
| 6 | reserved | RO | 0 | 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 | HSE | RW | 0 | High-Speed Enable<br><br>Value Description<br>0 The UART is clocked using the system clock divided by 16.<br>1 The UART is clocked using the system clock divided by 8.<br><br>**Note:** System clock used is also dependent on the baud-rate divisor configuration (see page 1336) and page 1337).<br><br>The state of this bit has no effect on clock generation in ISO 7816 smart card mode (the SMART bit is set). |
| 4 | EOT | RW | 0 | End of Transmission<br>This bit determines the behavior of the TXRIS bit in the **UARTRIS** register.<br><br>Value Description<br>0 The TXRIS bit is set when the transmit FIFO condition specified in **UARTIFLS** is met.<br>1 The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer. |
| 3 | SMART | RW | 0 | ISO 7816 Smart Card Support<br><br>Value Description<br>0 Normal operation.<br>1 The UART operates in Smart Card mode.<br><br>The application must ensure that it sets 8-bit word length (WLEN set to 0x3) and even parity (PEN set to 1, EPS set to 1, SPS set to 0) in **UARTLCRH** when using ISO 7816 mode.<br>In this mode, the value of the STP2 bit in **UARTLCRH** is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | SIRLP | RW | 0 | UART SIR Low-Power Mode<br>This bit selects the IrDA encoding mode.<br><br>Value Description<br>0 Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.<br>1 The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the `IrLPBaud16` input signal, regardless of the selected bit rate.<br><br>Setting this bit uses less power, but might reduce transmission distances. See page 1335 for more information. |
| 1 | SIREN | RW | 0 | UART SIR Enable<br><br>Value Description<br>0 Normal operation.<br>1 The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol. |
| 0 | UARTEN | RW | 0 | UART Enable<br><br>Value Description<br>0 The UART is disabled.<br>1 The UART is enabled.<br><br>If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping. |

## Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

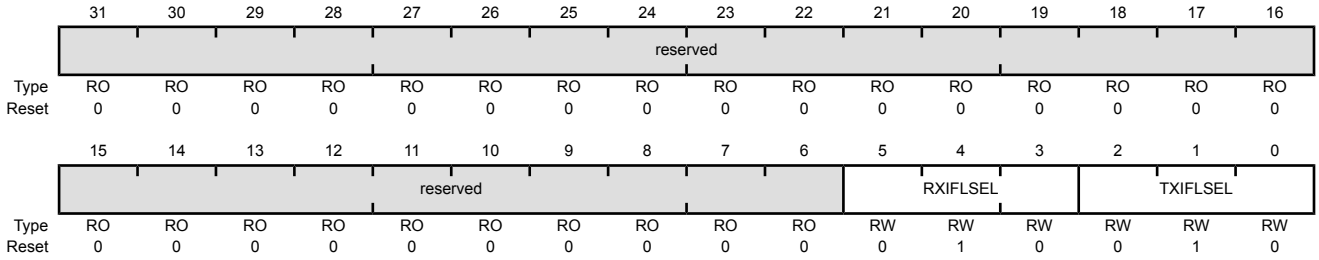
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x034
Type RW, reset 0x0000.0012




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.00 | 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. |
| 5:3 | RXIFLSEL | RW | 0x2 | UART Receive Interrupt FIFO Level Select<br>The trigger points for the receive interrupt are as follows:<br>Value Description<br>0x0 RX FIFO ≥ ⅛ full<br>0x1 RX FIFO ≥ ¼ full<br>0x2 RX FIFO ≥ ½ full (default)<br>0x3 RX FIFO ≥ ¾ full<br>0x4 RX FIFO ≥ ⅞ full<br>0x5-0x7 Reserved |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2:0 | TXIFLSEL | RW | 0x2 | UART Transmit Interrupt FIFO Level Select<br>The trigger points for the transmit interrupt are as follows: |

| Value | Description |
|---|---|
| 0x0 | TX FIFO ≤ ⅞ empty |
| 0x1 | TX FIFO ≤ ¾ empty |
| 0x2 | TX FIFO ≤ ½ empty (default) |
| 0x3 | TX FIFO ≤ ¼ empty |
| 0x4 | TX FIFO ≤ ⅛ empty |
| 0x5-0x7 | Reserved |

**Note:** If the EOT bit in **UARTCTL** is set (see page 1340), the transmit interrupt is generated once the FIFO is completely empty and all data including stop bits have left the transmit serializer. In this case, the setting of TXIFLSEL is ignored.

## Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

**Note:** Registers that contain bits for modem control or status only apply to the following UARTs:

- UART0 (modem flow control and modem status)
- UART1 (modem flow control and modem status)
- UART2 (modem flow control)
- UART3 (modem flow control)
- UART4 (modem flow control)

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x038
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | DMATXIM | DMARXIM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | 9BITIM | EOTIM | OEIM | BEIM | PEIM | FEIM | RTIM | TXIM | RXIM | DSRIM | DCDIM | CTSIM | RIIM |
| Type | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | DMATXIM | RW | 0 | Transmit DMA Interrupt Mask |

| Value | Description |
|---|---|
| 0 | The DMATXRIS interrupt is suppressed and not sent to the interrupt controller. |
| 1 | An interrupt is sent to the interrupt controller when the DMATXRIS bit in the **UARTRIS** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | DMARXIM | RW | 0 | Receive DMA Interrupt Mask<br>Value Description<br>0 The DMARXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the DMARXRIS bit in the **UARTRIS** register is set. |
| 15:13 | reserved | RO | 0 | 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. |
| 12 | 9BITIM | RW | 0 | 9-Bit Mode Interrupt Mask<br>Value Description<br>0 The 9BITRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the 9BITRIS bit in the **UARTRIS** register is set. |
| 11 | EOTIM | RW | 0 | End of Transmission Interrupt Mask<br>Value Description<br>0 The EOTRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the EOTRIS bit in the **UARTRIS** register is set. |
| 10 | OEIM | RW | 0 | UART Overrun Error Interrupt Mask<br>Value Description<br>0 The OERIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the OERIS bit in the **UARTRIS** register is set. |
| 9 | BEIM | RW | 0 | UART Break Error Interrupt Mask<br>Value Description<br>0 The BERIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the BERIS bit in the **UARTRIS** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | PEIM | RW | 0 | UART Parity Error Interrupt Mask<br>Value Description<br>0 The PERIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PERIS bit in the **UARTRIS** register is set. |
| 7 | FEIM | RW | 0 | UART Framing Error Interrupt Mask<br>Value Description<br>0 The FERIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the FERIS bit in the **UARTRIS** register is set. |
| 6 | RTIM | RW | 0 | UART Receive Time-Out Interrupt Mask<br>Value Description<br>0 The RTRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the RTRIS bit in the **UARTRIS** register is set. |
| 5 | TXIM | RW | 0 | UART Transmit Interrupt Mask<br>Value Description<br>0 The TXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the TXRIS bit in the **UARTRIS** register is set. |
| 4 | RXIM | RW | 0 | UART Receive Interrupt Mask<br>Value Description<br>0 The RXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the RXRIS bit in the **UARTRIS** register is set. |
| 3 | DSRIM | RW | 0 | UART Data Set Ready Modem Interrupt Mask<br>Value Description<br>0 The DSRRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the DSRRIS bit in the **UARTRIS** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | DCDIM | RW | 0 | UART Data Carrier Detect Modem Interrupt Mask<br>Value Description<br>0 The DCDRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the DCDRIS bit in the **UARTRIS** register is set. |
| 1 | CTSIM | RW | 0 | UART Clear to Send Modem Interrupt Mask<br>Value Description<br>0 The CTSRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the CTSRIS bit in the **UARTRIS** register is set. |
| 0 | RIIM | RW | 0 | UART Ring Indicator Modem Interrupt Mask<br>Value Description<br>0 The RIRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the RIRIS bit in the **UARTRIS** register is set. |

## Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x03C
Type RO, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | DMATXRIS | DMARXRIS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | 9BITRIS | EOTRIS | OERIS | BERIS | PERIS | FERIS | RTRIS | TXRIS | RXRIS | DSRRIS | DCDRIS | CTSRIS | RIRIS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | DMATXRIS | RO | 0 | Transmit DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The transmit DMA has completed.<br><br>This bit is cleared by writing a 1 to the DMATXIC bit in the **UARTICR** register. |
| 16 | DMARXRIS | RO | 0 | Receive DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The receive DMA has completed.<br><br>This bit is cleared by writing a 1 to the DMARXIC bit in the **UARTICR** register. |
| 15:13 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | 9BITRIS | RO | 0 | 9-Bit Mode Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A receive address match has occurred.<br><br>This bit is cleared by writing a 1 to the 9BITIC bit in the **UARTICR** register. |
| 11 | EOTRIS | RO | 0 | End of Transmission Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The last bit of all transmitted data and flags has left the serializer.<br><br>This bit is cleared by writing a 1 to the EOTIC bit in the **UARTICR** register. |
| 10 | OERIS | RO | 0 | UART Overrun Error Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 An overrun error has occurred.<br><br>This bit is cleared by writing a 1 to the OEIC bit in the **UARTICR** register. |
| 9 | BERIS | RO | 0 | UART Break Error Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A break error has occurred.<br><br>This bit is cleared by writing a 1 to the BEIC bit in the **UARTICR** register. |
| 8 | PERIS | RO | 0 | UART Parity Error Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A parity error has occurred.<br><br>This bit is cleared by writing a 1 to the PEIC bit in the **UARTICR** register. |
| 7 | FERIS | RO | 0 | UART Framing Error Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A framing error has occurred.<br><br>This bit is cleared by writing a 1 to the FEIC bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | RTRIS | RO | 0 | UART Receive Time-Out Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 A receive time out has occurred.<br><br>This bit is cleared by writing a 1 to the RTIC bit in the **UARTICR** register. For receive timeout, the RTIM bit in the **UARTIM** register must be set to see the RTRIS status. |
| 5 | TXRIS | RO | 0 | UART Transmit Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 If the EOT bit in the **UARTCTL** register is clear, the transmit FIFO level has passed through the condition defined in the **UARTIFLS** register.<br>If the EOT bit is set, the last bit of all transmitted data and flags has left the serializer.<br><br>This bit is cleared by writing a 1 to the TXIC bit in the **UARTICR** register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled. |
| 4 | RXRIS | RO | 0 | UART Receive Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The receive FIFO level has passed through the condition defined in the **UARTIFLS** register.<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled. |
| 3 | DSRRIS | RO | 0 | UART Data Set Ready Modem Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 Data Set Ready used for software flow control.<br><br>This bit is cleared by writing a 1 to the DSRIC bit in the **UARTICR** register. |
| 2 | DCDRIS | RO | 0 | UART Data Carrier Detect Modem Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 Data Carrier Detect used for software flow control.<br><br>This bit is cleared by writing a 1 to the DCDIC bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CTSRIS | RO | 0 | UART Clear to Send Modem Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 Clear to Send used for software flow control.<br><br>This bit is cleared by writing a 1 to the CTSIC bit in the **UARTICR** register. |
| 0 | RIRIS | RO | 0 | UART Ring Indicator Modem Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 Ring Indicator used for software flow control.<br><br>This bit is cleared by writing a 1 to the RIIC bit in the **UARTICR** register. |

## Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x040
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | DMATXMIS | DMARXMIS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | 9BITMIS | EOTMIS | OEMIS | BEMIS | PEMIS | FEMIS | RTMIS | TXMIS | RXMIS | DSRMIS | DCDMIS | CTSMIS | RIMIS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | DMATXMIS | RO | 0 | Transmit DMA Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the completion of the transmit DMA.<br><br>This bit is cleared by writing a 1 to the DMATXIC bit in the **UARTICR** register. |
| 16 | DMARXMIS | RO | 0 | Receive DMA Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the completion of the receive DMA.<br><br>This bit is cleared by writing a 1 to the DMARXIC bit in the **UARTICR** register. |
| 15:13 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | 9BITMIS | RO | 0 | 9-Bit Mode Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a receive address match.<br><br>This bit is cleared by writing a 1 to the 9BITIC bit in the **UARTICR** register. |
| 11 | EOTMIS | RO | 0 | End of Transmission Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the transmission of the last data bit.<br><br>This bit is cleared by writing a 1 to the EOTIC bit in the **UARTICR** register. |
| 10 | OEMIS | RO | 0 | UART Overrun Error Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to an overrun error.<br><br>This bit is cleared by writing a 1 to the OEIC bit in the **UARTICR** register. |
| 9 | BEMIS | RO | 0 | UART Break Error Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a break error.<br><br>This bit is cleared by writing a 1 to the BEIC bit in the **UARTICR** register. |
| 8 | PEMIS | RO | 0 | UART Parity Error Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a parity error.<br><br>This bit is cleared by writing a 1 to the PEIC bit in the **UARTICR** register. |
| 7 | FEMIS | RO | 0 | UART Framing Error Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a framing error.<br><br>This bit is cleared by writing a 1 to the FEIC bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | RTMIS | RO | 0 | UART Receive Time-Out Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to a receive time out.<br><br>This bit is cleared by writing a 1 to the RTIC bit in the **UARTICR** register. For receive timeout, the RTIM bit in the **UARTIM** register must be set to see the RTMIS status. |
| 5 | TXMIS | RO | 0 | UART Transmit Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set).<br><br>This bit is cleared by writing a 1 to the TXIC bit in the **UARTICR** register or by writing data to the transmit FIFO until it becomes greater than the trigger level, if the FIFO is enabled, or by writing a single byte if the FIFO is disabled. |
| 4 | RXMIS | RO | 0 | UART Receive Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to passing through the specified receive FIFO level.<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register or by reading data from the receive FIFO until it becomes less than the trigger level, if the FIFO is enabled, or by reading a single byte if the FIFO is disabled. |
| 3 | DSRMIS | RO | 0 | UART Data Set Ready Modem Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to Data Set Ready.<br><br>This bit is cleared by writing a 1 to the DSRIC bit in the **UARTICR** register. |
| 2 | DCDMIS | RO | 0 | UART Data Carrier Detect Modem Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to Data Carrier Detect.<br><br>This bit is cleared by writing a 1 to the DCDIC bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CTSMIS | RO | 0 | UART Clear to Send Modem Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to Clear to Send.<br><br>This bit is cleared by writing a 1 to the CTSIC bit in the **UARTICR** register. |
| 0 | RIMIS | RO | 0 | UART Ring Indicator Modem Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to Ring Indicator.<br><br>This bit is cleared by writing a 1 to the RIIC bit in the **UARTICR** register. |

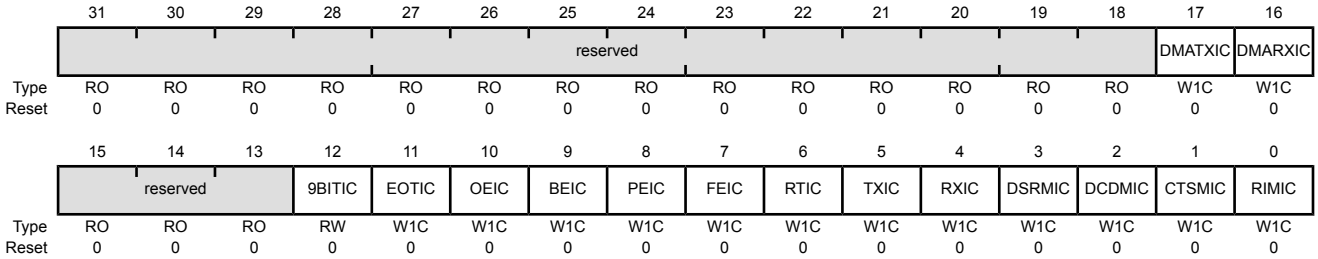
## Register 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x044
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0 | 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. |
| 17 | DMATXIC | W1C | 0 | Transmit DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMATXRIS bit in the **UARTRIS** register and the DMATXMIS bit in the **UARTMIS** register. |
| 16 | DMARXIC | W1C | 0 | Receive DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMARXRIS bit in the **UARTRIS** register and the DMARXMIS bit in the **UARTMIS** register. |
| 15:13 | reserved | RO | 0 | 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. |
| 12 | 9BITIC | RW | 0 | 9-Bit Mode Interrupt Clear<br>Writing a 1 to this bit clears the 9BITRIS bit in the **UARTRIS** register and the 9BITMIS bit in the **UARTMIS** register. |
| 11 | EOTIC | W1C | 0 | End of Transmission Interrupt Clear<br>Writing a 1 to this bit clears the EOTRIS bit in the **UARTRIS** register and the EOTMIS bit in the **UARTMIS** register. |
| 10 | OEIC | W1C | 0 | Overrun Error Interrupt Clear<br>Writing a 1 to this bit clears the OERIS bit in the **UARTRIS** register and the OEMIS bit in the **UARTMIS** register. |

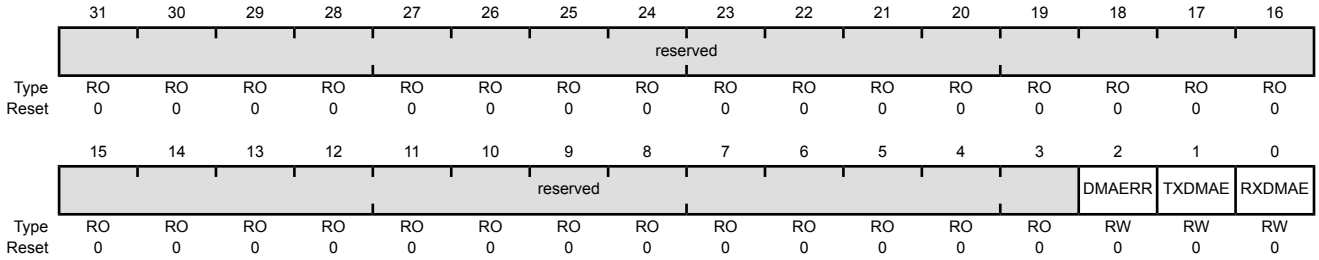
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | BEIC | W1C | 0 | Break Error Interrupt Clear<br>Writing a 1 to this bit clears the BERIS bit in the **UARTRIS** register and the BEMIS bit in the **UARTMIS** register. |
| 8 | PEIC | W1C | 0 | Parity Error Interrupt Clear<br>Writing a 1 to this bit clears the PERIS bit in the **UARTRIS** register and the PEMIS bit in the **UARTMIS** register. |
| 7 | FEIC | W1C | 0 | Framing Error Interrupt Clear<br>Writing a 1 to this bit clears the FERIS bit in the **UARTRIS** register and the FEMIS bit in the **UARTMIS** register. |
| 6 | RTIC | W1C | 0 | Receive Time-Out Interrupt Clear<br>Writing a 1 to this bit clears the RTRIS bit in the **UARTRIS** register and the RTMIS bit in the **UARTMIS** register. |
| 5 | TXIC | W1C | 0 | Transmit Interrupt Clear<br>Writing a 1 to this bit clears the TXRIS bit in the **UARTRIS** register and the TXMIS bit in the **UARTMIS** register. |
| 4 | RXIC | W1C | 0 | Receive Interrupt Clear<br>Writing a 1 to this bit clears the RXRIS bit in the **UARTRIS** register and the RXMIS bit in the **UARTMIS** register. |
| 3 | DSRMIC | W1C | 0 | UART Data Set Ready Modem Interrupt Clear<br>Writing a 1 to this bit clears the DSRRIS bit in the **UARTRIS** register and the DSRMIS bit in the **UARTMIS** register. |
| 2 | DCDMIC | W1C | 0 | UART Data Carrier Detect Modem Interrupt Clear<br>Writing a 1 to this bit clears the DCDRIS bit in the **UARTRIS** register and the DCDMIS bit in the **UARTMIS** register. |
| 1 | CTSMIC | W1C | 0 | UART Clear to Send Modem Interrupt Clear<br>Writing a 1 to this bit clears the CTSRIS bit in the **UARTRIS** register and the CTSMIS bit in the **UARTMIS** register. |
| 0 | RIMIC | W1C | 0 | UART Ring Indicator Modem Interrupt Clear<br>Writing a 1 to this bit clears the RIRIS bit in the **UARTRIS** register and the RIMIS bit in the **UARTMIS** register. |

## Register 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

UART DMA Control (UARTDMACTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x048
Type RW, reset 0x0000.0000



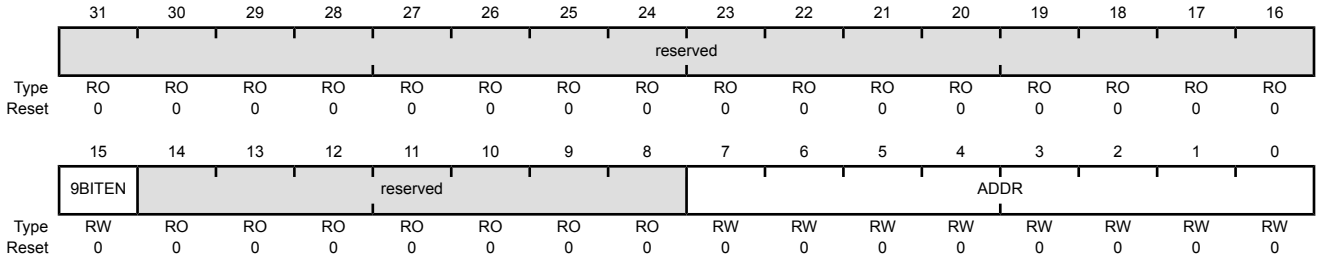

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x00000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DMAERR | RW | 0 | DMA on Error<br>Value Description<br>0 µDMA receive requests are unaffected when a receive error occurs.<br>1 µDMA receive requests are automatically disabled when a receive error occurs. |
| 1 | TXDMAE | RW | 0 | Transmit DMA Enable<br>Value Description<br>0 µDMA for the transmit FIFO is disabled.<br>1 µDMA for the transmit FIFO is enabled. |
| 0 | RXDMAE | RW | 0 | Receive DMA Enable<br>Value Description<br>0 µDMA for the receive FIFO is disabled.<br>1 µDMA for the receive FIFO is enabled. |

## Register 15: UART 9-Bit Self Address (UART9BITADDR), offset 0x0A4

The **UART9BITADDR** register is used to write the specific address that should be matched with the receiving byte when the 9-bit Address Mask (**UART9BITAMASK**) is set to 0xFF. This register is used in conjunction with **UART9BITAMASK** to form a match for address-byte received.

UART 9-Bit Self Address (UART9BITADDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x0A4
Type RW, reset 0x0000.0000



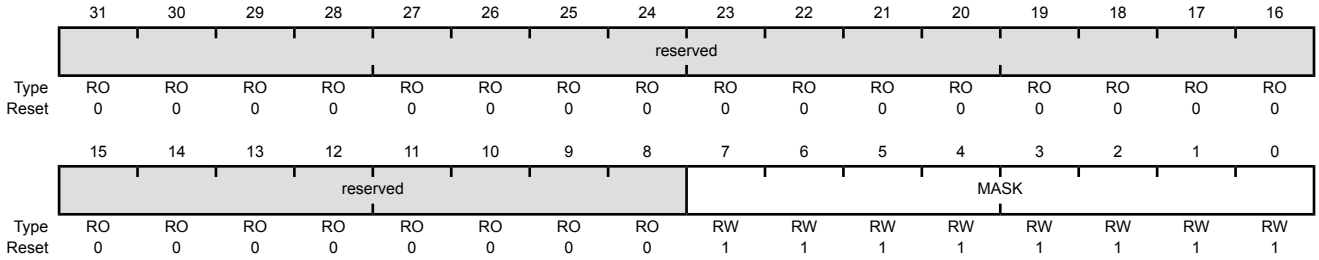

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15 | 9BITEN | RW | 0 | Enable 9-Bit Mode<br>Value Description<br>0 9-bit mode is disabled.<br>1 9-bit mode is enabled. |
| 14:8 | reserved | RO | 0x0 | 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. |
| 7:0 | ADDR | RW | 0x00 | Self Address for 9-Bit Mode<br>This field contains the address that should be matched when **UART9BITAMASK** is 0xFF. |

## Register 16: UART 9-Bit Self Address Mask (UART9BITAMASK), offset 0x0A8

The **UART9BITAMASK** register is used to enable the address mask for 9-bit mode. The address bits are masked to create a set of addresses to be matched with the received address byte.

UART 9-Bit Self Address Mask (UART9BITAMASK)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0x0A8
Type RW, reset 0x0000.00FF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x00 | 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. |
| 7:0 | MASK | RW | 0xFF | Self Address Mask for 9-Bit Mode<br>This field contains the address mask that creates a set of addresses that should be matched. |

## Register 17: UART Peripheral Properties (UARTPP), offset 0xFC0

The **UARTPP** register provides information regarding the properties of the UART module.

UART Peripheral Properties (UARTPP)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFC0
Type RO, reset 0x0000.000F

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | MSE | MS | NB | SC |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0 | 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. |
| 3 | MSE | RO | 0x1 | Modem Support Extended<br>Value Description<br>0 The UART module does not provide extended support for modem control.<br>1 The UART module provides extended support for modem control including UARTnDTR, UARTnDSR, UARTnDCD, and UARTnRI. |
| 2 | MS | RO | 0x1 | Modem Support<br>Value Description<br>0 The UART module does not provide support for modem control.<br>1 The UART module provides support for modem control including UARTnRTS and UARTnCTS. |
| 1 | NB | RO | 0x1 | 9-Bit Support<br>Value Description<br>0 The UART module does not provide support for the transmission of 9-bit data for RS-485 support.<br>1 The UART module provides support for the transmission of 9-bit data for RS-485 support. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | SC | RO | 0x1 | Smart Card Support |

| Value | Description |
|---|---|
| 0 | The UART module does not provide smart card support. |
| 1 | The UART module provides smart card support. |

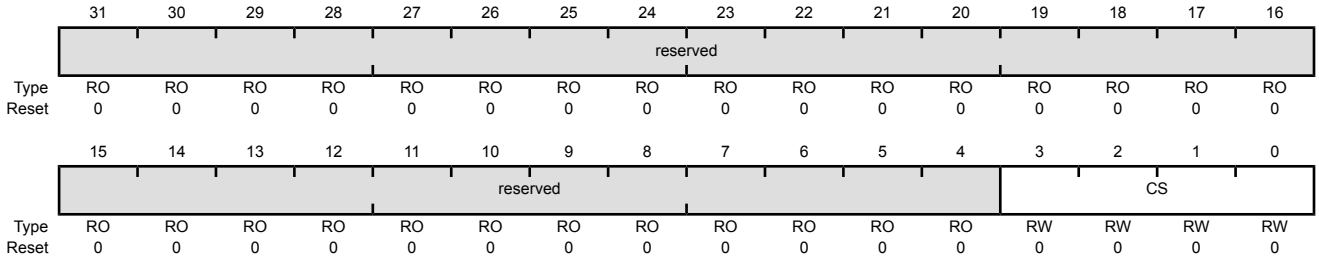
## Register 18: UART Clock Configuration (UARTCC), offset 0xFC8

The **UARTCC** register controls the baud clock source for the UART module. For more information, see the section called “Peripheral Clock Sources” on page 245.

**Note:** If the PIOSC is used for the UART baud clock, the system clock frequency must be at least 9 MHz in Run mode.

UART Clock Configuration (UARTCC)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFC8
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | CS | RW | 0 | UART Baud Clock Source<br>The following table specifies the source that generates for the UART baud clock: |

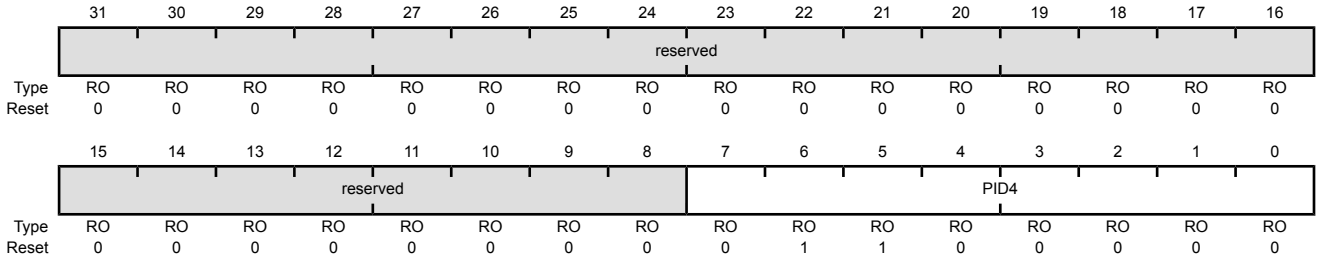
| Value | Description |
|---|---|
| 0x0 | System clock (based on clock source and divisor factor programmed in **RSCLKCFG** register in the System Control Module) |
| 0x1-0x4 | reserved |
| 0x5 | Alternate clock source as defined by **ALTCLKCFG** register in System Control Module. |
| 0x5-0xF | Reserved |

## Register 19: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFD0
Type RO, reset 0x0000.0060



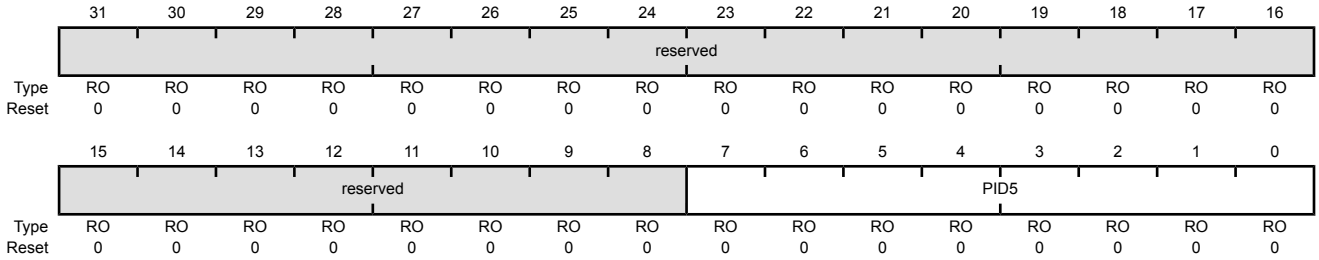

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x60 | UART Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 20: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFD4
Type RO, reset 0x0000.0000



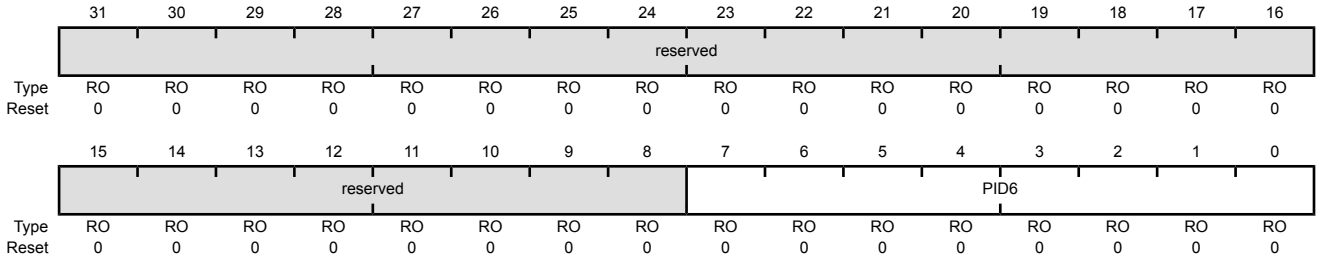

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | UART Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 21: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFD8
Type RO, reset 0x0000.0000



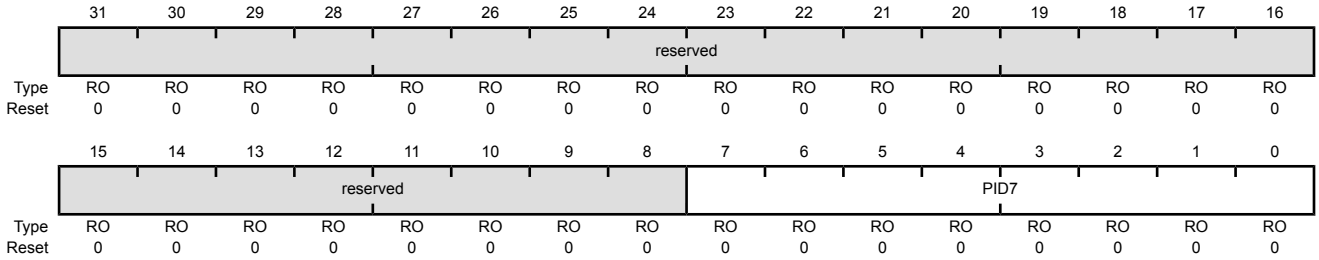

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | UART Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 22: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFDC
Type RO, reset 0x0000.0000



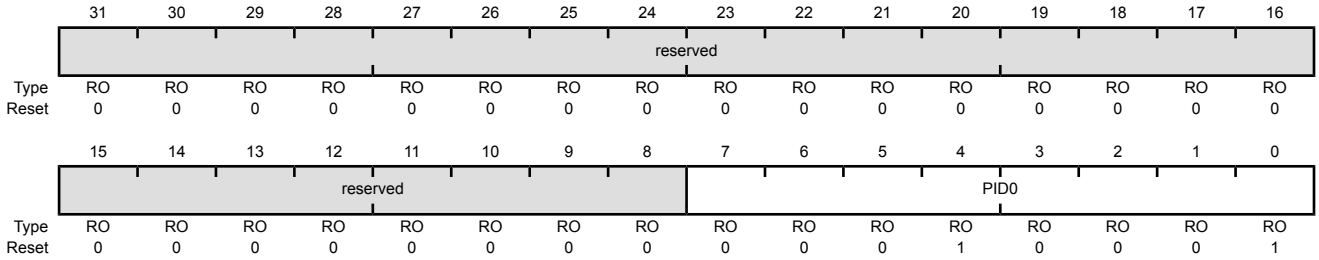

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | UART Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 23: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFE0
Type RO, reset 0x0000.0011



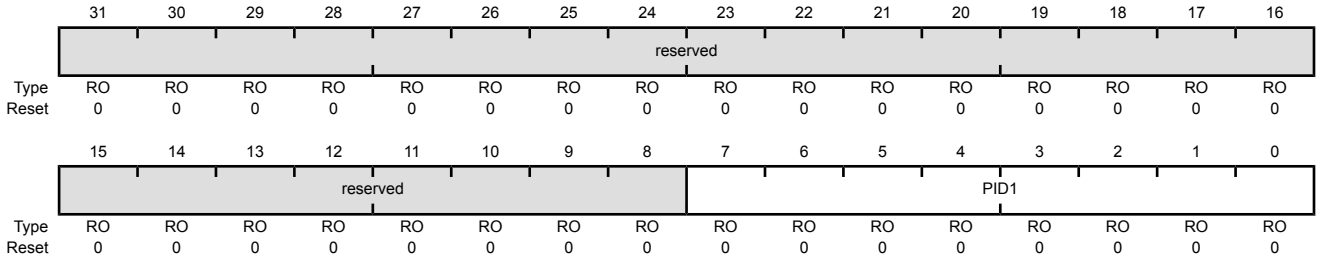

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x11 | UART Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 24: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFE4
Type RO, reset 0x0000.0000



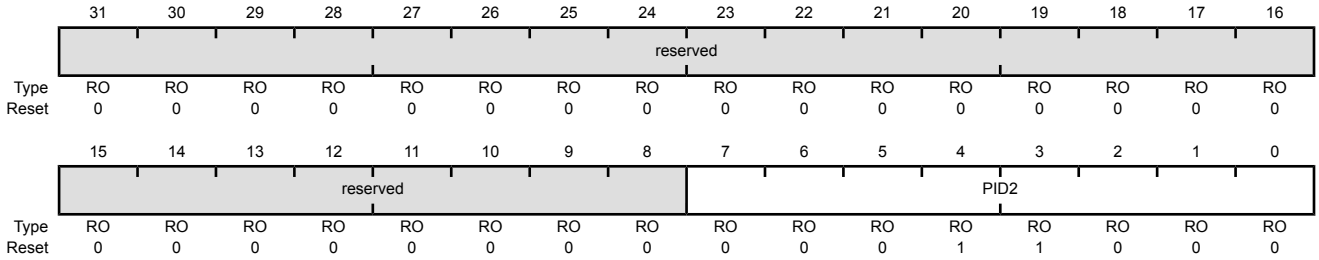

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x00 | UART Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 25: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFE8
Type RO, reset 0x0000.0018



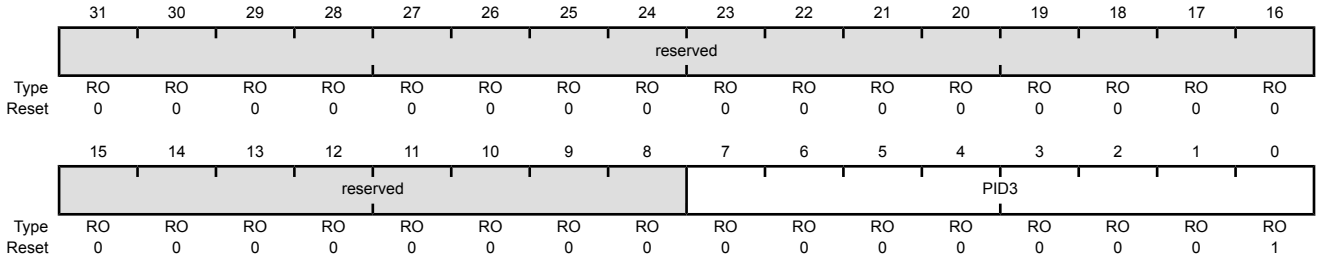

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | UART Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 26: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFEC
Type RO, reset 0x0000.0001



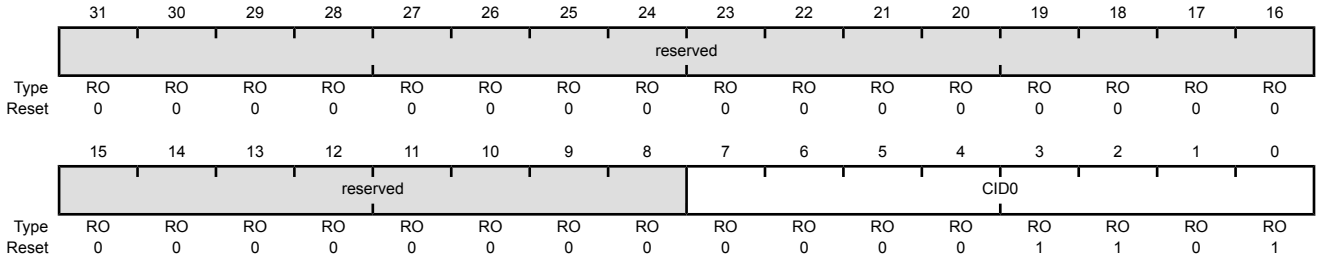

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | UART Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 27: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFF0
Type RO, reset 0x0000.000D



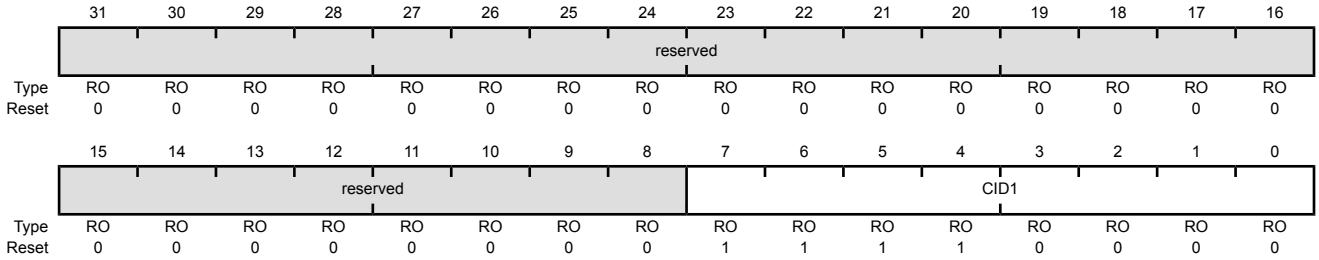

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | UART PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 28: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFF4
Type RO, reset 0x0000.00F0



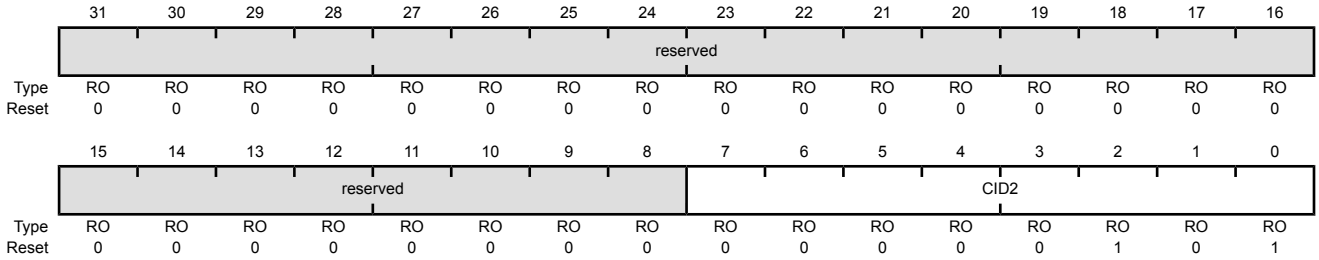

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | UART PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 29: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFF8
Type RO, reset 0x0000.0005



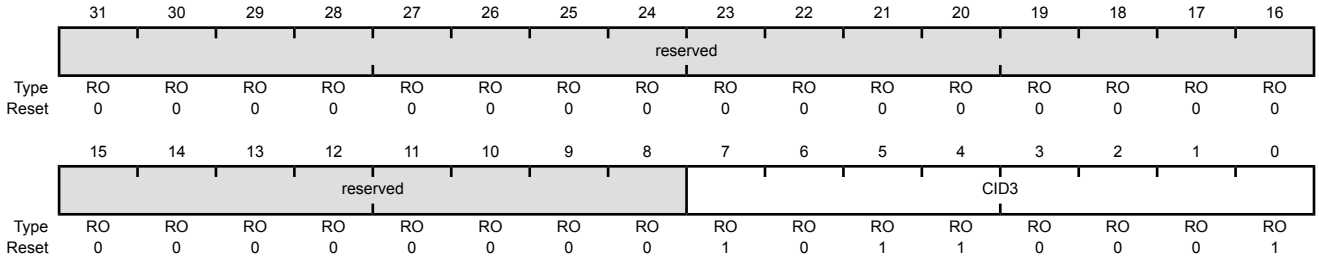

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | UART PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 30: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 3 (UARTPCellID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
UART3 base: 0x4000.F000
UART4 base: 0x4001.0000
UART5 base: 0x4001.1000
UART6 base: 0x4001.2000
UART7 base: 0x4001.3000
Offset 0xFFC
Type RO, reset 0x0000.00B1




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | CID3 | RO | 0xB1 | UART PrimeCell ID Register [31:24]<br>Provides software a standard cross-peripheral identification system. |

# 20 Quad Synchronous Serial Interface (QSSI)

The TM4C129XKCZAD microcontroller includes four Quad-Synchronous Serial Interface (QSSI) modules. All four of the modules support Advanced and Bi-SSI interfaces as well as a Quad-SSI enhancement to provide faster throughput of data. The QSSI module acts as a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, or Texas Instruments synchronous serial interfaces. The QSSI performs serial-to-parallel conversion on data received from a peripheral device. The transmit and receive paths are buffered with internal, independent FIFO memories allowing up to eight 16-bit values in Legacy mode and 8-bit values in Advanced, Bi-, and Quad-modes. The CPU can accesses data in these FIFOs as well as the QSSI's control and status information. A µDMA interface is also provided to allow the transmit and receive FIFOs to be programmed as source/destination addresses in the µDMA module.

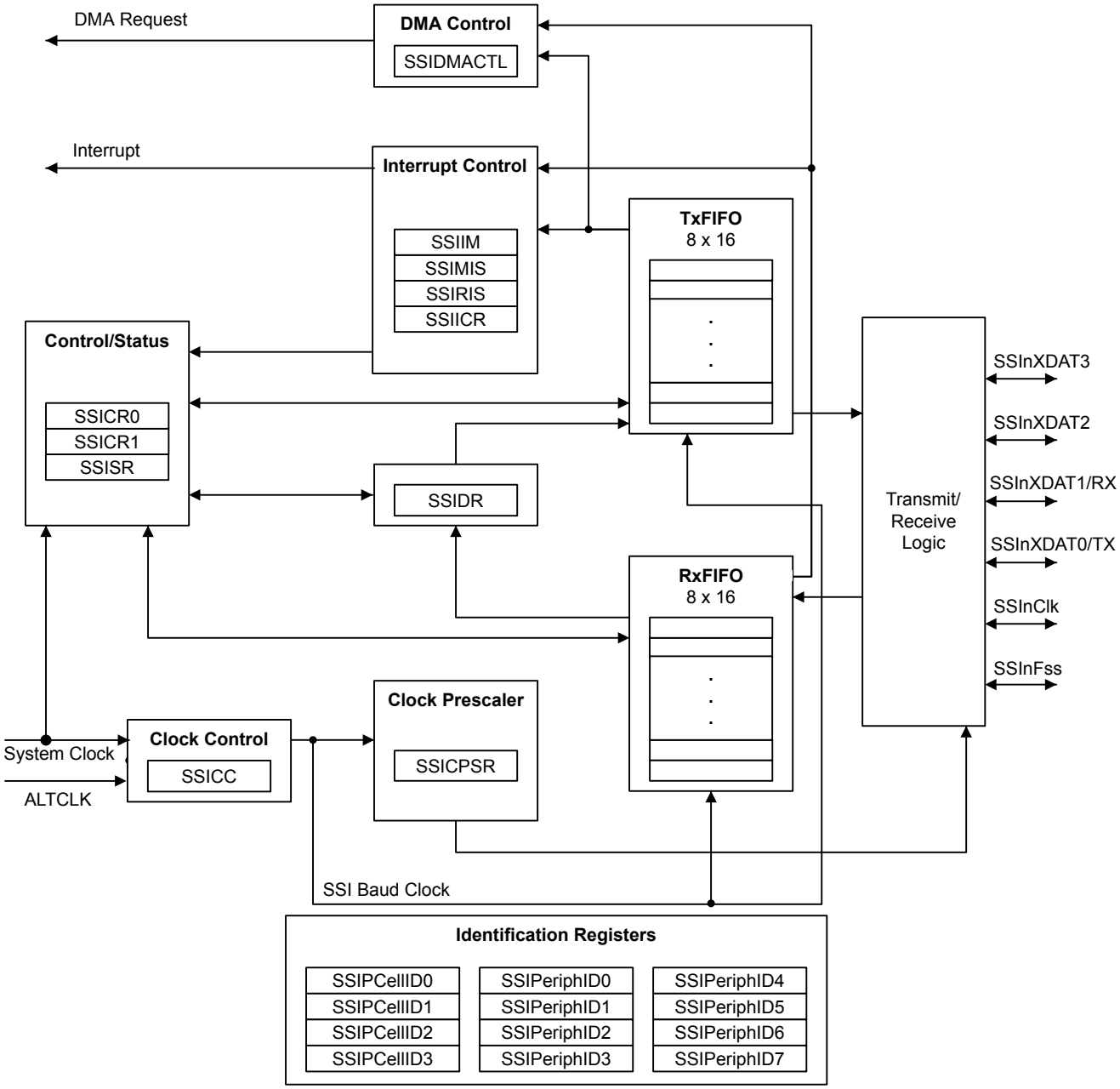
The TM4C129XKCZAD QSSI modules have the following features:

- Four QSSI channels with Advanced, Bi- and Quad-SSI functionality
- Programmable interface operation for Freescale SPI or Texas Instruments synchronous serial interfaces in Legacy Mode. Support for Freescale interface in Bi- and Quad-SSI mode.
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- 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 FIFO contains 4 entries
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when four or more entries are available to be written in the FIFO
  - Maskable µDMA interrupts for receive and transmit complete
- Global Alternate Clock (ALTCLK) resource or System Clock (SYSCLK) can be used to generate baud clock.

## 20.1 Block Diagram

The following figure below shows a block diagram of an QSSI module with Advanced, Bi- and Quad-SSI.

**Figure 20-1. QSSI Module with Advanced, Bi-SSI and Quad-SSI Support**




## 20.2 Signal Description

The following table lists the external signals of the QSSI module and describes the function of each. The QSSI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The "Pin Mux/Pin Assignment" column in the following table lists the possible GPIO pin placements for the QSSI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the QSSI function. The number in

parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the QSSI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772. Note that for the QSSI module, when operating in Legacy Mode, SSInXDAT0 functions as SSInTX and SSInXDAT1 functions as SSInRX.

**Table 20-1. SSI Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| SSI0Clk | T6 | PA2 (15) | I/O | TTL | SSI module 0 clock |
| SSI0Fss | U5 | PA3 (15) | I/O | TTL | SSI module 0 frame signal |
| SSI0XDAT0 | V4 | PA4 (15) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 0 (SSI0TX in Legacy SSI Mode). |
| SSI0XDAT1 | W4 | PA5 (15) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 1 (SSI0RX in Legacy SSI Mode). |
| SSI0XDAT2 | V5 | PA6 (13) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 2. |
| SSI0XDAT3 | R7 | PA7 (13) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 3. |
| SSI1Clk | B6 | PB5 (15) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | C6 | PB4 (15) | I/O | TTL | SSI module 1 frame signal. |
| SSI1XDAT0 | A5 | PE4 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 0 (SSI1TX in Legacy SSI Mode). |
| SSI1XDAT1 | B5 | PE5 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 1 (SSI1RX in Legacy SSI Mode). |
| SSI1XDAT2 | A4 | PD4 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 2. |
| SSI1XDAT3 | B4 | PD5 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 3. |
| SSI2Clk | D1<br>U14 | PD3 (15)<br>PG7 (15) | I/O | TTL | SSI module 2 clock. |
| SSI2Fss | D2<br>V12 | PD2 (15)<br>PG6 (15) | I/O | TTL | SSI module 2 frame signal. |
| SSI2XDAT0 | C1<br>K15 | PD1 (15)<br>PG5 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 0 (SSI2TX in Legacy SSI Mode). |
| SSI2XDAT1 | C2<br>K17 | PD0 (15)<br>PG4 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 1 (SSI2RX in Legacy SSI Mode). |
| SSI2XDAT2 | B2<br>M16 | PD7 (15)<br>PG3 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 2. |
| SSI2XDAT3 | B3<br>V11 | PD6 (15)<br>PG2 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 3. |
| SSI3Clk | T7<br>E3 | PF3 (14)<br>PQ0 (14) | I/O | TTL | SSI module 3 clock. |
| SSI3Fss | W6<br>E2 | PF2 (14)<br>PQ1 (14) | I/O | TTL | SSI module 3 frame signal. |
| SSI3XDAT0 | V6<br>H4 | PF1 (14)<br>PQ2 (14) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 0 (SSI3TX in Legacy SSI Mode). |
| SSI3XDAT1 | U6<br>M4 | PF0 (14)<br>PQ3 (14) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 1 (SSI3RX in Legacy SSI Mode). |
| SSI3XDAT2 | V7<br>D6 | PF4 (14)<br>PP0 (15) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 2. |
| SSI3XDAT3 | W7<br>D7 | PF5 (14)<br>PP1 (15) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 3. |

## 20.3 Functional Description

The QSSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The QSSI also supports the µDMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the µDMA module. µDMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 1412).

### 20.3.1 Bit Rate Generation

The QSSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (SysClk). The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 1404). The clock is further divided by a value from 1 to 256, which is 1 + SCR, where SCR is the value programmed in the **SSI Control 0 (SSICR0)** register (see page 1397).

The frequency of the output clock SSInClk is defined by:

```
SSInClk = SysClk / (CPSDVSR * (1 + SCR))
```

**Note:** SYSCLK or ALTCLK is used as the source for the SSInClk depending on how the CS field in the **SSI Clock Configuration (SSICC)** register is configured. For master legacy mode, the SYSCLK or ALTCLK must be at least two times faster than the SSInClk, with the restriction that SSInClk cannot be faster than 60 MHz. For slave mode, SYSCLK or ALTCLK must be at least 12 times faster than the SSInClk. In slave legacy mode, the maximum frequency of SSInClk is 10 MHz.

See “Synchronous Serial Interface (SSI)” on page 2144 to view legacy SSI and QSSI timing parameters.

### 20.3.2 FIFO Operation

#### 20.3.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 1401), 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 transmit FIFO prior to a legacy SSI serial conversion and transmission to the attached slave or master, respectively, through the SSInDAT0/SSInTX pin.

In slave mode, the legacy SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the 8th most recent value in the transmit FIFO. If less than 8 values have been written to the transmit FIFO since the SSI module clock was enabled using the Rn bit in the **RCGCSSI** register or if the QSSI is reset using the **SRSSI** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The QSSI can be configured to generate an interrupt or a µDMA request when the FIFO is empty.

**Note:** When operating in Legacy Mode, the QuadSSI's SSInXDAT0 signal functions as SSInTX.

#### 20.3.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data using the legacy serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register. If the receive FIFO is full when the master or slave receives new data, the data is held off until the receive FIFO has space.

The SSI only provides an SSIClk while transmitting data. When receiving data in master mode, a dummy write to the **SSIDR** register must be performed before any read so that the SSIClk can be properly received by the slave and allow data to be sent to the receive FIFO of the master.

When configured as a master or slave, serial data received through the SSInDAT1/SSInRX pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

**Note:** When operating in Legacy Mode, the QSSI's SSInXDAT1 signal functions as SSInRX.

### 20.3.3 Advanced, Bi- and Quad- SSI Function

Bi-SSI uses two data pins, SSInXDAT0 and SSInXDAT1, that can be configured to receive or transmit data. In Quad-SSI mode, SSInXDAT0, SSInXDAT1, SSInXDAT2 and SSInXDAT3 allow four bits of data to be received or transmitted at once. Note that in bi- and quad-SSI data transfers are only half-duplex.

By programming the MODE bits in the **SSICR1** register, Advanced, Bi- or Quad- SSI can be enabled. A direction bit, DIR, is provided to program the direction of operation during a Bi- or Quad SSI-transaction. Since Bi- and Quad-SSI cannot be full duplex, the DIR bit defines whether or not the RX FIFO is disabled. In Advanced operation, if the QSSI module TX (write) mode is enabled, the RX FIFO is automatically prevented from receiving any data. When Advanced SSI is in RX (read) mode, it operates as a full-duplex interface.

In Bi- and Quad-SSI mode, because only 8-bit data is allowed, the DSS bit field must be programmed to 0x7 in the **SSICR0** register before transferring data to the Rx and TX FIFOs. For a data transmit, the 8-bit data packet is placed in a TX FIFO entry bits [7:0] and the mode of operation is inserted in the three most significant bits of the TX FIFO entry. The mode of operation bits [15:13] in the TX FIFO are used by the QSSI module for configuring the data on the proper pins. The following modes that may be placed on bits [15:13] of the FIFO entry are:

- Bi-SSI mode (0x1)
- Quad-SSI mode (0x2)
- Advanced SSI mode (0x3)

When data is first written to the TX FIFO, a SSInFss is asserted low indicating the start of a frame. At the end of transmission, bit 12 of the last data entry in the TX FIFO signifies whether a a frame is ending. When the EOM bit is 1 it indicates a End of Message (EOM or STOP frame) and SSInFss is subsequently forced high. The EOM bit is cleared in the **SSICR1** register on the same clock that the write to TXFIFO is completed. An EOM bit value of 0 indicates no change in transmission. If TX FIFO is emptied and SSInFSS is still asserted low, it remains low but SSInCLK is not pulsed. Likewise, if SSInFss is high when the TX FIFO is empty, it remains high.

During a Bi-SSI transmit frame, data is shifted out by two bits and placed on the corresponding two SSInDATn pins. For a Quad-SSI transmit frame data is shifted out by four bits and placed on the corresponding four SSInDATn pins.

In Bi-, Quad- and Advanced SSI, the lower byte of the Rx FIFO contains received data. The upper byte contains no valid information.

**Note:** While the master is in Bi- or Quad-SSI mode, if the DSS bit in the **SSICR0** register is not set to 0x7, the QSSI module reverts to Legacy mode and behavior is not guaranteed.

The **SSICRI1** register bits DIR and MODE are used to program what operation is needed for the next data bytes that are being loaded into the FIFO. Table 20-2 on page 1383 shows available modes of operation:

**Table 20-2. QSSI Transaction Encodings**

| DIR | MODE | Operation |
|---|---|---|
| X | 0x0 | SSI Legacy operation supporting 4 to 16 data bits |
| 0 | 0x1 | Transmit (TX) Bi-SSI with 8-bits of packet data |
| 0 | 0x2 | Transmit (TX) Quad-SSI with 8-bits of packet data |
| 0 | 0x3 | Transmit (TX) Advanced SSI mode with 8-bits of packet data and write RX FIFO disabled |
| 1 | 0x1 | Receive (RX) Bi-SSI with 8-bits of packet data |
| 1 | 0x2 | Receive (RX) Quad-SSI with 8-bits of packet data |
| 1 | 0x3 | Full duplex Advance SSI with 8-bits of packet data |

**Note:** SPO = 0 and SPH =0 is the only frame structure allowed for Advanced, Bi- and Quad-mode.

Different transactions can follow one another in the FIFOs. The following transaction combinations are allowed:

- Legacy SSI mode (if configured for this mode, switching to any other alternate mode is not recommended)
- Advanced SSI mode followed by Bi-SSI mode
- Advanced SSI mode followed by Quad-SSI mode
- Advanced SSI mode followed by Bi-SSI mode followed by Advanced SSI mode
- Advanced SSI mode followed by Quad-SSI mode followed by Advanced SSI mode

Note that switching between Quad-SSI and Bi-SSI is not encouraged in a single transaction.

### 20.3.4 SSInFSS Function

For enhanced modes of operation, the SSInFss signal can be programmed to assert low at the start of each byte transfer for one clock or the entire frame. This is configured by programming the FSSHLDFRM bit in the **SSICR1** register. The EOM bit is also provided to signify end of frame transmission. This bit is embedded in the TXFIFO entry for use at the interface to deassert SSInFss at the appropriate time. The FSSHLDFRM bit can also be used when operating in 8-bit Legacy SSI mode.

The functionality of the FSSHLDFRM bit for both Legacy SSI mode and the enhanced modes are as follows:

**Table 20-3. SSInFss Functionality**

| Mode | FSSHLDFRM | Description |
|---|---|---|
| Legacy Mode | 0 | For Freescale format, with SPH = 0, the SSInFss signal is asserted low between continuous transfers. For SPH = 1, the SSInFss signal is deasserted (high) between continuous transfers.<br>For TI format, the SSInFss signal is deasserted (high) after every data transfer. |
| | 1 | For Freescale format with any SPH value, the SSInFss signal is forced high between continuous transfers; it is asserted low when there is available data in the Tx FIFO; otherwise it is forced high to be ready for a new frame |
| Advanced/Bi-/Quad-SSI Mode | 0 | SSInFss is asserted low after every byte of data |
| | 1 | New data written to the TX FIFO notifies SSInFss to assert low until the Tx FIFO is empty. |

### 20.3.5 High Speed Clock Operation

In master mode, QSSI module can enable a high speed clock by setting the HSCLKEN bit in the **SSI Control 1 (SSICR1)** register. In this mode of operation, SSInCLK from the QSSI master operation is reflected back as a loopback clock, HSPEEDCLK, to the QSSI module. This allows faster timing since the logic can can be used to adjust clock to external data relationships. HSPEEDCLK captures RX data in a separate register . This allows the time between the clock as seen by a remote device and the internal clock to match more closely.

Receive data is captured in a separate register sampled on loop-back clock (HSPEEDCLK) and the RX FIFO write control registered on HSPEEDCLK. If the HSCKEN = 1, the corresponding shift register and FIFO write enable will be selected for use. This supports faster QSSI master speed.

**Note:** For proper functionality of high speed mode, the HSCLKEN bit in the **SSICR1** register should be set before any SSI data transfer or after applying a reset to the QSSI module. In addition, the SSE bit must be set to 0x1 before the HSCLKEN bit is set.

### 20.3.6 Interrupts

The QSSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service (when the transmit FIFO is half full or less)
- Receive FIFO service (when the receive FIFO is half full or more)
- Receive FIFO time-out
- Receive FIFO overrun
- End of transmission
- Receive DMA transfer complete
- Transmit DMA transfer complete

All of the interrupt events are ORed together before being sent to the interrupt controller, so the QSSI generates a single interrupt request to the controller regardless of the number of active interrupts. Each of the seven individual maskable interrupts can be masked by clearing the appropriate bit in the **SSI Interrupt Mask (SSIIM)** register (see page 1405). Setting the appropriate mask bit enables the interrupt.

The individual outputs, along with a combined interrupt output, allow use of either a global interrupt service routine or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the **SSI Raw Interrupt Status (SSIRIS)** and **SSI Masked Interrupt Status (SSIMIS)** registers (see page 1407 and page 1409, respectively).

The RX FIFO has an associated time-out counter which starts to down count at the same time the RX FIFO is flagged as not empty by the RNE bit in the **SSISR** register. The counter is reset any time a new or next byte is written to the RX FIFO, thus the counter will continue to count down to zero unless there is new activity. The time-out period is 32 periods based on the period of SSInClk. When the counter reaches zero, a time-out interrupt bit, RTRIS, is set in the **SSIRIS** register. The time-out interrupt can be cleared by writing a 1 to the RTIC bit of the **SSI Interrupt Clear (SSIIC)** register or by emptying the RX FIFO. If the interrupt is cleared and there is residual data left in the RX FIFO or new data entries have been written, the timer count down initiates and the interrupt will be reasserted after 32 periods have been counted.

The End-of-Transmission (EOT) interrupt indicates that the data has been transmitted completely and is only valid for Master mode devices/operations. This interrupt can be used to indicate when it is safe to turn off the QSSI module clock or enter sleep mode. In addition, because transmitted data and received data complete at exactly the same time, the interrupt can also indicate that read data is ready immediately, without waiting for the receive FIFO time-out period to complete.

**Note:** In Freescale SPI mode only, a condition can be created where an EOT interrupt is generated for every byte transferred even if the FIFO is full. If the the µDMA has been configured to transfer data from this QSSI to a Master QSSI on the device using external loopback, an EOT interrupt is generated by the QSSI slave for every byte even if the FIFO is full.

### 20.3.7 Frame Formats

Each data frame is between 4 and 16 bits long in Legacy mode and 8-bits in Advanced/Bi-/Quad-SSI mode and is transmitted starting with the MSB. There are two basic frame types that can be selected by programming the FRF bit in the **SSICR0** register:

- Texas Instruments synchronous serial
- Freescale SPI

**Note:** Advanced, Bi- and Quad-SSI modules only supports Freescale mode when SPH=0; SPO=0 and DDS=0x8 in the **SSI Control 0 (SSICR0)** register.

For both formats, the serial clock (SSInClk) is held inactive while the QSSI is idle, and SSInClk transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSInClk is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI frame format, the serial frame (SSInFss) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the SSInFss pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the QSSI and the off-chip slave device drive their output data on the rising edge of SSInClk and latch data from the other device on the falling edge.

The following table gives a synopsis of the features supported in each frame format when operating in Legacy Mode:

**Table 20-4. Legacy Mode TI, Freescale SPI Frame Format Features**

| Feature | TI Mode | Freescale SPI Mode |
|---|---|---|
| Frame Hold | Not Available | Available |
| High Speed (Master RX Only) | Not Available | Available |
| SPO/SPH Configuration | Not Available | Available and can be used in combination with Frame Hold and High Speed Mode |
| Frequency (system clock : SSInCLK) | Master 1:2<br>Slave 1:12 | Master 1:2<br>Slave 1:12 |

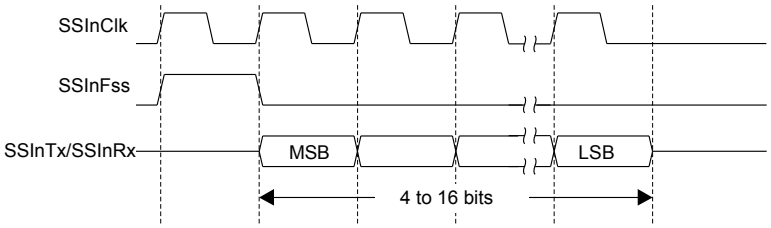
For Advanced, Bi- and Quad-SSI modes using the Freescale SPI Format or the Bi- and Quad-SSI modes using the TI format, the following features are supported:

- Frame Hold
- High Speed (Master RX Only)
- SPO/SPH Configuration with SPO=0 and SPH=0 only allowed
- Frequency (system clock : SSInCLK):
  – Master 1:2
  – Slave 1:12

### 20.3.7.1 Texas Instruments Synchronous Serial Frame Format

Figure 20-2 on page 1386 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

**Figure 20-2. TI Synchronous Serial Frame Format (Single Transfer)**

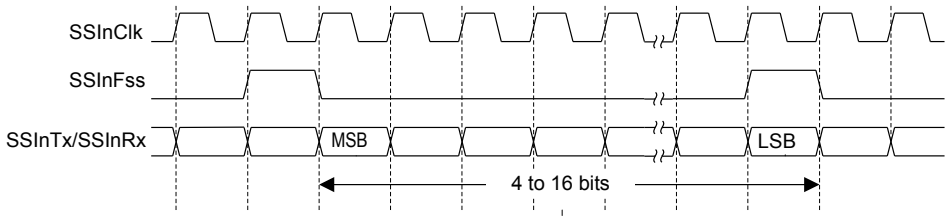



In this mode, SSInClk and SSInFss are forced Low, and the transmit data line SSInDAT0/SSInTX is tristated whenever the QSSI is idle. Once the bottom entry of the transmit FIFO contains data, SSInFss is pulsed High for one SSInClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSInClk, the MSB of the 4 to 16-bit data frame is shifted out on the SSInDAT0/SSInTX pin. Likewise, the MSB of the received data is shifted onto the SSInDAT1/SSInRX pin by the off-chip serial slave device.

Both the QSSI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of SSInClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSInClk after the LSB has been latched.

Figure 20-3 on page 1387 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

**Figure 20-3. TI Synchronous Serial Frame Format (Continuous Transfer)**




### 20.3.7.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSInFss signal behaves as a slave select. If operating in Legacy Mode and using the Freescale SPI Frame Format, the inactive state and phase of the SSInClk signal are programmable through the SPO and SPH bits in the **SSICR0** control register. If operating in Advanced/Bi-/Quad-SSI mode, the SP0 and SPH bits must be programmed to 0.

#### *SPO Clock Polarity Bit*

When the SPO clock polarity control bit is clear, it produces a steady state Low value on the SSInClk pin. If the SPO bit is set, a steady state High value is placed on the SSInClk pin when data is not being transferred.

#### *SPH Phase Control Bit*

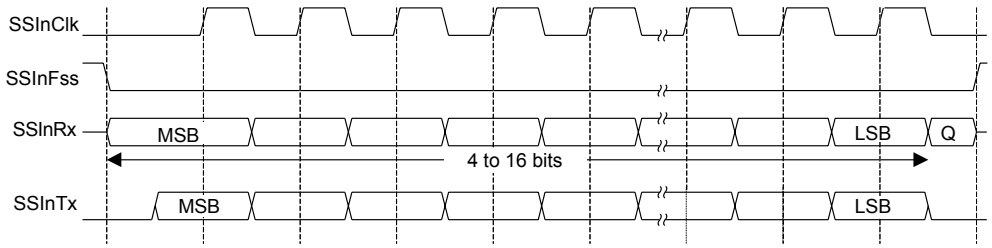
The SPH phase control bit selects the clock edge that captures data and allows it to change state. The state of this bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is clear, data is captured on the first clock edge transition. If the SPH bit is set, data is captured on the second clock edge transition.

### 20.3.7.3 Freescale SPI Frame Format with SPO=0 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 20-4 on page 1388 and Figure 20-5 on page 1388.

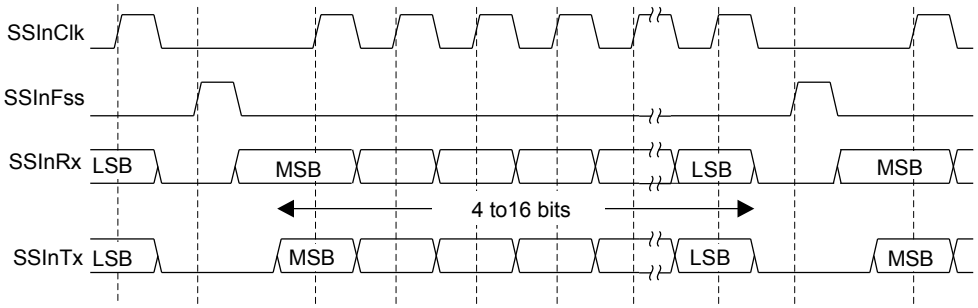
**Note:** This is the only Freescale SPI frame format configuration that can be used when operating in Advanced/Bi-/Quad-SSI mode.

**Figure 20-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0**




**Note:** Q is undefined.

**Figure 20-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0**




In this configuration, during idle periods:

- SSInClk is forced Low
- SSInFss is forced High
- The transmit data line SSInDAT0/SSInTX is tristated
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low, causing slave data to be enabled onto the SSInDAT1/SSInRX input line of the master. The master SSInDAT0/SSInTX output pad is enabled.

One half SSInClk period later, valid master data is transferred to the SSInDAT0/SSInTX pin. Once both the master and slave data have been set, the SSInClk master clock pin goes High after one additional half SSInClk period.

The data is now captured on the rising and propagated on the falling edges of the SSInClk signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

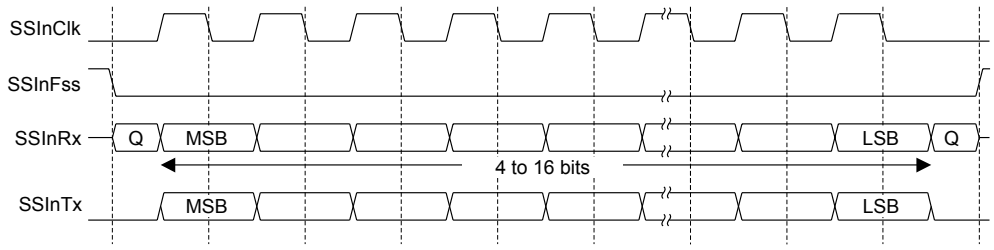
However, in the case of continuous back-to-back transmissions, the SSInFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSInFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSInFss pin is returned to its idle state one SSInClk period after the last bit has been captured.

### 20.3.7.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 20-6 on page 1389, which covers both single and continuous transfers.

**Note:** This Freescale SPI frame format configuration is only available when operating in Legacy SSI mode of operation.

**Figure 20-6. Freescale SPI Frame Format with SPO=0 and SPH=1**




**Note:** Q is undefined.

In this configuration, during idle periods:

- SSInClk is forced Low
- SSInFss is forced High
- The transmit data line SSInDAT0/SSInTX is tristated
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low. The master SSInDAT0/SSInTX output is enabled. After an additional one-half SSInClk period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the SSInClk is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSInClk signal.

In the case of a single word transfer, after all bits have been transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

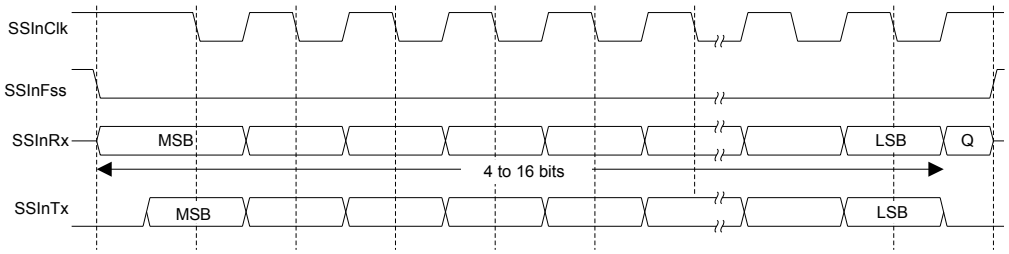
For continuous back-to-back transfers, the SSInFss pin is held Low between successive data words, and termination is the same as that of the single word transfer.

### 20.3.7.5 Freescale SPI Frame Format with SPO=1 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 20-7 on page 1390 and Figure 20-8 on page 1390.

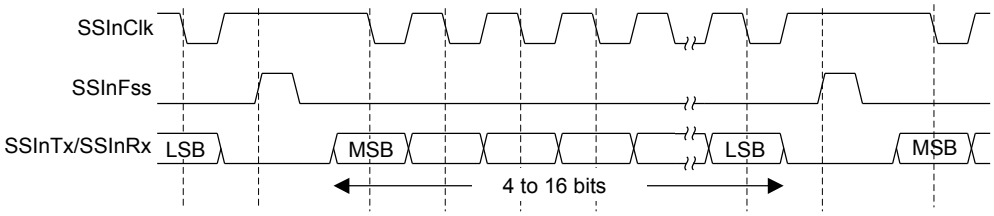
**Note:** This Freescale SPI frame format configuration is only available when operating in Legacy SSI mode of operation.

**Figure 20-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0**




**Note:** Q is undefined.

**Figure 20-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0**




In this configuration, during idle periods:

- SSInClk is forced High
- SSInFss is forced High
- The transmit data line SSInDAT0/SSInTX is tristated
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low, causing slave data to be immediately transferred onto the SSInDAT1/SSInRX line of the master. The master SSInDAT0/SSInTX output pad is enabled.

One-half period later, valid master data is transferred to the SSInDAT0/SSInTX line. Once both the master and slave data have been set, the SSInClk master clock pin becomes Low after one additional half SSInClk period, meaning that data is captured on the falling edges and propagated on the rising edges of the SSInClk signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSInFss line is returned to its idle High state one SSInClk period after the last bit has been captured.

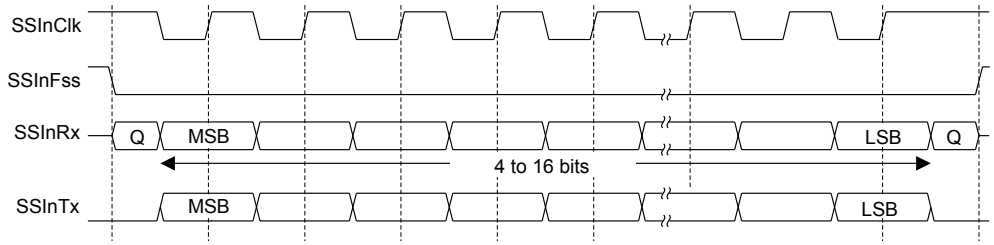
However, in the case of continuous back-to-back transmissions, the SSInFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSInFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSInFss pin is returned to its idle state one SSInClk period after the last bit has been captured.

### 20.3.7.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 20-9 on page 1391, which covers both single and continuous transfers.

**Note:** This Freescale SPI frame format configuration is only available when operating in Legacy SSI mode of operation.

**Figure 20-9. Freescale SPI Frame Format with SPO=1 and SPH=1**




**Note:** Q is undefined.

In this configuration, during idle periods:

- SSInClk is forced High
- SSInFss is forced High
- The transmit data line SSInDAT0/SSInTX is tristated
- When the QSSI is configured as a master, it enables the SSInClk pad
- When the QSSI is configured as a slave, it disables the SSInClk pad

If the QSSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSInFss master signal being driven Low. The master SSInDAT0/SSInTX output pad is enabled. After an additional one-half SSInClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSInClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSInClk signal.

After all bits have been transferred, in the case of a single word transmission, the SSInFss line is returned to its idle high state one SSInClk period after the last bit has been captured.

For continuous back-to-back transmissions, the SSInFss pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSInFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

### 20.3.8 DMA Operation

The QSSI peripheral provides an interface to the µDMA controller with separate channels for transmit and receive. The µDMA operation of the QSSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When µDMA operation is enabled, the QSSI 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 receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever at least one empty location is in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst µDMA transfer requests are handled automatically by the µDMA controller depending how the µDMA channel is configured.

To enable µDMA operation for the receive channel, the RXDMAE bit of the **DMA Control (SSIDMACTL)** register should be set after configuring the µDMA. To enable µDMA operation for the transmit channel, the TXDMAE bit of **SSIDMACTL** should be set after configuring the µDMA.

If the µDMA is enabled and has completed a data transfer from the Tx FIFO, the DMATXRIS bit is set in the **SSIRIS** register and cannot be cleared by setting the DMATXIC bit in the **SSI Interrupt Clear (SSIICR)** register. In the DMA Completion Interrupt Service Routine, software must disable the µDMA transmit enable to the SSI by clearing the TXDMAE bit in the **QSSI DMA Control (SSIDMACTL)** register and then setting the DMATXIC bit in the **SSIICR** register. This clears the DMA completion interrupt. When the µDMA is needed to transmit more data, the TXDMAE bit must be set (enabled) again.

If a data transfer by the µDMA from the Rx FIFO completes, the DMARXRIS bit is set. The EOT bit in the **SSIRIS** register is also provided to indicate when the Tx FIFO is empty and the last bit has been transmitted out of the serializer

**Note:** Wait states are inserted at every byte transfer when using Bi- or Quad-SSI modes as a master with the µDMA at SSICLK frequencies greater than 1/6 of the system clock. These wait states are because of arbitration stall cycles from the µDMA accesses to SRAM and increased output throughput from the SSI.

See “Micro Direct Memory Access (µDMA)” on page 708 for more details about programming the µDMA controller.

## 20.4 Initialization and Configuration

To enable and initialize the QSSI, the following steps are necessary:

1. Enable the QSSI module using the **RCGCSSI** register (see page 407).
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register (see page 399). To find out which GPIO port to enable, refer to Table 31-5 on page 2081.
3. Set the GPIO AFSEL bits for the appropriate pins (see page 803). To determine which GPIOs to configure, see Table 31-4 on page 2064.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the QSSI signals to the appropriate pins. See page 820 and Table 31-5 on page 2081.
5. Program the **GPIODEN** register to enable the pin's digital function. In addition, the drive strength, drain select and pull-up/pull-down functions must be configured. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 772 for more information.

   **Note:** Pull-ups can be used to avoid unnecessary toggles on the QSSI pins, which can take the slave to a wrong state. In addition, if the SSIClk signal is programmed to steady state High through the SPO bit in the **SSICR0** register, then software must also configure the GPIO port pin corresponding to the SSInClk signal as a pull-up in the **GPIO Pull-Up Select (GPIOPUR)** register.

For each of the frame formats, the QSSI is configured using the following steps:

1. If initializing out of reset, ensure that the SSE bit in the **SSICR1** register is clear before making any configuration changes. Otherwise, configuration changes for Advanced SSI can be made while the SSE bit is set.
2. Select whether the QSSI is a master or slave:
   a. For master operations, set the **SSICR1** register to 0x0000.0000.
   b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
   c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the QSSI clock source by writing to the **SSICC** register.
4. Configure the clock prescale divisor by writing the **SSICPSR** register.
5. Write the **SSICR0** register with the following configuration:
   - Serial clock rate (SCR)
   - Desired clock phase/polarity, if using Freescale SPI mode (SPH and SPO)
   - The protocol mode: Freescale SPI or TI SSF
   - The data size (DSS)
6. Optionally, configure the SSI module for µDMA use with the following steps:
   a. Configure a µDMA for SSI use. See “Micro Direct Memory Access (µDMA)” on page 708 for more information.
   b. Enable the SSI Module's TX FIFO or RX FIFO by setting the TXDMAE or RXDMAE bit in the **SSIDMACTL** register.

**c.** Optionally, enable the µDMA completion interrupt by setting the DMATXIM or DMARXIM bit in the **SSIIM** register.

**Note:** For a TX DMA completion interrupt, software must disable the µDMA transmit enable to the SSI by clearing the TXDMAE bit in the **QSSI DMA Control (SSIDMACTL)** register and then setting the DMATXIC bit in the **SSIICR** register. This clears the DMA completion interrupt. When the µDMA is needed to transmit more data, the TXDMAE bit must be set (enabled) again.

**7.** If this is the first initialization out of reset, enable the QSSI by setting the SSE bit in the **SSICR1** register.

As an example, assume the QSSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SPO=1, SPH=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

```
SSInClk = SysClk / (CPSDVSR * (1 + SCR))
1x10^6  = 20x10^6 / (CPSDVSR * (1 + SCR))
```

In this case, if CPSDVSR=0x2, SCR must be 0x9.

The configuration sequence would be as follows:

**1.** Ensure that the SSE bit in the **SSICR1** register is clear.

**2.** Write the **SSICR1** register with a value of 0x0000.0000.

**3.** Write the **SSICPSR** register with a value of 0x0000.0002.

**4.** Write the **SSICR0** register with a value of 0x0000.09C7.

**5.** The QSSI is then enabled by setting the SSE bit in the **SSICR1** register.

### 20.4.1 Enhanced Mode Configuration

If the QSSI module supports the Advanced/Bi-/Quad features, then these modes can be enabled after initializing the QSSI module. Below is an example of configuring the QSSI to transmit two data bytes in Advanced SSI mode followed by 2 bytes in Bi-SSI mode:

**1.** Set the MODE bit to 0x3, and the FSSHLDFM bit to 1 in the **SSICR1** register. To operate in the master mode, program the MS bit to 0. Program the remaining bits in the **SSICR0** and **SSICR1** register to relevant values.

**2.** Write one data byte to the TX FIFO; set the EOM bit to 1 and write the second data byte to the Tx FIFO.

3. Set the MODE bit to 0x1 and the FSSHLDFM bit to 1 in the **SSICR1** register. To operate in the master mode, program the MS bit to 0. Program the remaining bits in the **SSICR0** and **SSICR1** register to relevant values.
4. Fill the Tx FIFO with one data byte.
5. Set the EOM bit in the **SSICR1** register.
6. Fill the Tx FIFO with one data byte.

## 20.5 Register Map

Table 20-5 on page 1395 lists the QSSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that QSSI module's base address:

- QSSI0: 0x4000.8000
- QSSI1: 0x4000.9000
- QSSI2: 0x4000.A000
- QSSI3: 0x4000.B000

Note that the QSSI module clock must be enabled before the registers can be programmed (see page 407). The Rn bit of the **PRSSI** register must be read as 0x1 before any QSSI module registers are accessed.

**Table 20-5. SSI Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | SSICR0 | RW | 0x0000.0000 | QSSI Control 0 | 1397 |
| 0x004 | SSICR1 | RW | 0x0000.0000 | QSSI Control 1 | 1399 |
| 0x008 | SSIDR | RW | 0x0000.0000 | QSSI Data | 1401 |
| 0x00C | SSISR | RO | 0x0000.0003 | QSSI Status | 1402 |
| 0x010 | SSICPSR | RW | 0x0000.0000 | QSSI Clock Prescale | 1404 |
| 0x014 | SSIIM | RW | 0x0000.0000 | QSSI Interrupt Mask | 1405 |
| 0x018 | SSIRIS | RO | 0x0000.0008 | QSSI Raw Interrupt Status | 1407 |
| 0x01C | SSIMIS | RO | 0x0000.0000 | QSSI Masked Interrupt Status | 1409 |
| 0x020 | SSIICR | W1C | 0x0000.0000 | QSSI Interrupt Clear | 1411 |
| 0x024 | SSIDMACTL | RW | 0x0000.0000 | QSSI DMA Control | 1412 |
| 0xFC0 | SSIPP | RO | 0x0000.000D | QSSI Peripheral Properties | 1413 |
| 0xFC8 | SSICC | RW | 0x0000.0000 | QSSI Clock Configuration | 1414 |
| 0xFD0 | SSIPeriphID4 | RO | 0x0000.0000 | QSSI Peripheral Identification 4 | 1415 |
| 0xFD4 | SSIPeriphID5 | RO | 0x0000.0000 | QSSI Peripheral Identification 5 | 1416 |
| 0xFD8 | SSIPeriphID6 | RO | 0x0000.0000 | QSSI Peripheral Identification 6 | 1417 |
| 0xFDC | SSIPeriphID7 | RO | 0x0000.0000 | QSSI Peripheral Identification 7 | 1418 |
| 0xFE0 | SSIPeriphID0 | RO | 0x0000.0022 | QSSI Peripheral Identification 0 | 1419 |

**Table 20-5. SSI Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xFE4 | SSIPeriphID1 | RO | 0x0000.0000 | QSSI Peripheral Identification 1 | 1420 |
| 0xFE8 | SSIPeriphID2 | RO | 0x0000.0018 | QSSI Peripheral Identification 2 | 1421 |
| 0xFEC | SSIPeriphID3 | RO | 0x0000.0001 | QSSI Peripheral Identification 3 | 1422 |
| 0xFF0 | SSIPCellID0 | RO | 0x0000.000D | QSSI PrimeCell Identification 0 | 1423 |
| 0xFF4 | SSIPCellID1 | RO | 0x0000.00F0 | QSSI PrimeCell Identification 1 | 1424 |
| 0xFF8 | SSIPCellID2 | RO | 0x0000.0005 | QSSI PrimeCell Identification 2 | 1425 |
| 0xFFC | SSIPCellID3 | RO | 0x0000.00B1 | QSSI PrimeCell Identification 3 | 1426 |

## 20.6 Register Descriptions

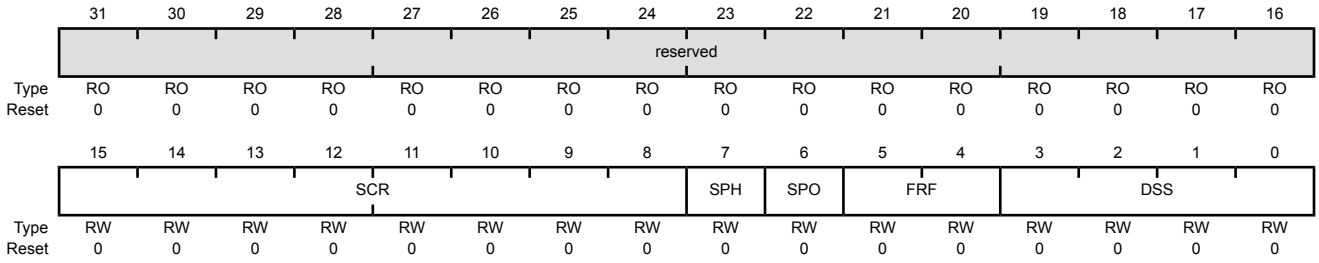
The remainder of this section lists and describes the QSSI registers, in numerical order by address offset.

## Register 1: QSSI Control 0 (SSICR0), offset 0x000

The **SSICR0** register contains bit fields that control various functions within the QSSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

QSSI Control 0 (SSICR0)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:8 | SCR | RW | 0x00 | QSSI Serial Clock Rate<br>This bit field is used to generate the transmit and receive bit rate of the QSSI. The bit rate is:<br>BR=SysClk/(CPSDVSR * (1 + SCR))<br>where CPSDVSR is an even value from 2-254 programmed in the **SSICPSR** register, and SCR is a value from 0-255. |
| 7 | SPH | RW | 0 | QSSI Serial Clock Phase<br>This bit is only applicable to the Freescale SPI Format.<br>The SPH control bit selects the clock edge that captures data and allows it to change state. 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.<br>Value Description<br>0 Data is captured on the first clock edge transition.<br>1 Data is captured on the second clock edge transition. |
| 6 | SPO | RW | 0 | QSSI Serial Clock Polarity<br>Value Description<br>0 A steady state Low value is placed on the SSInClk pin.<br>1 A steady state High value is placed on the SSInClk pin when data is not being transferred.<br>**Note:** If this bit is set, then software must also configure the GPIO port pin corresponding to the SSInClk signal as a pull-up in the **GPIO Pull-Up Select (GPIOPUR)** register. |

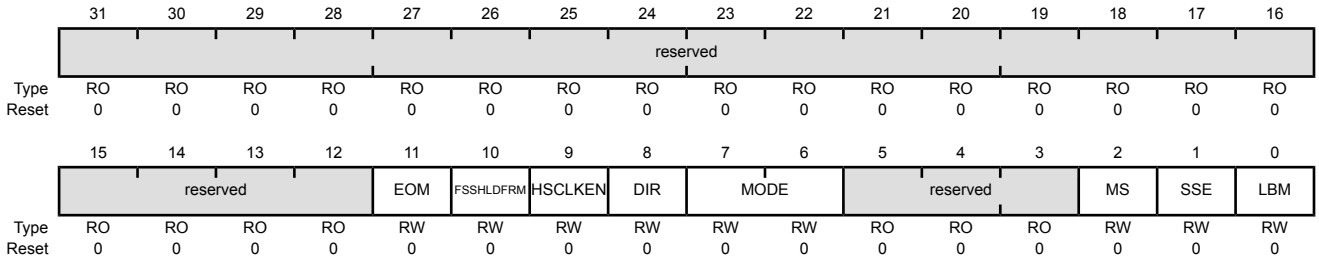
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | FRF | RW | 0x0 | QSSI Frame Format Select<br>**Note:** When operating in Advanced/Bi-/Quad-SSI mode these bits must be programmed to 0x0 (Freescale SPI Frame Format).<br>Value Frame Format<br>0x0 Freescale SPI Frame Format<br>0x1 Texas Instruments Synchronous Serial Frame Format<br>0x2-0x3 Reserved |
| 3:0 | DSS | RW | 0x0 | QSSI Data Size Select<br>**Note:** When operating in Advanced, Bi- or Quad-SSI, data size can only be 8-bit. All other fields will be ignored.<br>Value Data Size<br>0x0-0x2 Reserved<br>0x3 4-bit data<br>0x4 5-bit data<br>0x5 6-bit data<br>0x6 7-bit data<br>0x7 8-bit data<br>0x8 9-bit data<br>0x9 10-bit data<br>0xA 11-bit data<br>0xB 12-bit data<br>0xC 13-bit data<br>0xD 14-bit data<br>0xE 15-bit data<br>0xF 16-bit data |

## Register 2: QSSI Control 1 (SSICR1), offset 0x004

The **SSICR1** register contains bit fields that control various functions within the QSSI module. Master and slave mode functionality is controlled by this register.

QSSI Control 1 (SSICR1)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x004
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |
| 11 | EOM | RW | 0 | Stop Frame (End of Message)<br>This bit is applicable when MODE is set to Advanced, Bi- or Quad- SSI. This bit is inserted into bit 12 of the TXFIFO data entry by the QSSI module.<br>Value Description<br>0 No change is transmission status.<br>1 End of message (Stop Frame). |
| 10 | FSSHLDFRM | RW | 0 | FSS Hold Frame<br>Value Description<br>0 Pulse SSInFss at every byte (the DSS bit in the **SSICR0** register must be set to 0x7 (data size 8 bits) in this configuration)<br>1 Hold SSInFss for the whole frame |
| 9 | HSCLKEN | RW | 0 | High Speed Clock Enable<br>High speed clock enable is available only when operating as a master.<br>Value Description<br>0 Use Input Clock<br>1 Use High Speed Clock<br>**Note:** For proper functionality of high speed mode, the HSCLKEN bit in the **SSICR1** register should be set before any SSI data transfer or after applying a reset to the QSSI module. In addition, the SSE bit must be set to 0x1 before the HSCLKEN bit is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | DIR | RW | 0 | QSSI Direction of Operation<br>Value Description<br>0 TX (Transmit Mode) write direction<br>1 RX (Receive Mode) read direction |
| 7:6 | MODE | RW | 0x0 | QSSI Mode<br>Value Description<br>0x0 Legacy SSI mode<br>0x1 Bi-SSI mode<br>0x2 Quad-SSI Mode<br>0x3 Advanced SSI Mode with 8-bit packet size |
| 5:3 | reserved | RO | 0 | 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. |
| 2 | MS | RW | 0 | QSSI Master/Slave Select<br>This bit selects Master or Slave mode and can be modified only when the QSSI is disabled (SSE=0).<br>Value Description<br>0 The QSSI is configured as a master.<br>1 The QSSI is configured as a slave. |
| 1 | SSE | RW | 0 | QSSI Synchronous Serial Port Enable<br>Value Description<br>0 QSSI operation is disabled.<br>1 QSSI operation is enabled.<br>**Note:** The HSCLKEN bit in the **SSICR1** register should be set only after applying reset to the QSSI module and enabling the QSSI by setting the SSE bit, and before any SSI data transfer. All other bits in the **SSICR1** register and all bits in **SSICR0** register can only be programmed when the SSE is clear. |
| 0 | LBM | RW | 0 | QSSI Loopback Mode<br>Value Description<br>0 Normal serial port operation enabled.<br>1 Output of the transmit serial shift register is connected internally to the input of the receive serial shift register. |

## Register 3: QSSI Data (SSIDR), offset 0x008

**Important:** This register is read-sensitive. See the register description for details.

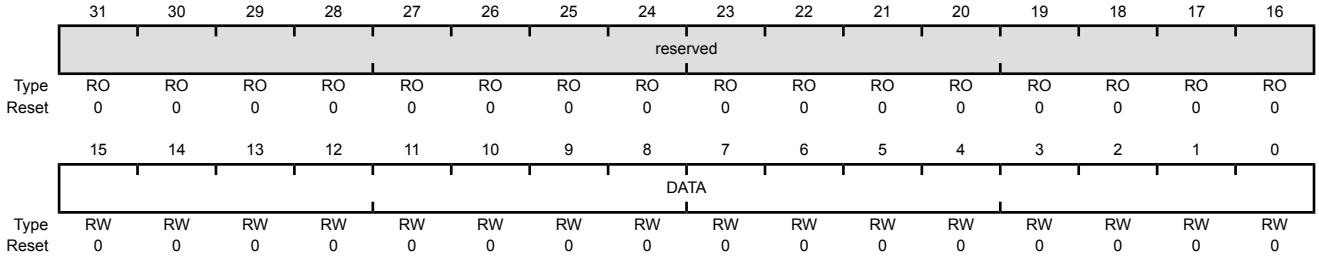
The **SSIDR** register is 16-bits wide. When the **SSIDR** register is read, the entry in the receive FIFO that is pointed to by the current FIFO read pointer is accessed. When a data value is removed by the QSSI receive logic from the incoming data frame, it is placed into the entry in the receive FIFO pointed to by the current FIFO write pointer.

When the **SSIDR** register is written to, the entry in the transmit FIFO that is pointed to by the write pointer is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. Each data value is loaded into the transmit serial shifter, then serially shifted out onto the SSInDAT0/SSInTX pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

QSSI Data (SSIDR)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x008
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | DATA | RW | 0x0000 | QSSI Receive/Transmit Data<br>A read operation reads the receive FIFO. A write operation writes the transmit FIFO.<br>Software must right-justify data when the QSSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data. |

## Register 4: QSSI Status (SSISR), offset 0x00C

The **SSISR** register contains bits that indicate the FIFO fill status and the QSSI busy status.

QSSI Status (SSISR)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x00C
Type RO, reset 0x0000.0003

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | BSY | RFF | RNE | TNF | TFE |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x0000.00 | 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. |
| 4 | BSY | RO | 0 | QSSI Busy Bit<br>Value Description<br>0 The QSSI is idle.<br>1 The QSSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty. |
| 3 | RFF | RO | 0 | QSSI Receive FIFO Full<br>Value Description<br>0 The receive FIFO is not full.<br>1 The receive FIFO is full. |
| 2 | RNE | RO | 0 | QSSI Receive FIFO Not Empty<br>Value Description<br>0 The receive FIFO is empty.<br>1 The receive FIFO is not empty. |
| 1 | TNF | RO | 1 | QSSI Transmit FIFO Not Full<br>Value Description<br>0 The transmit FIFO is full.<br>1 The transmit FIFO is not full. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TFE | RO | 1 | QSSI Transmit FIFO Empty |

| Value | Description |
|---|---|
| 0 | The transmit FIFO is not empty. |
| 1 | The transmit FIFO is empty. |

## Register 5: QSSI Clock Prescale (SSICPSR), offset 0x010

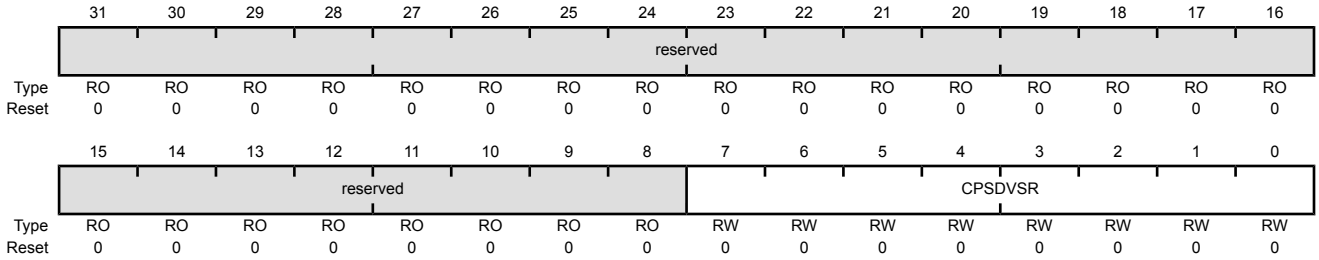
The **SSICPSR** register specifies the division factor which is used to derive the SSInClk from the system clock. The clock is further divided by a value from 1 to 256, which is 1 + SCR. SCR is programmed in the **SSICR0** register. The frequency of the SSInClk is defined by:

```
SSInClk = SysClk / (CPSDVSR * (1 + SCR))
```

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

QSSI Clock Prescale (SSICPSR)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x010
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x00 | 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. |
| 7:0 | CPSDVSR | RW | 0x00 | QSSI Clock Prescale Divisor<br>This value must be an even number from 2 to 254, depending on the frequency of SSInClk. The LSB always returns 0 on reads. |

## Register 6: QSSI Interrupt Mask (SSIIM), offset 0x014

The **SSIIM** register is the interrupt mask set or clear register. It is a read/write register and all bits are cleared on reset.

On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit clears the mask, enabling the interrupt to be sent to the interrupt controller. Clearing a bit sets the corresponding mask, preventing the interrupt from being signaled to the controller.

QSSI Interrupt Mask (SSIIM)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x014
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | EOTIM | DMATXIM | DMARXIM | TXIM | RXIM | RTIM | RORIM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0 | 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. |
| 6 | EOTIM | RW | 0 | End of Transmit Interrupt Mask<br>Value Description<br>0 The end of transmit interrupt is masked.<br>1 The end of transmit interrupt is not masked. |
| 5 | DMATXIM | RW | 0 | QSSI Transmit DMA Interrupt Mask<br>Value Description<br>0 The transmit DMA interrupt is masked.<br>1 The transmit DMA interrupt is not masked. |
| 4 | DMARXIM | RW | 0 | QSSI Receive DMA Interrupt Mask<br>Value Description<br>0 The receive DMA interrupt is masked.<br>1 The receive DMA interrupt is not masked. |
| 3 | TXIM | RW | 0 | QSSI Transmit FIFO Interrupt Mask<br>Value Description<br>0 The transmit FIFO interrupt is masked.<br>1 The transmit FIFO interrupt is not masked. |

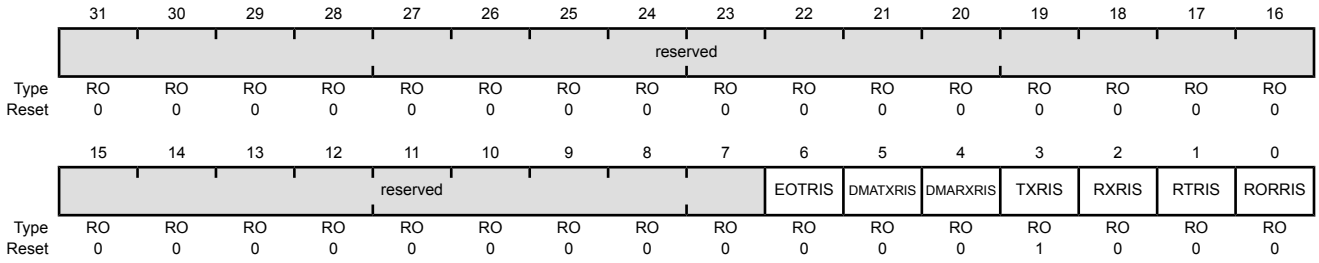
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | RXIM | RW | 0 | QSSI Receive FIFO Interrupt Mask<br>Value Description<br>0 The receive FIFO interrupt is masked.<br>1 The receive FIFO interrupt is not masked. |
| 1 | RTIM | RW | 0 | QSSI Receive Time-Out Interrupt Mask<br>Value Description<br>0 The receive FIFO time-out interrupt is masked.<br>1 The receive FIFO time-out interrupt is not masked. |
| 0 | RORIM | RW | 0 | QSSI Receive Overrun Interrupt Mask<br>Value Description<br>0 The receive FIFO overrun interrupt is masked.<br>1 The receive FIFO overrun interrupt is not masked. |

## Register 7: QSSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

QSSI Raw Interrupt Status (SSIRIS)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x018
Type RO, reset 0x0000.0008




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0 | 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. |
| 6 | EOTRIS | RO | 0 | End of Transmit Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The transmit FIFO is empty, and the last bit has been transmitted out of the serializer.<br><br>This bit is cleared when a 1 is written to the EOTIC bit in the **SSI Interrupt Clear (SSIICR)** register. |
| 5 | DMATXRIS | RO | 0 | QSSI Transmit DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The transmit DMA has completed.<br><br>This bit is cleared when a 1 is written to the DMATXIC bit in the **SSI Interrupt Clear (SSIICR)** register. |
| 4 | DMARXRIS | RO | 0 | QSSI Receive DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The receive DMA has completed.<br><br>This bit is cleared when a 1 is written to the DMARXIC bit in the **SSI Interrupt Clear (SSIICR)** register. |

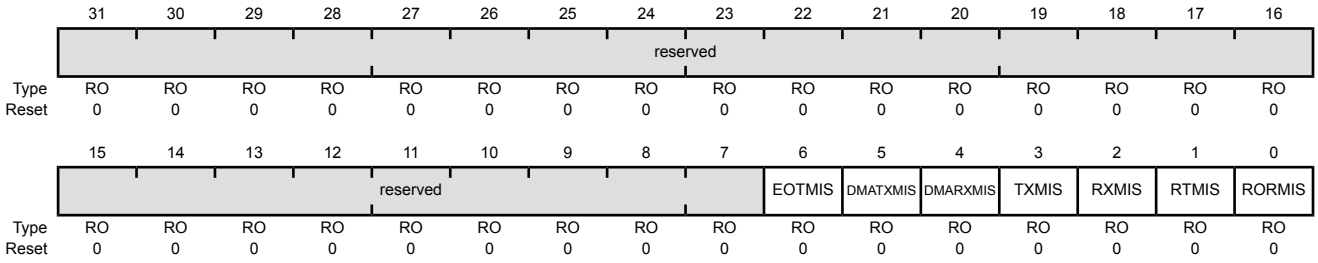
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | TXRIS | RO | 1 | QSSI Transmit FIFO Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The transmit FIFO is half empty or less.<br><br>This bit is cleared when the transmit FIFO is more than half full. |
| 2 | RXRIS | RO | 0 | QSSI Receive FIFO Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The receive FIFO is half full or more.<br><br>This bit is cleared when the receive FIFO is less than half full. |
| 1 | RTRIS | RO | 0 | QSSI Receive Time-Out Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The receive time-out has occurred.<br><br>This bit is cleared when a 1 is written to the RTIC bit in the **SSI Interrupt Clear (SSIICR)** register. |
| 0 | RORRIS | RO | 0 | QSSI Receive Overrun Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The receive FIFO has overflowed<br><br>This bit is cleared when a 1 is written to the RORIC bit in the **SSI Interrupt Clear (SSIICR)** register. |

## Register 8: QSSI Masked Interrupt Status (SSIMIS), offset 0x01C

The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

QSSI Masked Interrupt Status (SSIMIS)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x01C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0 | 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. |
| 6 | EOTMIS | RO | 0 | End of Transmit Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the transmission of the last data bit.<br><br>This bit is cleared when a 1 is written to the EOTIC bit in the **SSI Interrupt Clear (SSIICR)** register. |
| 5 | DMATXMIS | RO | 0 | QSSI Transmit DMA Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the completion of the transmit DMA.<br><br>This bit is cleared when a 1 is written to the DMATXIC bit in the **SSI Interrupt Clear (SSIICR)** register. |
| 4 | DMARXMIS | RO | 0 | QSSI Receive DMA Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the completion of the receive DMA.<br><br>This bit is cleared when a 1 is written to the DMARXIC bit in the **SSI Interrupt Clear (SSIICR)** register. |

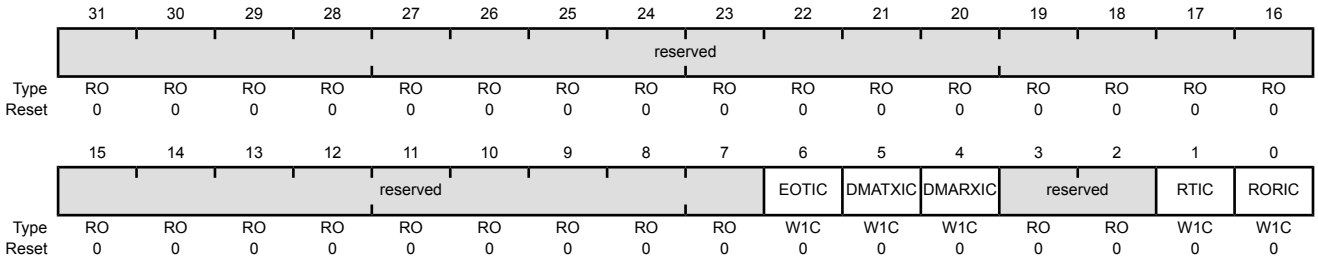
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | TXMIS | RO | 0 | QSSI Transmit FIFO Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the transmit FIFO being half empty or less.<br><br>This bit is cleared when the transmit FIFO is more than half empty . |
| 2 | RXMIS | RO | 0 | QSSI Receive FIFO Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the receive FIFO being half full or more.<br><br>This bit is cleared when the receive FIFO is less than half full. |
| 1 | RTMIS | RO | 0 | QSSI Receive Time-Out Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the receive time out.<br><br>This bit is cleared when a 1 is written to the RTIC bit in the **SSI Interrupt Clear (SSIICR)** register. |
| 0 | RORMIS | RO | 0 | QSSI Receive Overrun Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked interrupt was signaled due to the receive FIFO overflowing.<br><br>This bit is cleared when a 1 is written to the RORIC bit in the **SSI Interrupt Clear (SSIICR)** register. |

## Register 9: QSSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

QSSI Interrupt Clear (SSIICR)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x020
Type W1C, reset 0x0000.0000



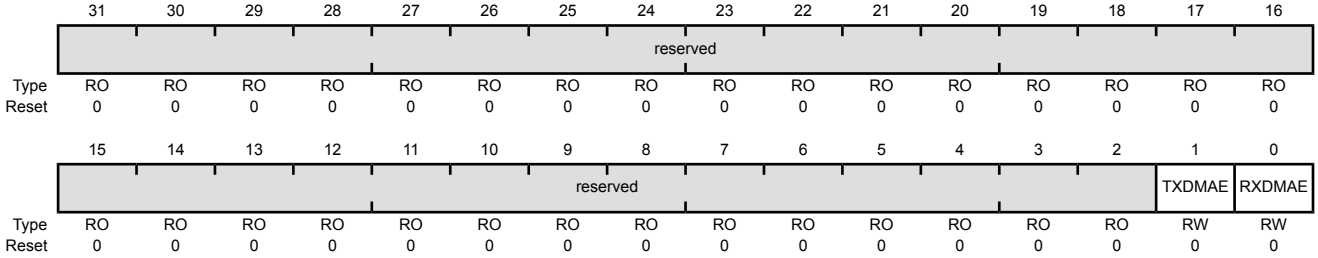

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6 | EOTIC | W1C | 0 | End of Transmit Interrupt Clear<br>Writing a 1 to this bit clears the EOTRIS bit in the **SSIRIS** register and the EOTMIS bit in the **SSIMIS** register. |
| 5 | DMATXIC | W1C | 0 | QSSI Transmit DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMATXRIS bit in the **SSIRIS** register and the DMATXMIS bit in the **SSIMIS** register. |
| 4 | DMARXIC | W1C | 0 | QSSI Receive DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMARXRIS bit in the **SSIRIS** register and the DMARXMIS bit in the **SSIMIS** register. |
| 3:2 | reserved | RO | 0x0 | 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. |
| 1 | RTIC | W1C | 0 | QSSI Receive Time-Out Interrupt Clear<br>Writing a 1 to this bit clears the RTRIS bit in the **SSIRIS** register and the RTMIS bit in the **SSIMIS** register. |
| 0 | RORIC | W1C | 0 | QSSI Receive Overrun Interrupt Clear<br>Writing a 1 to this bit clears the RORRIS bit in the **SSIRIS** register and the RORMIS bit in the **SSIMIS** register. |

## Register 10: QSSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the µDMA control register.

QSSI DMA Control (SSIDMACTL)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0x024
Type RW, reset 0x0000.0000



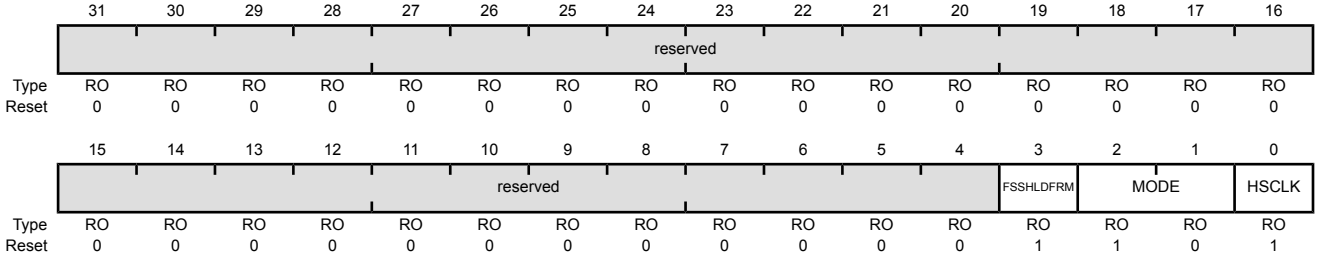

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | TXDMAE | RW | 0 | Transmit DMA Enable<br><br>Value Description<br>0 µDMA for the transmit FIFO is disabled.<br>1 µDMA for the transmit FIFO is enabled. |
| 0 | RXDMAE | RW | 0 | Receive DMA Enable<br><br>Value Description<br>0 µDMA for the receive FIFO is disabled.<br>1 µDMA for the receive FIFO is enabled. |

## Register 11: QSSI Peripheral Properties (SSIPP), offset 0xFC0

The **SSIPP** register provides information regarding the properties of the QSSI module.

QSSI Peripheral Properties (SSIPP)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFC0
Type RO, reset 0x0000.000D




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.0000 | 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. |
| 3 | FSSHLDFRM | RO | 0x1 | `SSInFss` Hold Frame Capability<br>Value Description<br>0 `SSInFss` Hold Frame capability disabled.<br>1 `SSinFss` Hold Frame capability enabled. |
| 2:1 | MODE | RO | 0x2 | Mode of Operation<br>Indicates what QSSI functionality is supported.<br>Value Description<br>0x0 Legacy SSI mode<br>0x1 Legacy mode, Advanced SSI mode and Bi-SSI mode enabled.<br>0x2 Legacy mode, Advanced mode, Bi-SSI and Quad-SSI mode enabled.<br>0x3 reserved |
| 0 | HSCLK | RO | 0x1 | High Speed Capability<br>Value Description<br>0 High Speed clock capability disabled.<br>1 High speed clock capability enabled. |

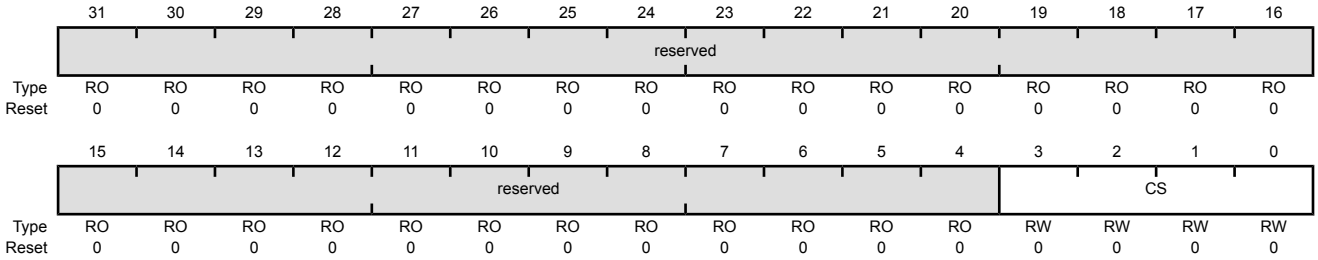
## Register 12: QSSI Clock Configuration (SSICC), offset 0xFC8

The **SSICC** register controls the baud clock source for the QSSI module.

**Note:** If ALTCLK is used for the QSSI baud clock, the system clock frequency must be at least twice that of the ALTCLK programmed value in Run mode.

QSSI Clock Configuration (SSICC)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFC8
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | CS | RW | 0 | QSSI Baud Clock Source<br>The following table specifies the source that generates for the QSSI baud clock: |

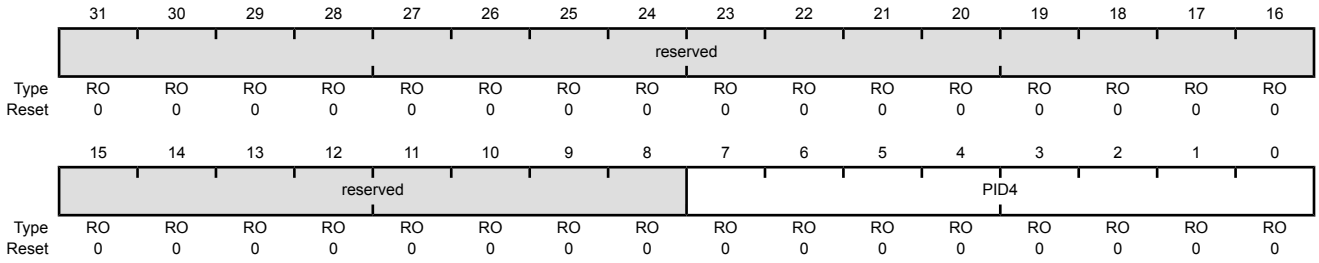
| Value | Description |
|---|---|
| 0x0 | System clock (based on clock source and divisor factor programmed in **RSCLKCFG** register in the System Control Module) |
| 0x1-0x4 | reserved |
| 0x5 | Alternate clock source as defined by **ALTCLKCFG** register in System Control Module. |
| 0x6 - 0xF | Reserved |

## Register 13: QSSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 4 (SSIPeriphID4)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFD0
Type RO, reset 0x0000.0000



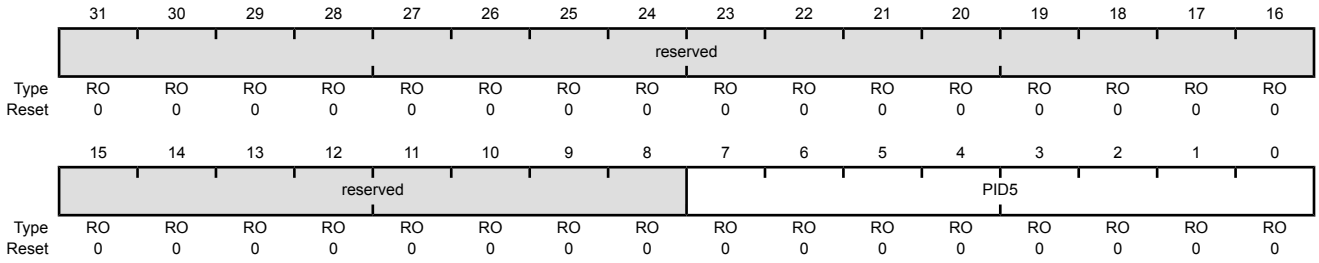

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | QSSI Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 14: QSSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 5 (SSIPeriphID5)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFD4
Type RO, reset 0x0000.0000



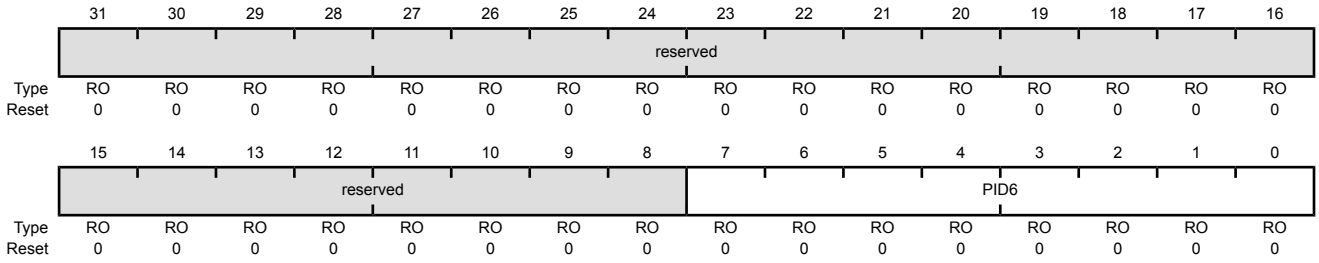

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | QSSI Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 15: QSSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 6 (SSIPeriphID6)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFD8
Type RO, reset 0x0000.0000



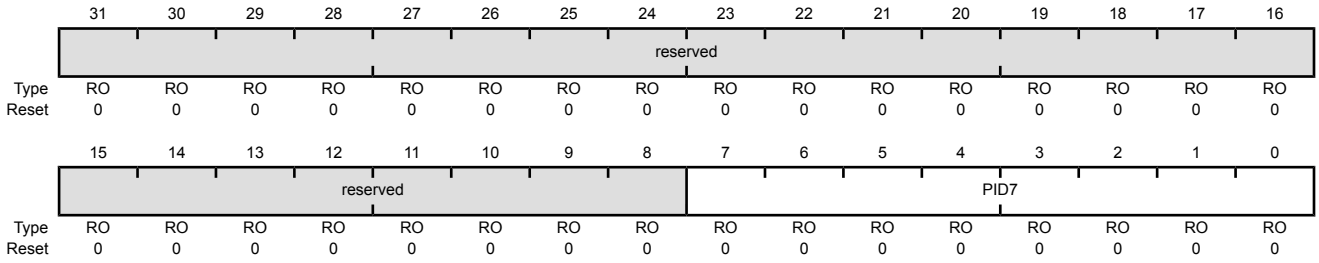

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | QSSI Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 16: QSSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 7 (SSIPeriphID7)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFDC
Type RO, reset 0x0000.0000



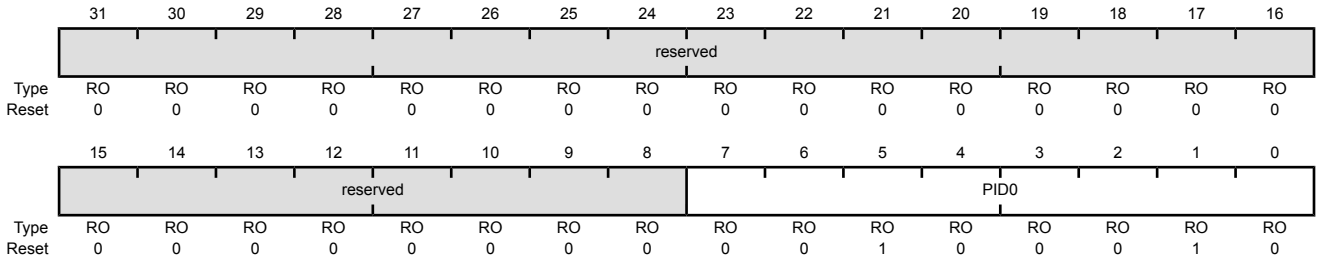

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | QSSI Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 17: QSSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 0 (SSIPeriphID0)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFE0
Type RO, reset 0x0000.0022



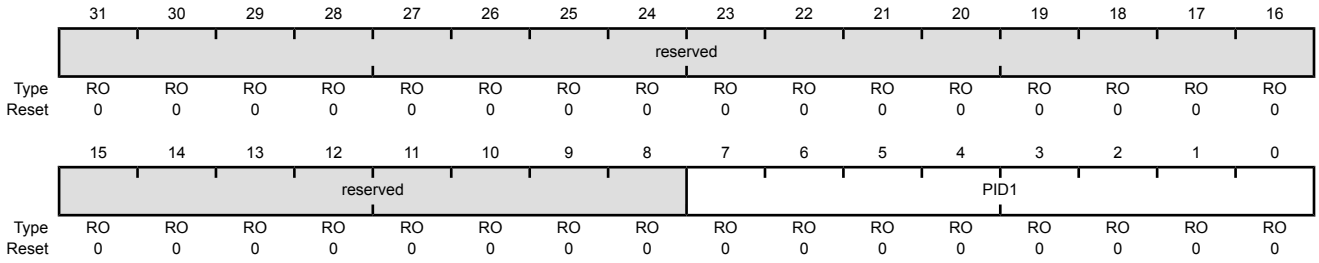

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x22 | QSSI Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 18: QSSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 1 (SSIPeriphID1)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFE4
Type RO, reset 0x0000.0000



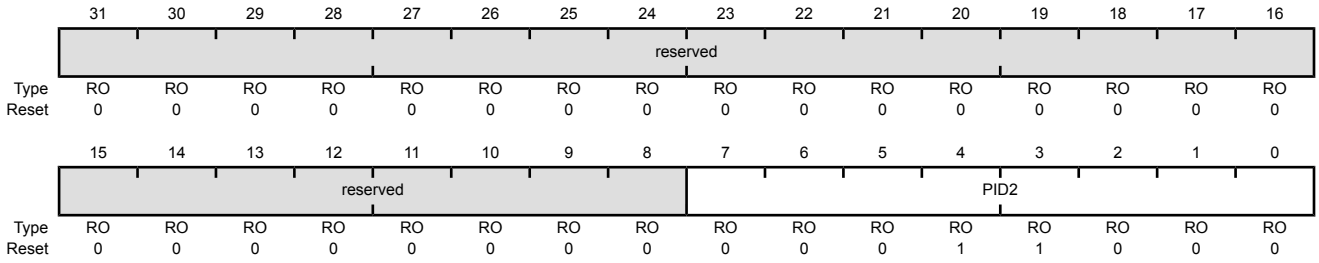

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x00 | QSSI Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 19: QSSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 2 (SSIPeriphID2)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFE8
Type RO, reset 0x0000.0018



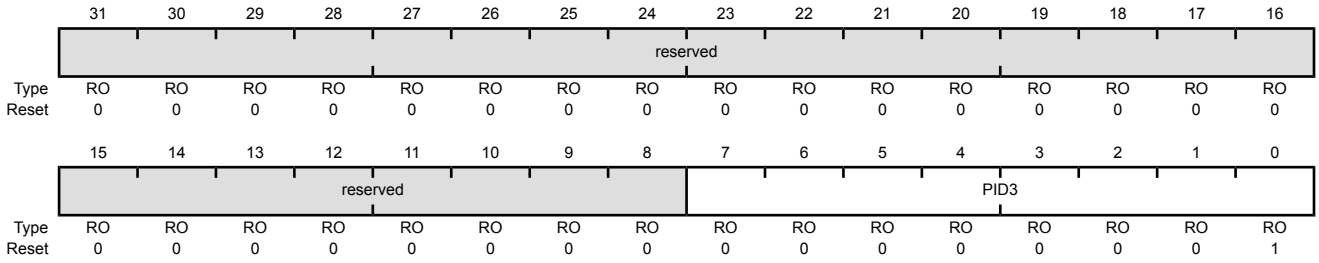

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | QSSI Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 20: QSSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

QSSI Peripheral Identification 3 (SSIPeriphID3)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFEC
Type RO, reset 0x0000.0001



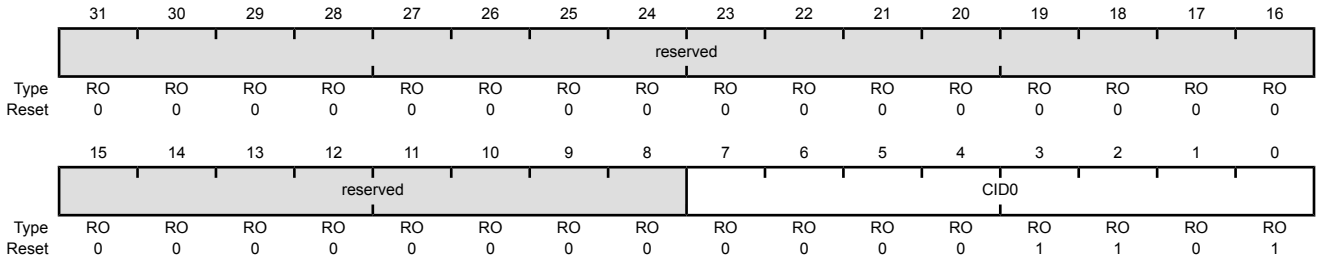

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | QSSI Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 21: QSSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

QSSI PrimeCell Identification 0 (SSIPCellID0)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFF0
Type RO, reset 0x0000.000D



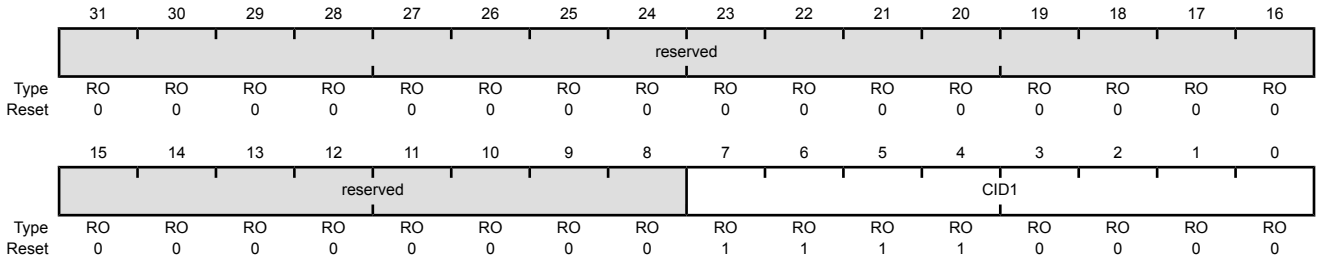

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | QSSI PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 22: QSSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

QSSI PrimeCell Identification 1 (SSIPCellID1)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFF4
Type RO, reset 0x0000.00F0



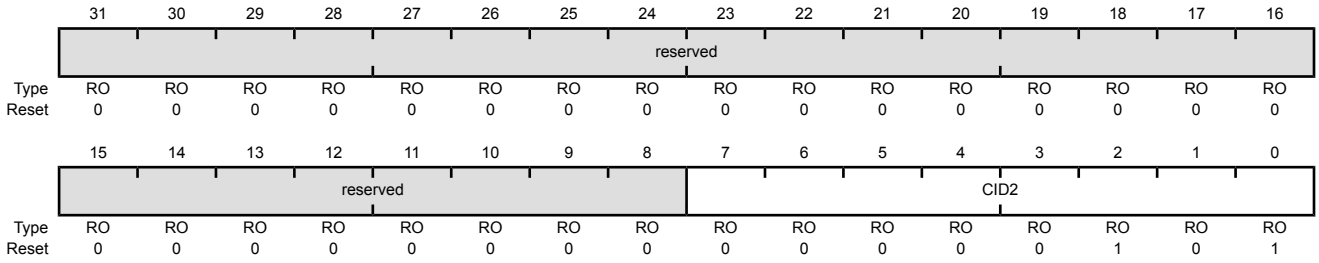

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | QSSI PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 23: QSSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

QSSI PrimeCell Identification 2 (SSIPCellID2)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFF8
Type RO, reset 0x0000.0005



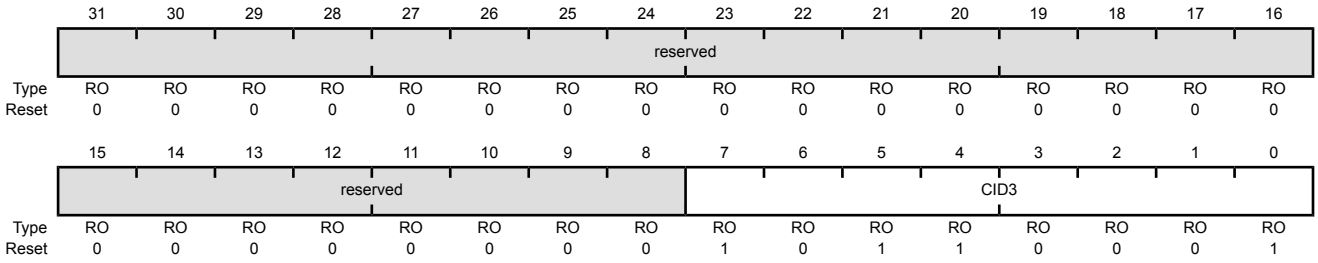

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | QSSI PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 24: QSSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

QSSI PrimeCell Identification 3 (SSIPCellID3)

QSSI0 base: 0x4000.8000
QSSI1 base: 0x4000.9000
QSSI2 base: 0x4000.A000
QSSI3 base: 0x4000.B000
Offset 0xFFC
Type RO, reset 0x0000.00B1




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | CID3 | RO | 0xB1 | QSSI PrimeCell ID Register [31:24]<br>Provides software a standard cross-peripheral identification system. |

# 21 Inter-Integrated Circuit ($I^2C$) Interface

The Inter-Integrated Circuit ($I^2C$) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external $I^2C$ devices such as serial memory (RAMs and ROMs), 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. The TM4C129XKCZAD microcontroller includes providing the ability to communicate (both transmit and receive) with other $I^2C$ devices on the bus.

The TM4C129XKCZAD controller includes $I^2C$ modules with the following features:

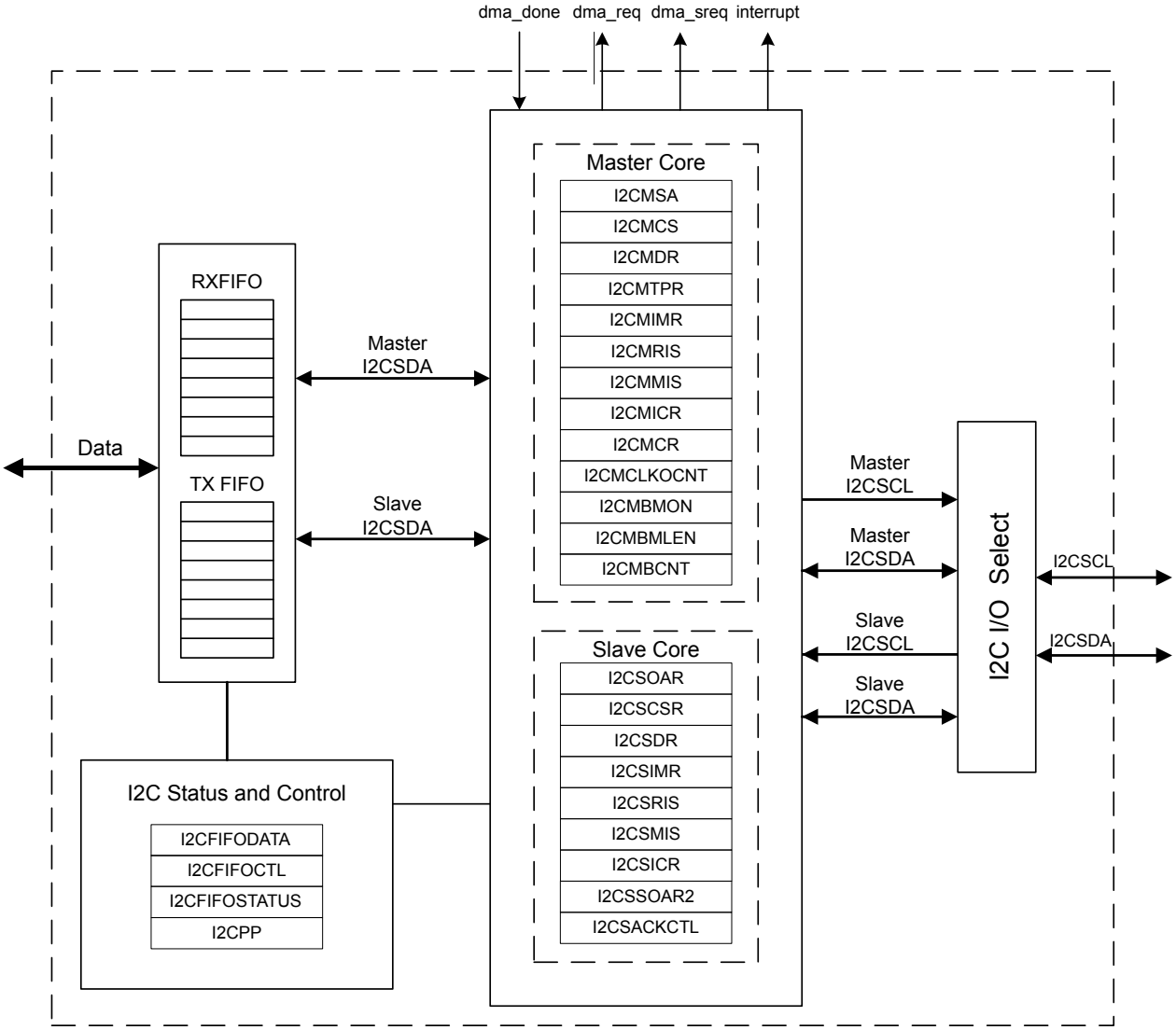
- Devices on the $I^2C$ bus can be designated as either a master or a slave
  - Supports both transmitting and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation
- Four $I^2C$ modes
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive
- Two 8-entry FIFOs for receive and transmit data
  - FIFOs can be independently assigned to master or slave
- Four transmission speeds:
  - Standard (100 Kbps)
  - Fast-mode (400 Kbps)
  - Fast-mode plus (1 Mbps)
  - High-speed mode (3.33 Mbps)
- Glitch suppression
- SMBus support through software
  - Clock low timeout interrupt
  - Dual slave address capability
  - Quick command capability
- 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
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Ability to execute single data transfers or burst data transfers using the RX and TX FIFOs in the $I^2C$

## 21.1 Block Diagram

**Figure 21-1. $I^2C$ Block Diagram**



## 21.2 Signal Description

The following table lists the external signals of the I²C interface and describes the function of each. The I²C interface signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²C signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the I²C function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the I²C signal to the specified GPIO port pin. Note that the I2CSDA pin should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 21-1. I2C Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| I2C0SCL | A17 | PB2 (2) | I/O | OD | I²C module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C0SDA | B17 | PB3 (2) | I/O | OD | I²C module 0 data. |
| I2C1SCL | N15<br>N5 | PG0 (2)<br>PR0 (2) | I/O | OD | I²C module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C1SDA | T14<br>N4 | PG1 (2)<br>PR1 (2) | I/O | OD | I²C module 1 data. |
| I2C2SCL | V11<br>H19<br>B9<br>B12<br>N2 | PG2 (2)<br>PL1 (2)<br>PN5 (3)<br>PP5 (2)<br>PR2 (2) | I/O | OD | I²C module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C2SDA | M16<br>G16<br>A10<br>B8<br>V8 | PG3 (2)<br>PL0 (2)<br>PN4 (3)<br>PP6 (2)<br>PR3 (2) | I/O | OD | I²C module 2 data. |
| I2C3SCL | K17<br>U19<br>P3 | PG4 (2)<br>PK4 (2)<br>PR4 (2) | I/O | OD | I²C module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C3SDA | K15<br>V17<br>P2 | PG5 (2)<br>PK5 (2)<br>PR5 (2) | I/O | OD | I²C module 3 data. |
| I2C4SCL | V12<br>V16<br>W9 | PG6 (2)<br>PK6 (2)<br>PR6 (2) | I/O | OD | I²C module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C4SDA | U14<br>W16<br>R10 | PG7 (2)<br>PK7 (2)<br>PR7 (2) | I/O | OD | I²C module 4 data. |
| I2C5SCL | A16<br>C6 | PB0 (2)<br>PB4 (2) | I/O | OD | I²C module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C5SDA | B16<br>B6 | PB1 (2)<br>PB5 (2) | I/O | OD | I²C module 5 data. |

**Table 21-1. I2C Signals (212BGA) *(continued)***

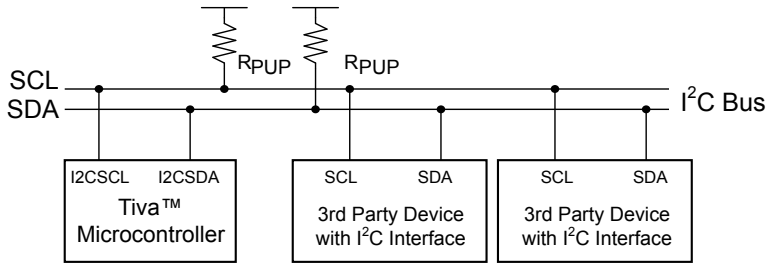
| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| I2C6SCL | V5<br>F2 | PA6 (2)<br>PB6 (2) | I/O | OD | I2C module 6 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C6SDA | R7<br>F1 | PA7 (2)<br>PB7 (2) | I/O | OD | I2C module 6 data. |
| I2C7SCL | V4<br>C2 | PA4 (2)<br>PD0 (2) | I/O | OD | I2C module 7 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C7SDA | W4<br>C1 | PA5 (2)<br>PD1 (2) | I/O | OD | I2C module 7 data. |
| I2C8SCL | T6<br>D2 | PA2 (2)<br>PD2 (2) | I/O | OD | I2C module 8 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C8SDA | U5<br>D1 | PA3 (2)<br>PD3 (2) | I/O | OD | I2C module 8 data. |
| I2C9SCL | V3<br>A7 | PA0 (2)<br>PE6 (2) | I/O | OD | I2C module 9 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C9SDA | W3<br>B7 | PA1 (2)<br>PE7 (2) | I/O | OD | I2C module 9 data. |

## 21.3 Functional Description

Each I2C module is comprised of both master and slave functions and is identified by a unique address. A master-initiated communication generates the clock signal, SCL. For proper operation, the SDA pin must be configured as an open-drain signal. Due to the internal circuitry that supports high-speed operation, the SCL pin must not be configured as an open-drain signal, although the internal circuitry causes it to act as if it were an open drain signal. Both SDA and SCL signals must be connected to a positive supply voltage using a pull-up resistor. A typical I2C bus configuration is shown in Figure 21-2. Refer to the *I2C-bus specification and user manual* to determine the size of the pull-ups needed for proper operation.

See “Inter-Integrated Circuit (I2C) Interface” on page 2147 for I2C timing diagrams.

**Figure 21-2. I2C Bus Configuration**




### 21.3.1 I2C Bus Functional Overview

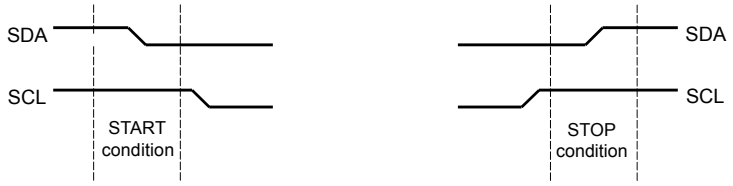
The I2C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on TM4C129XKCZAD microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

Every transaction on the I²C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in "START and STOP Conditions" on page 1431) is unrestricted, but each data byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it 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.

#### 21.3.1.1 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 SCL 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 21-3.

**Figure 21-3. START and STOP Conditions**



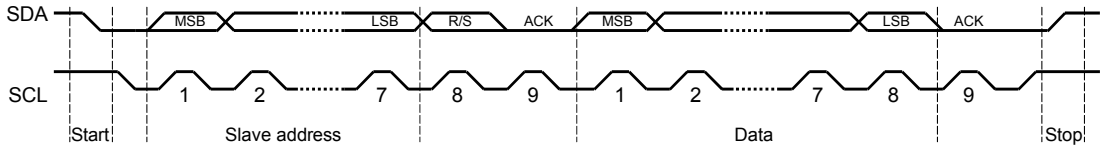

The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I²C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and the Control register is written 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 may be read from the **I²C Master Data (I2CMDR)** 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. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

When operating in slave mode, the STARTRIS and STOPRIS bits in the **I²C Slave Raw Interrupt Status (I2CSRIS)** register indicate detection of start and stop conditions on the bus and the **I²C Slave Masked Interrupt Status (I2CSMIS)** register can be configured to allow STARTRIS and STOPRIS to be promoted to controller interrupts (when interrupts are enabled).

#### 21.3.1.2 Data Format with 7-Bit Address

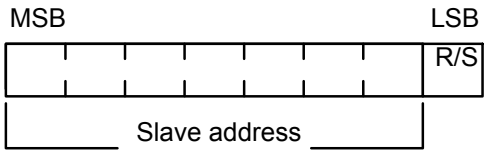
Data transfers follow the format shown in Figure 21-4. After the START condition, a slave address is transmitted. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the **I2CMSA** register). If the R/S bit is clear, it indicates a transmit operation (send), 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/transmit formats are then possible within a single transfer.

**Figure 21-4. Complete Data Transfer with a 7-Bit Address**




The first seven bits of the first byte make up the slave address (see Figure 21-5). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave.

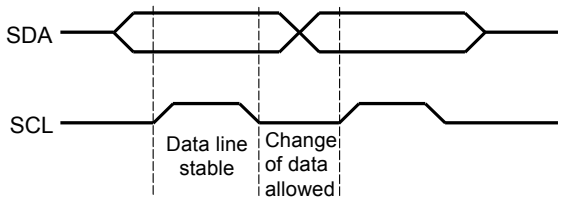
**Figure 21-5. R/S Bit in First Byte**




### 21.3.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is Low (see Figure 21-6).

**Figure 21-6. Data Validity During Bit Transfer on the I2C Bus**




### 21.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the 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 out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 1432.

When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave 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 allow the master to generate the STOP or a repeated START condition.

If the slave is required to provide a manual ACK or NACK, the **I2C Slave ACK Control (I2CSACKCTL)** register allows the slave to NACK for invalid data or command or ACK for valid data or command. When this operation is enabled, the MCU slave module I2C clock is pulled low after the last data bit until this register is written with the indicated response.

#### 21.3.1.5 Repeated Start

The I²C master module has the capability of executing a repeated START (transmit or receive) after an initial transfer has occurred.

A repeated start sequence for a Master transmit is as follows:

1. When the device is in the idle state, the Master writes the slave address to the **I2CMSA** register and configures the R/S bit for the desired transfer type.
2. Data is written to the **I2CMDR** register.
3. When the BUSY bit in the **I2CMCS** register is 0 , the Master writes 0x3 to the **I2CMCS** register to initiate a transfer.
4. The Master does not generate a STOP condition but instead writes another slave address to the **I2CMSA** register and then writes 0x3 to initiate the repeated START.

A repeated start sequence for a Master receive is similar:

1. When the device is in idle, the Master writes the slave address to the **I2CMSA** register and configures the R/S bit for the desired transfer type.
2. The master reads data from the **I2CMDR** register.
3. When the BUSY bit in the **I2CMCS** register is 0 , the Master writes 0x3 to the **I2CMCS** register to initiate a transfer.
4. The Master does not generate a STOP condition but instead writes another slave address to the **I2CMSA** register and then writes 0x3 to initiate the repeated START.

For more information on repeated START, refer to Figure 21-12 on page 1446 and Figure 21-13 on page 1447.

#### 21.3.1.6 Clock Low Timeout (CLTO)

The I²C slave can extend the transaction by pulling the clock low periodically to create a slow bit transfer rate. The I²C module has a 12-bit programmable counter that is used to track how long the clock has been held low. The upper 8 bits of the count value are software programmable through the **I²C Master Clock Low Timeout Count (I2CMCLKOCNT)** register. The lower four bits are not user visible and are 0x0. The CNTL value programmed in the **I2CMCLKOCNT** register has to be greater than 0x01. The application can program the eight most significant bits of the counter to reflect the acceptable cumulative low period in transaction. The count is loaded at the START condition and counts down on each falling edge of the internal bus clock of the Master. Note that the internal bus clock generated for this counter keeps running at the programmed I²C speed even if SCL is held low on the bus. Upon reaching terminal count, the master state machine forces ABORT on the bus by issuing a STOP condition at the instance of SCL and SDA release.

As an example, if an I²C module was operating at 100 kHz speed, programming the **I2CMCLKOCNT** register to 0xDA would translate to the value 0xDA0 since the lower four bits are set to 0x0. This would translate to a decimal value of 3488 clocks or a cumulative clock low period of 34.88 ms at 100 kHz.

The CLKRIS bit in the **I²C Master Raw Interrupt Status (I2CMRIS)** register is set when the clock timeout period is reached, allowing the master to start corrective action to resolve the remote slave state. In addition, the CLKTO bit in the **I²C Master Control/Status (I2CMCS)** register is set; this bit

is cleared when a STOP condition is sent or during the I2C master reset. The status of the raw SDA and SCL signals are readable by software through the SDA and SCL bits in the **I2C Master Bus Monitor (I2CMBMON)** register to help determine the state of the remote slave.

In the event of a CLTO condition, application software must choose how it intends to attempt bus recovery. Most applications may attempt to manually toggle the I2C pins to force the slave to let go of the clock signal (a common solution is to attempt to force a STOP on the bus). If a CLTO is detected before the end of a burst transfer, and the bus is successfully recovered by the master, the master hardware attempts to finish the pending burst operation. Depending on the state of the slave after bus recovery, the actual behavior on the bus varies. If the slave resumes in a state where it can acknowledge the master (essentially, where it was before the bus hang), it continues where it left off. However, if the slave resumes in a reset state (or if a forced STOP by the master causes the slave to enter the idle state), it may ignore the master's attempt to complete the burst operation and NAK the first data byte that the master sends or requests.

Since the behavior of slaves cannot always be predicted, it is suggested that the application software always write the STOP bit in the **I2C Master Configuration (I2CMCR)** register during the CLTO interrupt service routine. This limits the amount of data the master attempts to send or receive upon bus recovery to a single byte, and after the single byte is on the wire, the master issues a STOP. An alternative solution is to have the application software reset the I2C peripheral before attempting to manually recover the bus. This solution allows the I2C master hardware to be returned to a known good (and idle) state before attempting to recover a stuck bus and prevents any unwanted data from appearing on the wire.

**Note:** The Master Clock Low Timeout counter counts for the entire time SCL is held Low continuously. If SCL is deasserted at any point, the Master Clock Low Timeout Counter is reloaded with the value in the **I2CMCLKOCNT** register and begins counting down from this value.

#### 21.3.1.7 Dual Address

The I2C interface supports dual address capability for the slave. The additional programmable address is provided and can be matched if enabled. In legacy mode with dual address disabled, the I2C slave provides an ACK on the bus if the address matches the OAR field in the **I2CSOAR** register. In dual address mode, the I2C slave provides an ACK on the bus if either the OAR field in the **I2CSOAR** register or the OAR2 field in the **I2CSOAR2** register is matched. The enable for dual address is programmable through the OAR2EN bit in the **I2CSOAR2** register and there is no disable on the legacy address.

The OAR2SEL bit in the **I2CSCSR** register indicates if the address that was ACKed is the alternate address or not. When this bit is clear, it indicates either legacy operation or no address match.

#### 21.3.1.8 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a 1 (High) on SDA, while another master transmits a 0 (Low), switches off its data output stage and retires until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

If arbitration is lost when the I2C master is initiating a BURST with the TX FIFO enabled, the application should execute the following steps to correctly handle the arbitration loss:

1. Flush and disable the TX FIFO

2. Clear and mask the TXFE interrupt by clearing the TXFEIM bit in the **I2CMIMR** register.

Once the bus is IDLE, the TXFIFO can be filled and enabled, the TXFE bit can be unmasked and a new BURST transaction can be initiated.

#### 21.3.1.9 Glitch Suppression in Multi-Master Configuration

When a multi-master configuration is being used, the PULSEL bit in the **I2CMTPR** register can be programmed to provide glitch suppression on the SCL and SDA lines and assure proper signal values. The glitch suppression value is in terms of buffered system clocks. Note that all signals will be delayed internally when glitch suppression is nonzero. For example, if PULSEL is set to 0x7, 31 clocks should be added onto the calculation for the expected transaction time.

#### 21.3.1.10 SMBus Operation

The SMBus interface is based on the $I^2C$ protocol; however, some differences exist between the two. These differences must be handled through software in order to make sure the SMBus protocol, including timing specifications, is met. Note that the SMBus 2.0 specification limits the maximum frequency of the interface to 100 KHz, as a result, $I^2C$ Standard speed operation is used for SMBus.

The SMBus/ $I^2C$ slave can extend the transaction if it is not ready by pulling the clock low. The SMbus specification allows the maximum timeout for such elongated transaction to be between 25 to 35 ms. The $I^2C$ specification does not have this requirement. The $I^2C$ module supports a programmable count to support clock-low timeout for the master to error out and take appropriate action as required. This feature is explained in “Clock Low Timeout (CLTO)” on page 1433. Note that if transactions are extended, a timeout period should be programmed in the **I2CMCLKOCNT** register, and the CLKRIS bit in the **I2CMRIS** register should not be masked.

Unlike the $I^2C$ slave, the SMBus slave must respond with an ACK response to its address regardless of whether it is ready or not. As a result, the $I^2C$ slave sends an ACK response to its address and a NACK response on the data byte if it is not ready. The ARBLST bit in the **I2CMCS** register is set if there were any issues with the transfer. In addition, the slave can send a NACK at any time to force the master to stop sending additional bytes.

The $I^2C$ Interface supports µDMA for efficient data handling. The µDMA operation needs FIFOs to be enabled for appropriate transfer type to perform $I^2C$ Master for burst transfers and all types of Slave transfers. The $I^2C$ interface is supported by two channels, one for Rx ($I^2C$ to Memory) and one for Tx (Memory to $I^2C$) transfers.. See“FIFO and µDMA Operation” on page 1439 for more information.

##### *Quick Command*

Quick Command is a simple, compact SMBus protocol that sends an address and 1-bit of data in the R/S bit of the $I^2C$ header byte to communicate a command to the slave, typically a "turn off" or "turn on". The $I^2C$ master peripheral has the ability to send a Quick Command by writing the target address and R/S value into the **I2CMSA** register followed by a write to **I2CMCS** with a value of 0x27. SMBus requires the slave to be able to accept and process commands and the master to generate the Quick Command transactions. The master also has the capability to stop the transaction after acknowledgement from a slave.

The $I^2C$ slave peripheral requires special handling when a Quick Command is sent. In the case where a master sends a Quick Command with the R/S (data) bit cleared, the QCMDST bit in **I2CSCSR** is set, and the QCMDRW bit shows the data value (which in this case is 0) when the STOPRIS bit is set in **I2CSRIS** and the STOP interrupt is asserted. In this scenario, a DATARIS interrupt bit is not

set. When the master sends a Quick Command with the R/S (data) bit set, the DATARIS bit is set to notify the slave to write a data byte to **I2CSDR** in which bit 7 is set. A “dummy write” of 0xFF to the **I2CSDR** register is recommended. After the write to **I2CSDR**, the STOP interrupt is asserted and the QCMDST and QCMDRW bits are set in the **I2CSCSR** register to indicate that a quick command read occurred and the last transaction was a Quick Command. Therefore, when the slave must receive a Quick Command, it should be expecting such a command because it must write the **I2CSDR** with a specific value when R/S is set.

## 21.3.2 Available Speed Modes

The I2C bus can run in Standard mode (100 kbps), Fast mode (400 kbps), Fast mode plus (1 Mbps) or High-Speed mode (3.4 Mbps, provided correct system clock frequency is set and there is appropriate pull strength on SCL and SDA lines). The selected mode should match the speed of the other I2C devices on the bus.

### 21.3.2.1 Standard, Fast, and Fast Plus Modes

Standard, Fast, and Fast Plus modes are selected using a value in the **I2C Master Timer Period (I2CMTPR)** register that results in an SCL frequency of 100 kbps for Standard mode, 400 kbps for Fast mode, or 1 Mbps for Fast mode plus.

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

*SCL_LP* is the low phase of SCL (fixed at 6)

*SCL_HP* is the high phase of SCL (fixed at 4)

*TIMER_PRD* is the programmed value in the **I2CMTPR** register (see page 1465). This value is determined by replacing the known variables in the equation below and solving for *TIMER_PRD*.

The I2C clock period is calculated as follows:

$SCL_PERIOD = 2 \times (1 + TIMER_PRD) \times (SCL_LP + SCL_HP) \times CLK_PRD$

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 21-2 gives examples of the timer periods that should be used to generate Standard, Fast mode, and Fast mode plus SCL frequencies based on various system clock frequencies.

**Table 21-2. Examples of I2C Master Timer Period Versus Speed Mode**

| System Clock | Timer Period | Standard Mode | Timer Period | Fast Mode | Timer Period | Fast Mode Plus |
|---|---|---|---|---|---|---|
| 4 MHz | 0x01 | 100 Kbps | - | - | - | - |
| 6 MHz | 0x02 | 100 Kbps | - | - | - | - |

**Table 21-2. Examples of I²C Master Timer Period Versus Speed Mode *(continued)***

| System Clock | Timer Period | Standard Mode | Timer Period | Fast Mode | Timer Period | Fast Mode Plus |
|---|---|---|---|---|---|---|
| 12.5 MHz | 0x06 | 89 Kbps | 0x01 | 312 Kbps | - | - |
| 16.7 MHz | 0x08 | 93 Kbps | 0x02 | 278 Kbps | - | - |
| 20 MHz | 0x09 | 100 Kbps | 0x02 | 333 Kbps | - | - |
| 25 MHz | 0x0C | 96.2 Kbps | 0x03 | 312 Kbps | - | - |
| 33 MHz | 0x10 | 97.1 Kbps | 0x04 | 330 Kbps | - | - |
| 40 MHz | 0x13 | 100 Kbps | 0x04 | 400 Kbps | 0x01 | 1000 Kbps |
| 50 MHz | 0x18 | 100 Kbps | 0x06 | 357 Kbps | 0x02 | 833 Kbps |
| 80 MHz | 0x27 | 100 Kbps | 0x09 | 400 Kbps | 0x03 | 1000 Kbps |
| 100 MHz | 0x31 | 100 Kbps | 0x0C | 385 Kbps | 0x04 | 1000 Kbps |
| 120 MHz | 0x3B | 100 Kbps | 0xE | 400 Kbps | 0x5 | 1000 Kbps |

#### 21.3.2.2 High-Speed Mode

The TM4C129XKCZAD I²C peripheral has support for High-speed operation as both a master and slave. High-Speed mode is configured by setting the HS bit in the **I²C Master Control/Status (I2CMCS)** register. High-Speed mode transmits data at a high bit rate with a 66.6%/33.3% duty cycle, but communication and arbitration are done at Standard, Fast mode, or Fast-mode plus speed, depending on which is selected by the user. When the HS bit in the **I2CMCS** register is set, current mode pull-ups are enabled.

The clock period can be selected using the equation below, but in this case, SCL_LP=2 and SCL_HP=1.

$$SCL_PERIOD = 2 \times (1 + TIMER_PRD) \times (SCL_LP + SCL_HP) \times CLK_PRD$$

So for example:

```
CLK_PRD = 25 ns
TIMER_PRD = 1
SCL_LP=2
SCL_HP=1
```

yields a SCL frequency of:

```
1/T = 3.33 Mhz
```

Table 21-3 on page 1437 gives examples of timer period and system clock in High-Speed mode. Note that the HS bit in the **I2CMTPR** register needs to be set for the TPR value to be used in High-Speed mode.

**Table 21-3. Examples of I²C Master Timer Period in High-Speed Mode**

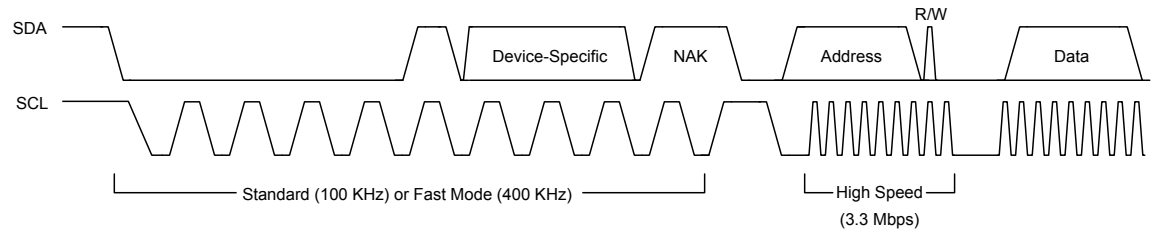
| System Clock | Timer Period | Transmission Mode |
|---|---|---|
| 40 MHz | 0x01 | 3.33 Mbps |
| 50 MHz | 0x02 | 2.77 Mbps |
| 80 MHz | 0x03 | 3.33 Mbps |

When operating as a master, the protocol is shown in Figure 21-7. The master is responsible for sending a master code byte in either Standard (100 Kbps) or Fast-mode (400 Kbps) before it begins

transferring in High-speed mode. The master code byte must contain data in the form of 0000.1XXX and is used to tell the slave devices to prepare for a High-speed transfer. The master code byte should never be acknowledged by a slave since it is only used to indicate that the upcoming data is going to be transferred at a higher data rate. To send the master code byte for a standard high-speed transfer, software should place the value of the master code byte into the **I2CMSA** register and write the **I2CMCS** register with a value of 0x13. If a high-speed burst transfer is required, then to send the master code byte, software should place the value of the master code byte into the **I2CMSA** register and write the **I2CMCS** register with 0x50. Either configuration places the $I^2C$ master peripheral in High-speed mode, and all subsequent transfers (until STOP) are carried out at High-speed data rate using the normal **I2CMCS** command bits, without setting the HS bit in the **I2CMCS** register. Again, setting the HS bit in the **I2CMCS** register is only necessary during the master code byte.

When operating as a High-speed slave, there is no additional software required.

**Figure 21-7. High-Speed Data Format**




**Note:** High-Speed mode is 3.4 Mbps, provided correct system clock frequency is set and there is appropriate pull strength on SCL and SDA lines.

### 21.3.3 Interrupts

The $I^2C$ can generate interrupts when the following conditions are observed in the Master Module:

- Master transaction completed (RIS bit)
- Master arbitration lost (ARBLOSTRIS bit)
- Master Address/Data NACK (NACKRIS bit)
- Master bus timeout (CLKRIS bit)
- Next byte request (RIS bit)
- Stop condition on bus detected (STOPRIS bit)
- Start condition on bus detected (STARTRIS bit)
- RX DMA interrupt pending (DMARXRIS bit)
- TX DMA interrupt pending (DMATXRIS bit)
- Trigger value for FIFO has been reached and a TX FIFO request interrupt is pending (TXRIS bit)
- Trigger value for FIFO has been reached and a RX FIFO request interrupt is pending (RXRIS bit)

- Transmit FIFO is empty (TXFERIS bit)
- Receive FIFO is full (RXFFRIS bit)

Interrupts are generated when the following conditions are observed in the Slave Module:

- Slave transaction received (DATARIS bit)
- Slave transaction requested (DATARIS bit)
- Slave next byte transfer request (DATARIS bit)
- Stop condition on bus detected (STOPRIS bit)
- Start condition on bus detected (STARTRIS bit)
- RX DMA interrupt pending (DMARXRIS bit)
- TX DMA interrupt pending (DMATXRIS bit)
- Programmable trigger value for FIFO has been reached and a TX FIFO request interrupt is pending (TXRIS bit)
- Programmable trigger value for FIFO has been reached and a RX FIFO request interrupt is pending (RXRIS bit)
- Transmit FIFO is empty (TXFERIS bit)
- Receive FIFO is full (RXFFRIS bit)

The $I^2C$ master and $I^2C$ slave modules have separate interrupt registers. Interrupts can be masked by clearing the appropriate bit in the **I2CMIMR** or **I2CSIMR** register. Note that the RIS bit in the **Master Raw Interrupt Status (I2CMRIS)** register and the DATARIS bit in the **Slave Raw Interrupt Status (I2CSRIS)** register have multiple interrupt causes including a next byte transfer request interrupt. This interrupt is generated when both master and slave are requesting a receive or transmit transaction.

### 21.3.4 Loopback Operation

The $I^2C$ modules can be placed into an internal loopback mode for diagnostic or debug work by setting the LPBK bit in the **$I^2C$ Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and are tied to the SDA and SCL signals of the slave module to allow internal testing of the device without having to go through I/O.

### 21.3.5 FIFO and µDMA Operation

Both the master and the slave module have the capability to access two 8-byte FIFOs that can be used in conjunction with the µDMA for fast transfer of data. The transmit (TX) FIFO and receive (RX) FIFO can be independently assigned to either the $I^2C$ master or $I^2C$ slave. Thus, the following FIFO assignments are allowed:

- The transmit and receive FIFOs can be assigned to the master
- The transmit and receive FIFOs can be assigned to the slave

- The transmit FIFO can be assigned to the master, while the receive FIFO is assigned to the slave and vice versa.

In most cases, both FIFOs will be assigned to either the master or the slave. The FIFO assignment is configured by programming the TXASGNMT and RXASGNMT bit in the **I2C FIFO Control (I2CFIFOCTL)** register.

Each FIFO has a programmable threshold point which indicates when the FIFO service interrupt should be generated. Additionally, a FIFO receive full and transmit empty interrupt can be enabled in the Interrupt Mask **(I2CxIMR)** registers of both the master and slave. Note that if we clear the TXFERIS interrupt (by setting the TXFEIC bit) when the TX FIFO is empty, the TXFERIS interrupt does not reassert even though the TX FIFO remains empty in this situation.

When a FIFO is not assigned to a master or a slave module, the FIFO interrupt and status signals to the module are forced to a state that indicates the FIFO is empty. For example, if the TX FIFO is assigned to the master module, the status signals to the slave transmit interface indicates that the FIFO is empty.

**Note:** The FIFOs must be empty when reassigning the FIFOs for proper functionality

#### 21.3.5.1 Master Module Burst Mode

A BURST command is provided for the master module which allows a sequence of data transfers using the µDMA (or software, if desired) to handle the data in the FIFO. The BURST command is enabled by setting the BURST bit in the **Master Control/Status (I2CMCS)** register. The number of bytes transferred by a BURST request is programmed in the **I2C Master Burst Length (I2CMBLEN)** register and a copy of this value is automatically written to the **I2C Master Burst Count (I2CMBCNT)** register to be used as a down-counter during the BURST transfer. The bytes written to the **I2C FIFO Data (I2CFIFODATA)** register are transferred to the RX FIFO or TX FIFO depending on whether a transmit or receive is being executed. If data is NACKed during a BURST and the STOP bit is set in the **I2CMCS** register, the transfer terminates. If the STOP bit is not set, the software application must issue a repeated STOP or START when a NACK interrupt is asserted. In the case of a NACK, the **I2CMBCNT** register can be used to determine the amount of data that was transferred prior to the BURST termination. If the Address is NACKed during a transfer, then a STOP is issued.

***Master Module µDMA Functionality***

When the **Master Control/Status (I2CMCS)** register is set to enable BURST and the master I2C µDMA channel is enabled in the **DMA Channel Map Select n (DMACHMAPn)** registers in the µDMA, the master control module will assert either the internal single µDMA request signal (dma_sreq) or multiple µDMA request signal (dma_req) to the µDMA. Note that there are separate dma_req and dma_sreq signals for transmit and receive. A single µDMA request (dma_sreq) will be asserted by the Master module when the Rx FIFO has at least one data byte present in the FIFO and/or when the Tx FIFO has at least one space available to fill. The dma_req (or Burst) signal will be asserted when Rx FIFO fill level is higher than trigger level and/or the Tx FIFO burst length remaining is less than 4 bytes and the FIFO fill level is less than trigger level. If a single transfer or BURST operation has completed, the µDMA sends a dma_done signal to the master module represented by the DMATX/DMARX interrupts in the **I2CMIMR**/**I2CMRIS**/**I2CMMIS**/**I2CMICR** registers.

If the µDMA I2C channel is disabled and software is used to handle the BURST command, software can read the **FIFO Status (I2CFIFOSTAT)** Register and the **Master Burst Count (I2CMBC)** register to determine whether the FIFO needs servicing during the BURST transaction. A trigger value can be programmed in the **I2CFIFOCTL** register to allow for interrupts at various fill levels of the FIFOs.

The NACK and ARBLOST bits in the interrupt status registers can be enabled to indicate no acknowledgement of data transfer or an arbitration loss on the bus.

When the Master module is transmitting FIFO data, software can fill the Tx FIFO in advance of setting the BURST bit in the **I2CMCS** register. If the FIFO is empty when the µDMA is enabled for BURST mode, the dma_req and dma_sreq both assert (assuming the **I2CMBLEN** register is programmed to at least 4 bytes and the Tx FIFO fill level is less than the trigger set). If the **I2CMBLEN** register value is less than 4 and the Tx FIFO is not full but more than trigger level, only dma_sreq will assert. Single requests will be generated as required to keep the FIFO full until the number of bytes specified in the **I2CMBLEN** register has been transferred to the FIFO (and the **I2CMBCOUNT** register reaches 0x0). At this point, no further requests are generated until the next BURST command is issued. If the µDMA is disabled, FIFOs will be serviced based on the interrupts active in the Master interrupt status registers, the FIFO trigger values shown in the **I2CFIFOSTATUS** register and completion of a BURST transfer.

When the Master module is receiving FIFO data, the Rx FIFO is initially empty and no requests are asserted. If data is read from the slave and placed into the Rx FIFO, the dma_sreq signal to the µDMA is asserted to indicate there is data to be transferred. If the Rx FIFO contains at least 4 bytes, the dma_req signal is also asserted. The µDMA will continue to transfer data out of the Rx FIFO until it has reached the amount of bytes programmed in the **I2CMBLEN** register.

**Note:** The TXFEIM interrupt mask bit in the **I2CMIMR** register should be clear (masking the TXFE interrupt) when the master is performing an RX Burst from the RXFIFO and should be unmasked before starting a TX FIFO transfers.

#### 21.3.5.2 Slave Module

The slave module also has the capability to use the µDMA in Rx and Tx FIFO data transfers. If the Tx FIFO is assigned to the slave module and the TXFIFO bit is set in the **I2CSCSR** register, the slave module will generate a single µDMA request, dma_sreq, if the master module requests the next byte transfer. If the FIFO fill level is less than the trigger level, a µDMA multiple transfer request, dma_req, will be asserted to continue data transfers from the µDMA.

If the Rx FIFO is assigned to the slave module and the RXFIFO bit is set in the **I2CSCSR** register, then the slave module will generate a signal µDMA request, dma_sreq, if there is any data to be transferred. The dma_req signal will be asserted when the Rx FIFO has more data than the trigger level programmed by the RXTRIG bit in the **I2CFIFOCTL** register.

**Note:** Best practice recommends that an application should not switch between the **I2CSDR** register and TX FIFO or vice versa for successive transactions.

### 21.3.6 Command Sequence Flow Charts

This section details the steps required to perform the various $I^2C$ transfer types in both master and slave mode. Refer to Table 21-5 on page 1460 for further sequence information.

#### 21.3.6.1 $I^2C$ Master Command Sequences

The figures that follow show the command sequences available for the $I^2C$ master.

**Figure 21-8. Master Single TRANSMIT**

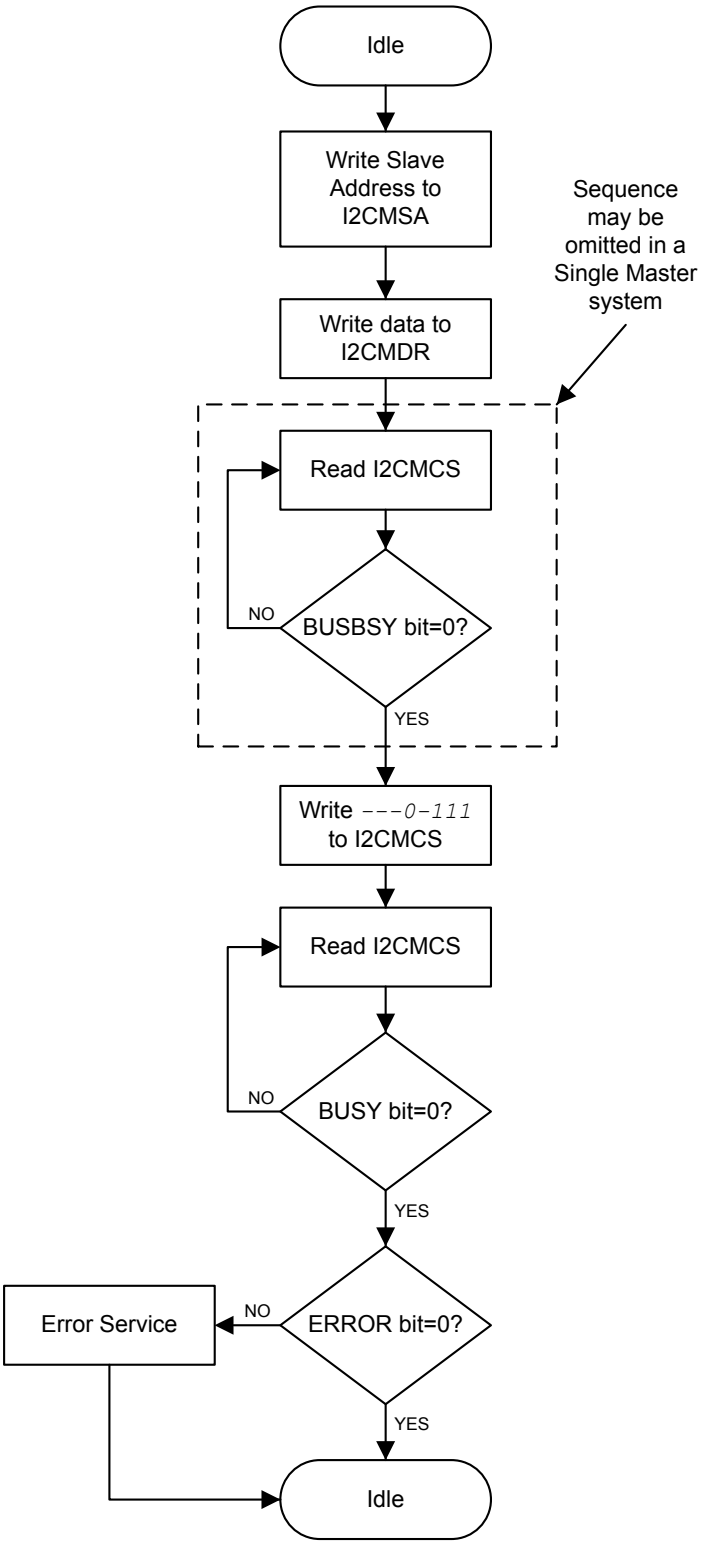


**Figure 21-9. Master Single RECEIVE**

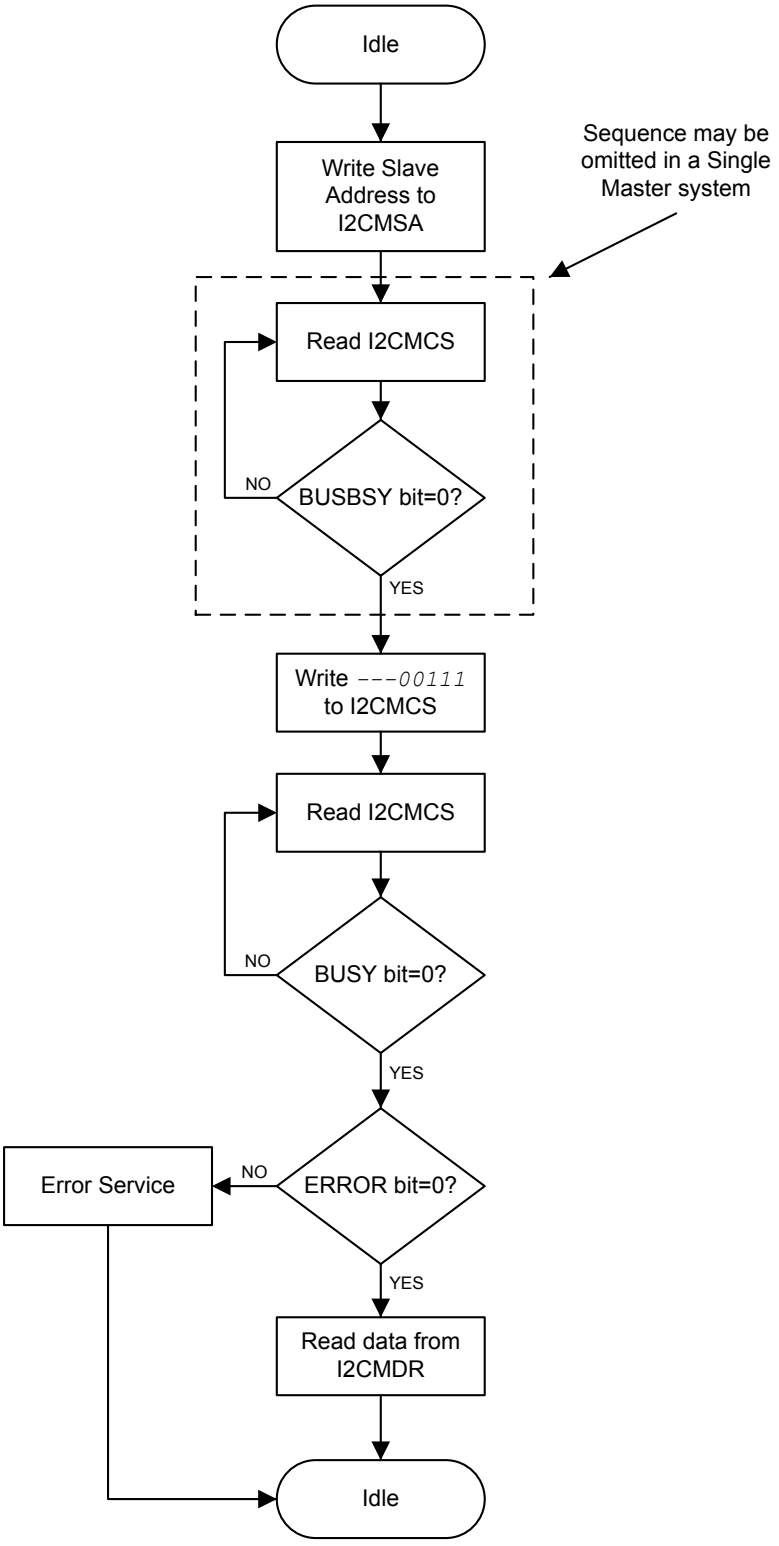


**Figure 21-10. Master TRANSMIT of Multiple Data Bytes**

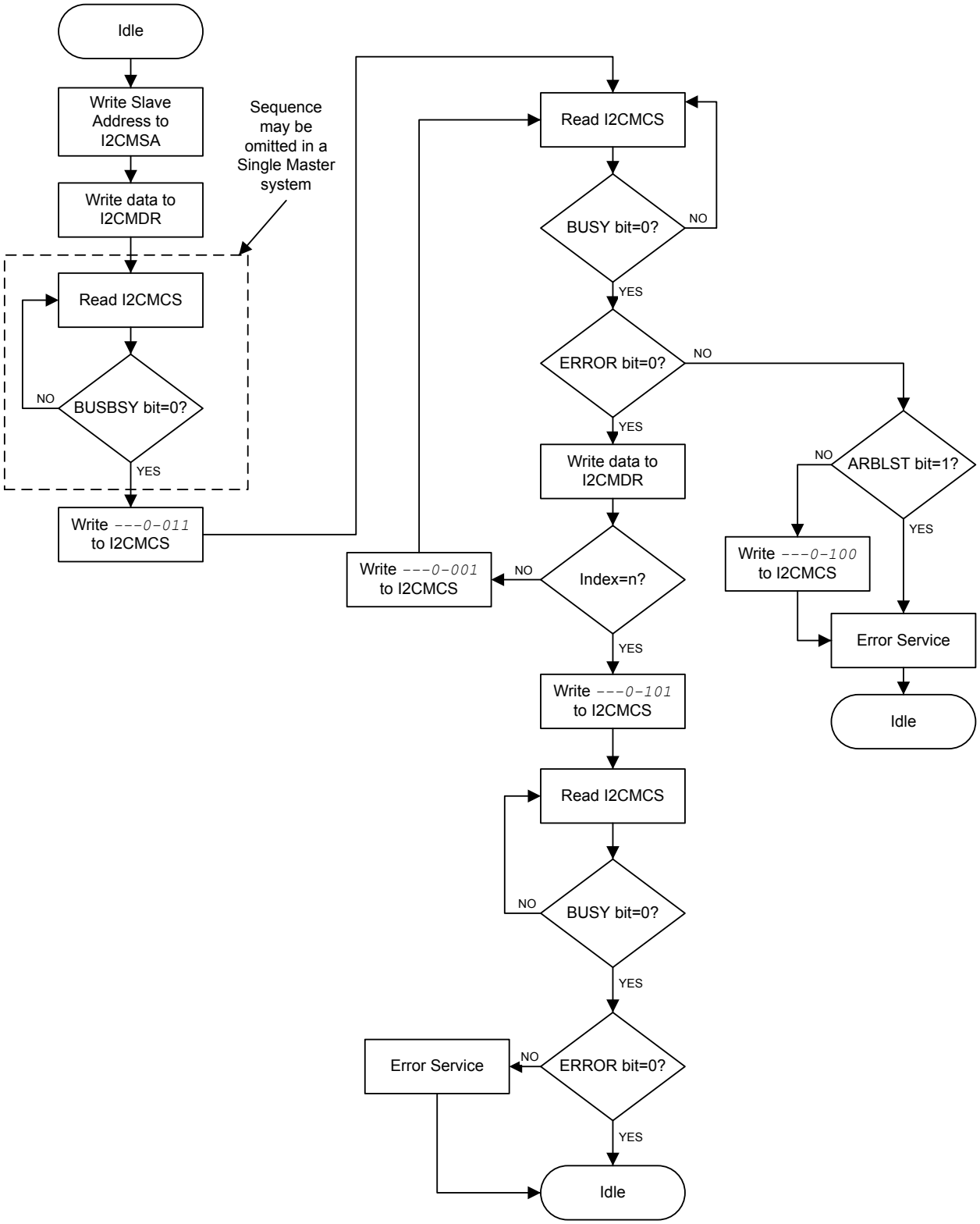


**Figure 21-11. Master RECEIVE of Multiple Data Bytes**

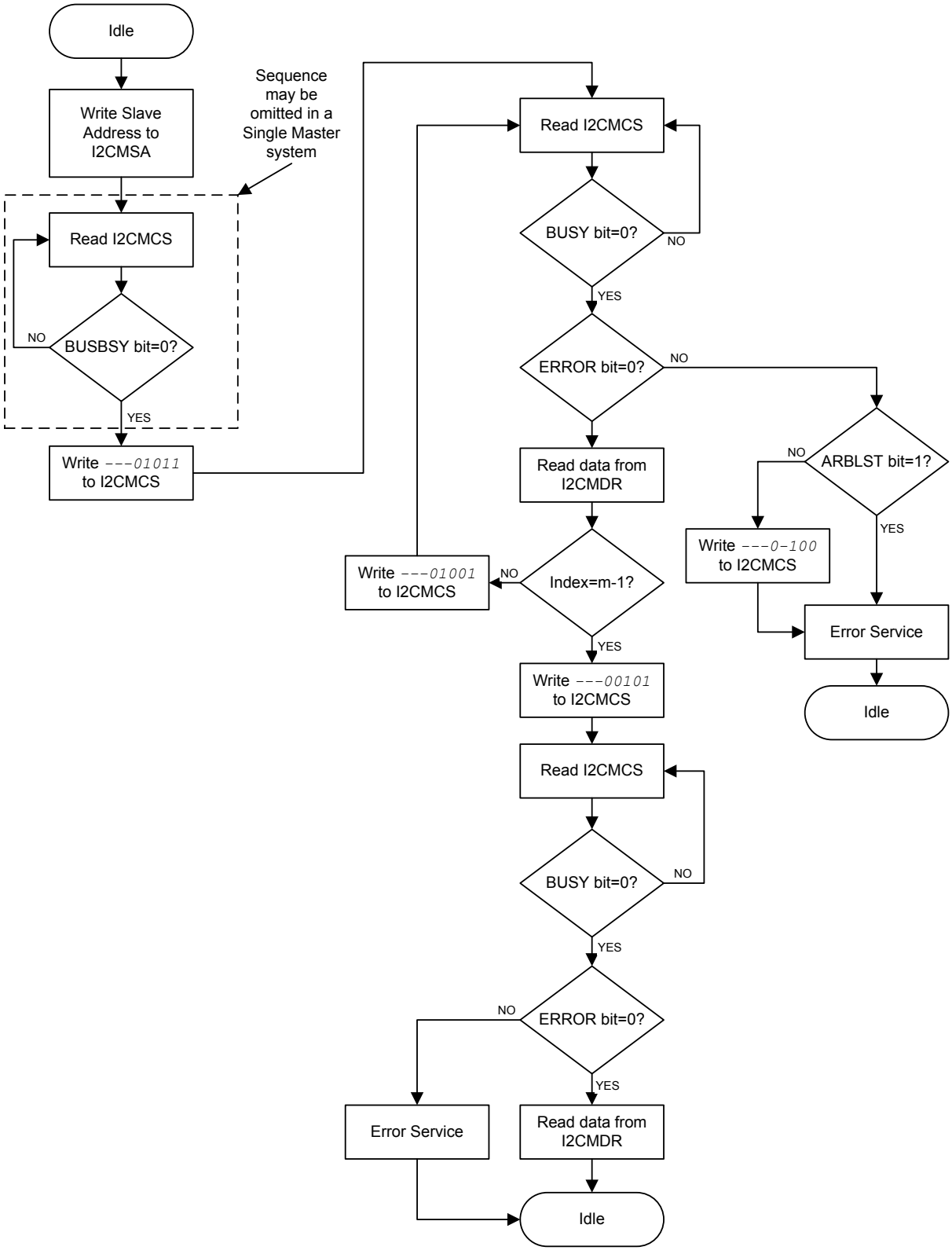


**Figure 21-12. Master RECEIVE with Repeated START after Master TRANSMIT**

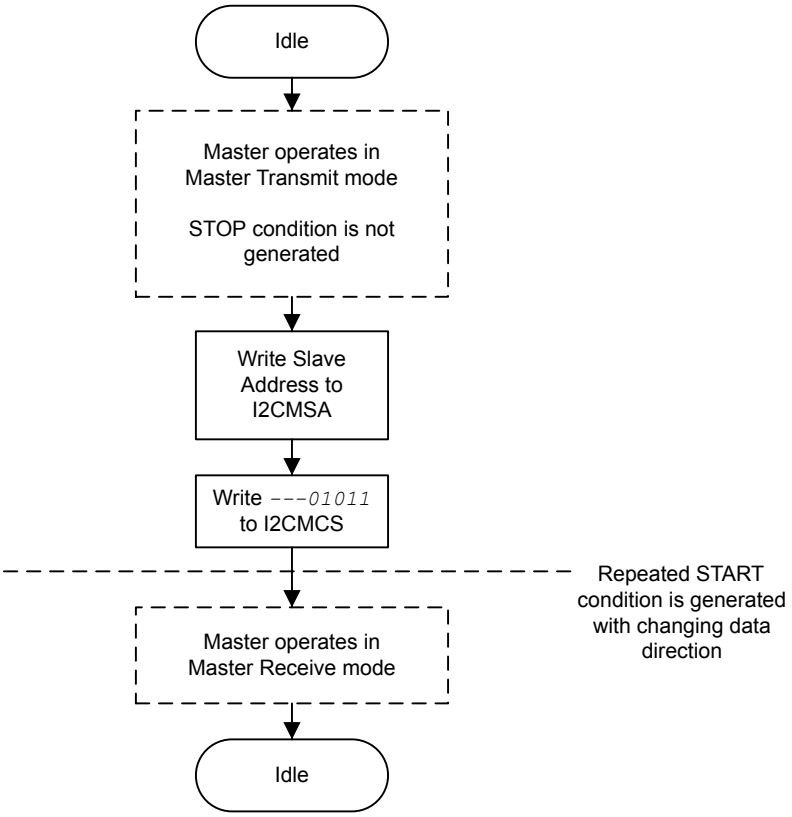


**Figure 21-13. Master TRANSMIT with Repeated START after Master RECEIVE**

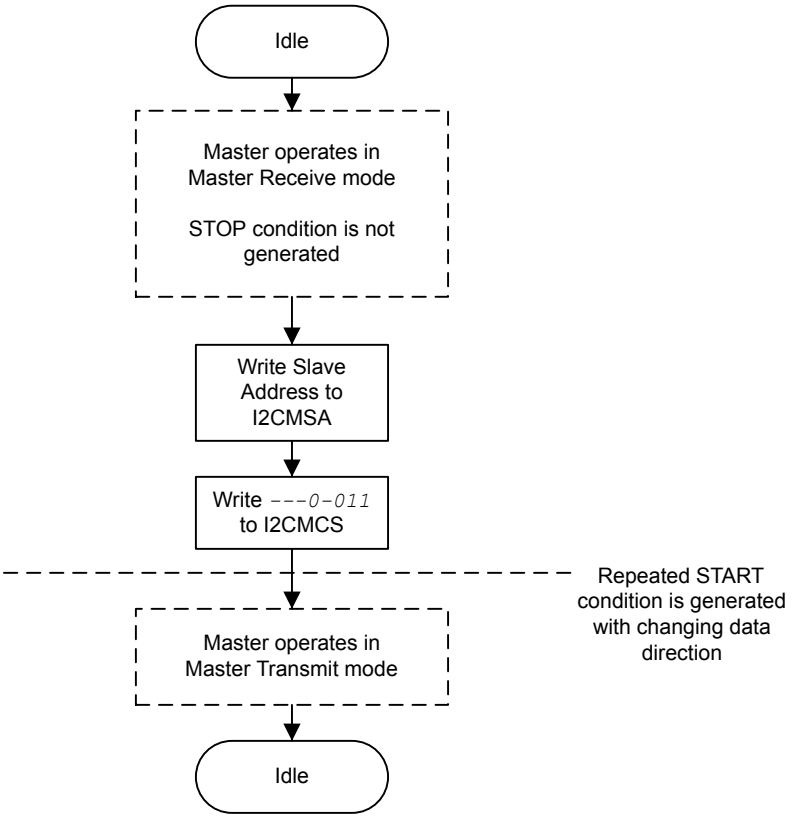
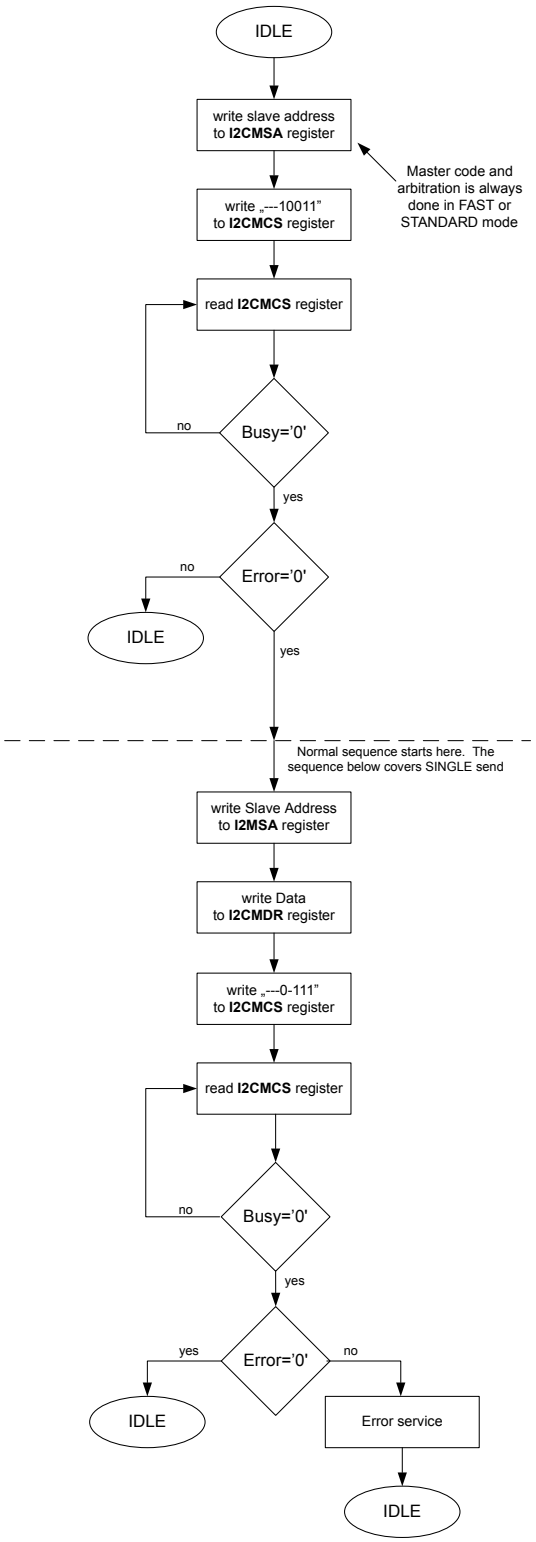


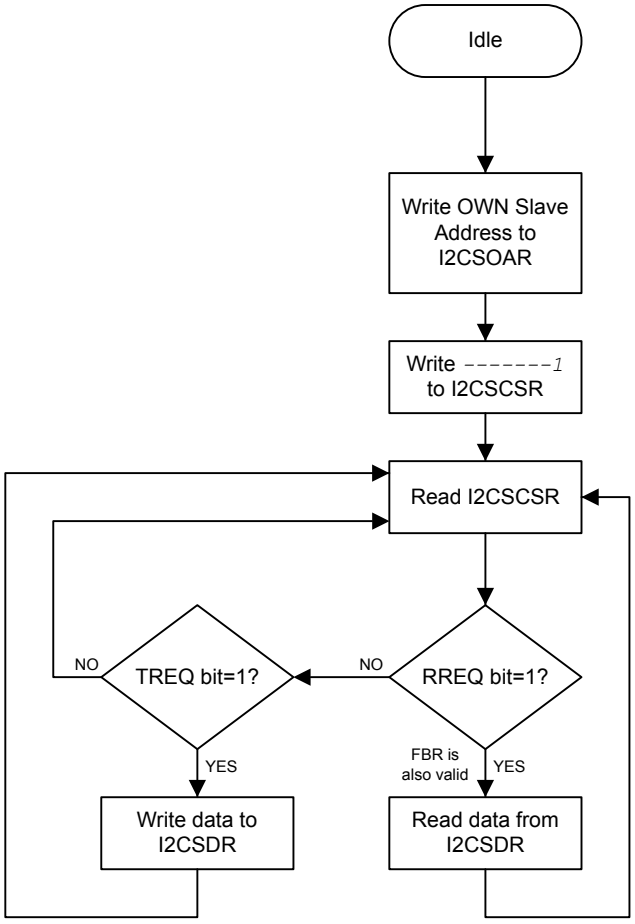
**Figure 21-14. Standard High Speed Mode Master Transmit**



#### 21.3.6.2 I²C Slave Command Sequences

Figure 21-15 on page 1449 presents the command sequence available for the I²C slave.

**Figure 21-15. Slave Command Sequence**




## 21.4 Initialization and Configuration

### 21.4.1 Configure the I²C Module to Transmit a Single Byte as a Master

The following example shows how to configure the I²C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I²C clock using the **RCGCI2C** register in the System Control module (see page 408).
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register in the System Control module (see page 399). To find out which GPIO port to enable, refer to Table 31-5 on page 2081.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 803). To determine which GPIOs to configure, see Table 31-4 on page 2064.

4. Enable the I2CSDA pin for open-drain operation. See page 808.
5. Configure the PMCn fields in the **GPIOPCTL** register to assign the I2C signals to the appropriate pins. See page 820 and Table 31-5 on page 2081.
6. Initialize the I2C Master by writing the **I2CMCR** register with a value of 0x0000.0010.
7. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

   ```
   TPR = (System Clock/(2*(SCL_LP + SCL_HP)*SCL_CLK))-1;
   TPR = (20MHz/(2*(6+4)*100000))-1;
   TPR = 9
   ```

   Write the **I2CMTPR** register with the value of 0x0000.0009.
8. Specify the slave address of the master and that the next operation is a Transmit by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
9. Place data (byte) to be transmitted in the data register by writing the **I2CMDR** register with the desired data.
10. Initiate a single byte transmit of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
11. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.
12. Check the ERROR bit in the **I2CMCS** register to confirm the transmit was acknowledged.

### 21.4.2 Configure the I2C Master to High Speed Mode

To configure the I2C master to High Speed mode:

1. Enable the I2C clock using the **RCGCI2C** register in the System Control module (see page 408).
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register in the System Control module (see page 399). To find out which GPIO port to enable, refer to Table 31-5 on page 2081.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 803). To determine which GPIOs to configure, see Table 31-4 on page 2064.
4. Enable the I2CSDA pin for open-drain operation. See page 808.
5. Configure the PMCn fields in the **GPIOPCTL** register to assign the I2C signals to the appropriate pins. See page 820 and Table 31-5 on page 2081.
6. Initialize the I2C Master by writing the **I2CMCR** register with a value of 0x0000.0010.
7. Set the desired SCL clock speed of 3.33 Mbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```
TPR = (System Clock/(2*(SCL_LP + SCL_HP)*SCL_CLK))-1;
TPR = (80 MHz/(2*(2+1)*3330000))-1;
TPR = 3
```

Write the **I2CMTPR** register with the value of 0x0000.0003.

8. To send the master code byte, software should place the value of the master code byte into the I2CMSA register and write the I2CMCS register with the following value depending on the required operation:
   - For Standard High-Speed mode, the **I2CMCS** register should be written with 0x13.
   - For Burst High-Speed mode, the **I2CMCS** register should be written with 0x50.
9. This places the I2C master peripheral in High-speed mode, and all subsequent transfers (until STOP) are carried out at High-speed data rate using the normal **I2CMCS** command bits, without setting the HS bit in the **I2CMCS** register.
10. The transaction is ended by setting the STOP bit in the **I2CMCS** register.
11. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.
12. Check the ERROR bit in the **I2CMCS** register to confirm the transmit was acknowledged.

## 21.5 Register Map

Table 21-4 on page 1452 lists the $I^2C$ registers. All addresses given are relative to the $I^2C$ base address:

- $I^2C$ 0: 0x4002.0000
- $I^2C$ 1: 0x4002.1000
- $I^2C$ 2: 0x4002.2000
- $I^2C$ 3: 0x4002.3000
- $I^2C$ 4: 0x400C.0000
- $I^2C$ 5: 0x400C.1000
- $I^2C$ 6: 0x400C.2000
- $I^2C$ 7: 0x400C.3000
- $I^2C$ 8: 0x400B.8000
- $I^2C$ 9: 0x400B.9000

Note that the $I^2C$ module clock must be enabled before the registers can be programmed (see page 408). There must be a delay of 3 system clocks after the $I^2C$ module clock is enabled before any $I^2C$ module registers are accessed.

The hw_i2c.h file in the TivaWare™ Driver Library uses a base address of 0x800 for the $I^2C$ slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that TivaWare™ for C Series uses an offset between 0x000 and 0x018 with the slave base address.

### Table 21-4. Inter-Integrated Circuit ($I^2C$) Interface Register Map

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **$I^2C$ Master** | | | | | |
| 0x000 | I2CMSA | RW | 0x0000.0000 | I2C Master Slave Address | 1454 |
| 0x004 | I2CMCS | RW | 0x0000.0020 | I2C Master Control/Status | 1455 |
| 0x008 | I2CMDR | RW | 0x0000.0000 | I2C Master Data | 1464 |
| 0x00C | I2CMTPR | RW | 0x0000.0001 | I2C Master Timer Period | 1465 |
| 0x010 | I2CMIMR | RW | 0x0000.0000 | I2C Master Interrupt Mask | 1467 |
| 0x014 | I2CMRIS | RO | 0x0000.0000 | I2C Master Raw Interrupt Status | 1470 |
| 0x018 | I2CMMIS | RO | 0x0000.0000 | I2C Master Masked Interrupt Status | 1473 |
| 0x01C | I2CMICR | WO | 0x0000.0000 | I2C Master Interrupt Clear | 1476 |
| 0x020 | I2CMCR | RW | 0x0000.0000 | I2C Master Configuration | 1478 |
| 0x024 | I2CMCLKOCNT | RW | 0x0000.0000 | I2C Master Clock Low Timeout Count | 1479 |
| 0x02C | I2CMBMON | RO | 0x0000.0003 | I2C Master Bus Monitor | 1480 |
| 0x030 | I2CMBLEN | RW | 0x0000.0000 | I2C Master Burst Length | 1481 |
| 0x034 | I2CMBCNT | RO | 0x0000.0000 | I2C Master Burst Count | 1482 |
| **$I^2C$ Slave** | | | | | |
| 0x800 | I2CSOAR | RW | 0x0000.0000 | I2C Slave Own Address | 1483 |
| 0x804 | I2CSCSR | RO | 0x0000.0000 | I2C Slave Control/Status | 1484 |
| 0x808 | I2CSDR | RW | 0x0000.0000 | I2C Slave Data | 1487 |
| 0x80C | I2CSIMR | RW | 0x0000.0000 | I2C Slave Interrupt Mask | 1488 |
| 0x810 | I2CSRIS | RO | 0x0000.0000 | I2C Slave Raw Interrupt Status | 1490 |
| 0x814 | I2CSMIS | RO | 0x0000.0000 | I2C Slave Masked Interrupt Status | 1493 |
| 0x818 | I2CSICR | WO | 0x0000.0000 | I2C Slave Interrupt Clear | 1496 |
| 0x81C | I2CSOAR2 | RW | 0x0000.0000 | I2C Slave Own Address 2 | 1498 |
| 0x820 | I2CSACKCTL | RW | 0x0000.0000 | I2C Slave ACK Control | 1499 |
| **$I^2C$ Status and Control** | | | | | |
| 0xF00 | I2CFIFODATA | RW | 0x0000.0000 | I2C FIFO Data | 1500 |
| 0xF04 | I2CFIFOCTL | RW | 0x0004.0004 | I2C FIFO Control | 1502 |
| 0xF08 | I2CFIFOSTATUS | RO | 0x0001.0005 | I2C FIFO Status | 1504 |
| 0xFC0 | I2CPP | RO | 0x0000.0001 | I2C Peripheral Properties | 1506 |
| 0xFC4 | I2CPC | RO | 0x0000.0001 | I2C Peripheral Configuration | 1507 |

## 21.6 Register Descriptions ($I^2C$ Master)

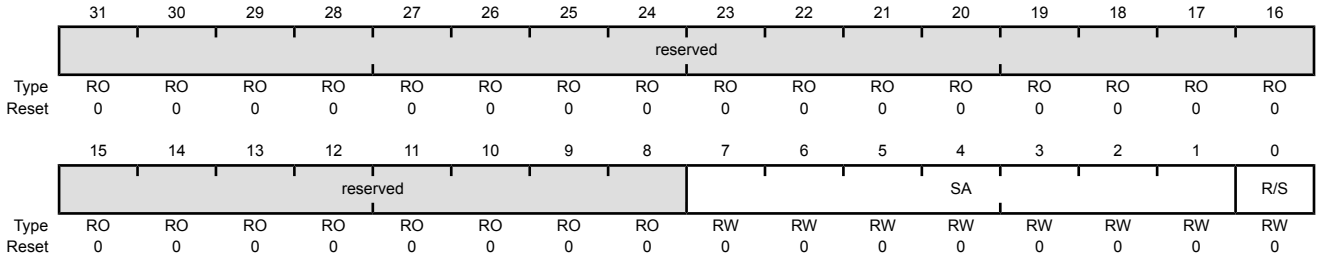
The remainder of this section lists and describes the $I^2C$ master registers, in numerical order by address offset.

## Register 1: I2C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Transmit (Low).

I2C Master Slave Address (I2CMSA)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:1 | SA | RW | 0x00 | I2C Slave Address<br>This field specifies bits A6 through A0 of the slave address. |
| 0 | R/S | RW | 0 | Receive/Send<br>The R/S bit specifies if the next master operation is a Receive (High) or Transmit (Low).<br><br>Value Description<br>0 Transmit<br>1 Receive |

## Register 2: I²C Master Control/Status (I2CMCS), offset 0x004

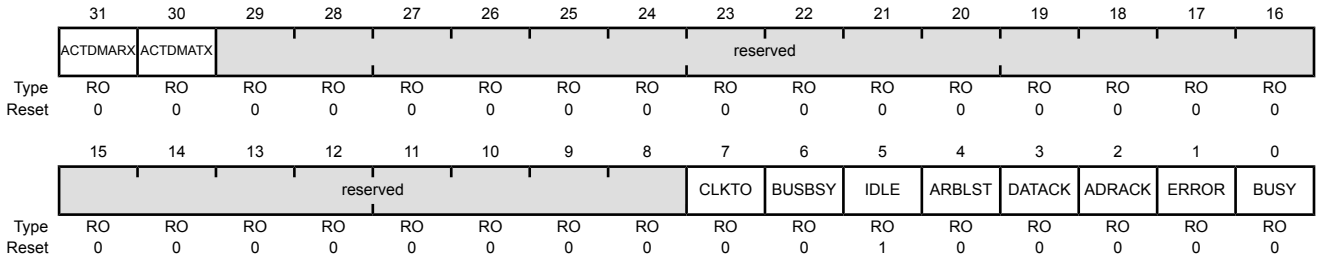
This register accesses status bits when read and control bits when written. When read, the status register indicates the state of the I²C bus controller. When written, the control register configures the I²C controller operation.

The START bit generates the START or REPEATED START condition. The STOP bit determines if the cycle stops at the end of the data cycle or continues to the next transfer cycle, which could be a repeated START. To generate a single transmit cycle, the **I²C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and this register 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), an interrupt becomes active and the data may be read from the **I2CMDR** 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. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

### Read-Only Status Register

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x004
Type RO, reset 0x0000.0020




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ACTDMARX | RO | 0 | DMA RX Active Status<br>Value Description<br>0 DMA RX is not active<br>1 DMA RX is active. |
| 30 | ACTDMATX | RO | 0 | DMA TX Active Status<br>Value Description<br>0 DMA TX is not active<br>1 DMA TX is active. |

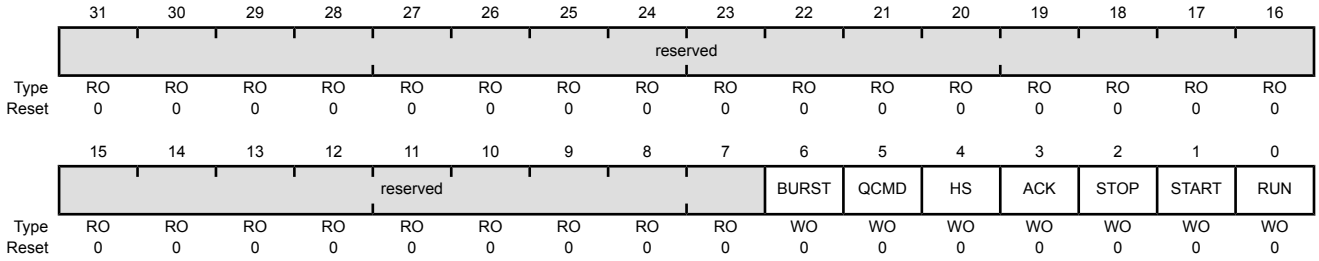
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 29:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7 | CLKTO | RO | 0 | Clock Timeout Error<br><br>Value Description<br>0 No clock timeout error.<br>1 The clock timeout error has occurred.<br><br>This bit is cleared when the master sends a STOP condition or if the $I^2C$ master is reset. |
| 6 | BUSBSY | RO | 0 | Bus Busy<br><br>Value Description<br>0 The $I^2C$ bus is idle.<br>1 The $I^2C$ bus is busy.<br><br>The bit changes based on the START and STOP conditions. |
| 5 | IDLE | RO | 1 | $I^2C$ Idle<br><br>Value Description<br>0 The $I^2C$ controller is not idle.<br>1 The $I^2C$ controller is idle. |
| 4 | ARBLST | RO | 0 | Arbitration Lost<br><br>Value Description<br>0 The $I^2C$ controller won arbitration.<br>1 The $I^2C$ controller lost arbitration. |
| 3 | DATACK | RO | 0 | Acknowledge Data<br><br>Value Description<br>0 The transmitted data was acknowledged<br>1 The transmitted data was not acknowledged. |
| 2 | ADRACK | RO | 0 | Acknowledge Address<br><br>Value Description<br>0 The transmitted address was acknowledged<br>1 The transmitted address was not acknowledged. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | ERROR | RO | 0 | Error<br><br>Value Description<br>0 No error was detected on the last operation.<br>1 An error occurred on the last operation.<br><br>The error can be from the slave address not being acknowledged or the transmit data not being acknowledged. |
| 0 | BUSY | RO | 0 | $I^2C$ Busy<br><br>Value Description<br>0 The controller is idle.<br>1 The controller is busy.<br><br>When the BUSY bit is set, the other status bits are not valid. |

## Write-Only Control Register

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x004
Type WO, reset 0x0000.0020




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0 | 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. |
| 6 | BURST | WO | 0 | Burst Enable<br><br>Value Description<br>0 Burst operation is disabled.<br>1 The master is enabled to burst using the receive and transmit FIFOs. See field decoding in Table 21-5 on page 1460.<br><br>Note that the BURST and RUN bits are mutually exclusive. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | QCMD | WO | 0 | Quick Command<br>Value Description<br>0 Bus transaction is not a quick command.<br>1 The bus transaction is a quick command. To execute a quick command, the START, STOP and RUN bits also need to be set. After the quick command is issued, the master generates a STOP. |
| 4 | HS | WO | 0 | High-Speed Enable<br>Value Description<br>0 The master operates in Standard, Fast mode, or Fast mode plus as selected by using a value in the **I2CMTPR** register that results in an SCL frequency of 100 kbps for Standard mode, 400 kbps for Fast mode, or 1 Mpbs for Fast mode plus.<br>1 The master operates in High-Speed mode with transmission speeds up to 3.33 Mbps. |
| 3 | ACK | WO | 0 | Data Acknowledge Enable<br>Value Description<br>0 The received data byte is not acknowledged automatically by the master.<br>1 The received data byte is acknowledged automatically by the master. See field decoding in Table 21-5 on page 1460. |
| 2 | STOP | WO | 0 | Generate STOP<br>Value Description<br>0 The controller does not generate the STOP condition.<br>1 The controller generates the STOP condition. See field decoding in Table 21-5 on page 1460. |
| 1 | START | WO | 0 | Generate START<br>Value Description<br>0 The controller does not generate the START condition.<br>1 The controller generates the START or repeated START condition. See field decoding in Table 21-5 on page 1460. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RUN | WO | 0 | $I^2C$ Master Enable |

| Value | Description |
|---|---|
| 0 | In standard and high speed mode, this encoding means the master is unable to transmit or receive data.<br>In Burst mode, this bit is not used and must be set to 0. |
| 1 | The master is able to transmit or receive data.<br>Note that this bit cannot be set in Burst mode. See field decoding in Table 21-5 on page 1460. |

Note that the BURST and RUN bits are mutually exclusive.

The Table 21-5 on page 1460 can be read from left to right to determine the next state after programming bits in the **I2CMSA** and **I2CMCS** registers.

**Table 21-5. Write Field Decoding for I2CMCS[6:0]**

| Current State | I2CMSA[0] | I2CMCS[6:0] | | | | | | | Next State Description |
|---|---|---|---|---|---|---|---|---|---|
| | R/S | BURST | QCCMD | HS | ACK | STOP | START | RUN | |
| Idle | 0 | 0 | 0 | 0 | X[a] | 0 | 1 | 1 | START condition followed by TRANSMIT (master goes to the Master Transmit state). |
| | 0 | 0 | 0 | 0 | X | 1 | 1 | 1 | START condition followed by a TRANSMIT and STOP condition (master remains in Idle state). |
| | 0 | 1 | 0 | 0 | X | 0 | 1 | 0 | START condition followed by N FIFO-serviced TRANSMITs (master goes to the Master Transmit state). |
| | 0 | 1 | 0 | 0 | X | 1 | 1 | 0 | START condition followed by N FIFO-serviced TRANSMITs and STOP condition (master remains in Idle state). |
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state). |
| | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | Quick Command (Send). After Quick Command is executed, the master returns to Idle state. |
| | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | Quick Command (Receive). After Quick Command is executed, the master returns to Idle state. |
| | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | START condition followed by RECEIVE and STOP condition (master remains in Idle state). |
| | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | START condition followed by RECEIVE (master goes to the Master Receive state). |
| | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE operation (master goes to the Master Receive state). |
| | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE and STOP condition (master remains in Idle state). |
| | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | START condition followed by N FIFO-serviced RECEIVE operations (master goes to the Master Receive state). |
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | START/RUN condition where master byte is sent with no ACK; followed by High Speed transmit Operation. All subsequent transfers are carried out using normal transmit commands. |
| | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | RUN/BURST condition where master byte is sent with no ACK; followed by High Speed Burst transmit Operation. |
| | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | Illegal |
| | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | Illegal |
| | All other combinations not listed are non-operations. | | | | | | | | NOP |

**Table 21-5. Write Field Decoding for I2CMCS[6:0] *(continued)***

| Current State | I2CMSA[0] | I2CMCS[6:0] | | | | | | | Next State Description |
|---|---|---|---|---|---|---|---|---|---|
| | R/S | BURST | QCCMD | HS | ACK | STOP | START | RUN | |
| Master Transmit | X | 0 | 0 | 0 | X | 0 | 0 | 1 | TRANSMIT operation (master remains in Master Transmit state). |
| | X | 0 | 0 | 0 | X | 1 | 0 | 0 | STOP condition (master goes to Idle state). |
| | X | 0 | 0 | 0 | X | 1 | 0 | 1 | TRANSMIT followed by STOP condition (master goes to Idle state). |
| | X | 1 | 0 | 0 | X | 0 | 0 | 0 | N FIFO-serviced TRANSMIT operations (master remains in Master Transmit state). |
| | X | 1 | 0 | 0 | X | 1 | 0 | 0 | N FIFO-serviced TRANSMIT operations followed by STOP condition (master goes to Idle state). |
| | 0 | 0 | 0 | 0 | X | 0 | 1 | 1 | Repeated START condition followed by a TRANSMIT (master remains in Master Transmit state). |
| | 0 | 0 | 0 | 0 | X | 1 | 1 | 1 | Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state). |
| | 0 | 1 | 0 | 0 | X | 0 | 1 | 0 | Repeated START condition followed by N FIFO-serviced TRANSMIT operations (master remains in Master Transmit state). |
| | 0 | 1 | 0 | 0 | X | 1 | 1 | 0 | Repeated START condition followed by N FIFO-serviced TRANSMIT operations and STOP condition (master goes to Idle state). |
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state). |
| | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Repeated START condition followed by a RECEIVE and STOP condition (master goes to Idle state). |
| | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | Repeated START condition followed by RECEIVE (master goes to Master Receive state). |
| | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | Repeated START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE operation (master goes to Master Receive state). |
| | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | Repeated START condition followed by N FIFO-serviced RECEIVE operations and STOP condition (master goes to Idle state). |
| | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | Repeated START condition followed by N FIFO-serviced RECEIVE operations (master goes to Master Receive state). |
| | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | Illegal. |
| | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | Illegal. |
| | All other combinations not listed are non-operations. | | | | | | | | NOP. |

**Table 21-5. Write Field Decoding for I2CMCS[6:0] *(continued)***

| Current State | I2CMSA[0] | I2CMCS[6:0] | | | | | | | Next State Description |
|---|---|---|---|---|---|---|---|---|---|
| | R/S | BURST | QCCMD | HS | ACK | STOP | START | RUN | |
| Master Receive | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 | RECEIVE operation with negative ACK (master remains in Master Receive state). |
| | X | 0 | 0 | 0 | X | 1 | 0 | 0 | STOP condition (master goes to Idle state).[b] |
| | X | 0 | 0 | 0 | 0 | 1 | 0 | 1 | RECEIVE followed by STOP condition (master goes to Idle state). |
| | X | 0 | 0 | 0 | 1 | 0 | 0 | 1 | RECEIVE operation (master remains in Master Receive state). |
| | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | N FIFO-serviced RECEIVE operations with negative ACK on the last RECEIVE (master remains in Master Receive state). |
| | X | 1 | 0 | 0 | 0 | 1 | 0 | 0 | N FIFO-serviced RECEIVE operations followed by STOP condition (master goes to Idle state). |
| | X | 1 | 0 | 0 | 1 | 0 | 0 | 0 | N FIFO-serviced RECEIVE operations (master remains in Master Receive state). |
| | X | 0 | 0 | 0 | 1 | 1 | 0 | 1 | Illegal. |
| | X | 1 | 0 | 0 | 1 | 1 | 0 | 0 | Illegal. |
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state). |
| | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state). |
| | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | Repeated START condition followed by RECEIVE (master remains in Master Receive state). |
| | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | Repeated START condition followed by N FIFO-serviced RECEIVE operations with a negative ACK on the last RECEIVE (master remains in Master Receive state). |
| | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | Repeated START condition followed by N FIFO-serviced RECEIVE operations and STOP condition (master goes to Idle state). |
| | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | Repeated START condition followed by N FIFO-serviced RECEIVE operations (master remains in Master Receive state). |
| | 0 | 0 | 0 | 0 | X | 0 | 1 | 1 | Repeated START condition followed by TRANSMIT (master goes to Master Transmit state). |
| | 0 | 0 | 0 | 0 | X | 1 | 1 | 1 | Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state). |
| | 0 | 1 | 0 | 0 | X | 0 | 1 | 0 | Repeated START condition followed by N FIFO-serviced TRANSMIT operations (master goes to Master Transmit state). |
| | 0 | 1 | 0 | 0 | X | 1 | 1 | 0 | Repeated START condition followed by N FIFO-serviced TRANSMIT operations and STOP condition (master goes to Idle state). |
| | All other combinations not listed are non-operations. | | | | | | | | NOP. |

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

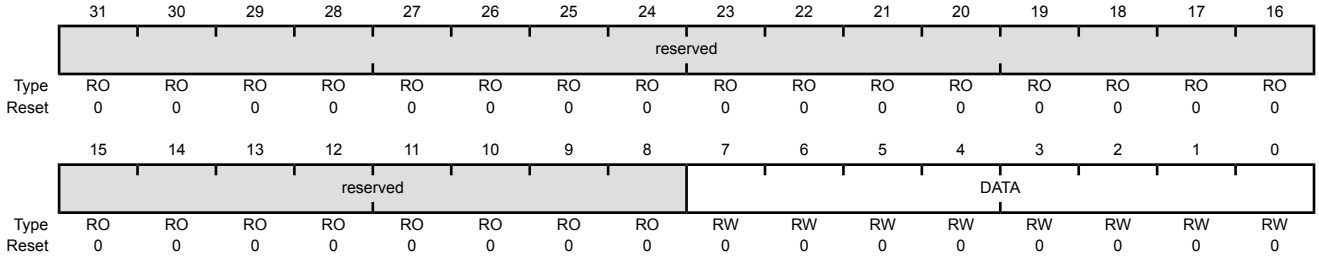
## Register 3: I2C Master Data (I2CMDR), offset 0x008

**Important:** This register is read-sensitive. See the register description for details.

This register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state. If the BURST bit is enabled in the **I2CMCS** register, then the **I2CFIFODATA** register is used for the current data transmit or receive value and this register is ignored.

I2C Master Data (I2CMDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x008
Type RW, reset 0x0000.0000



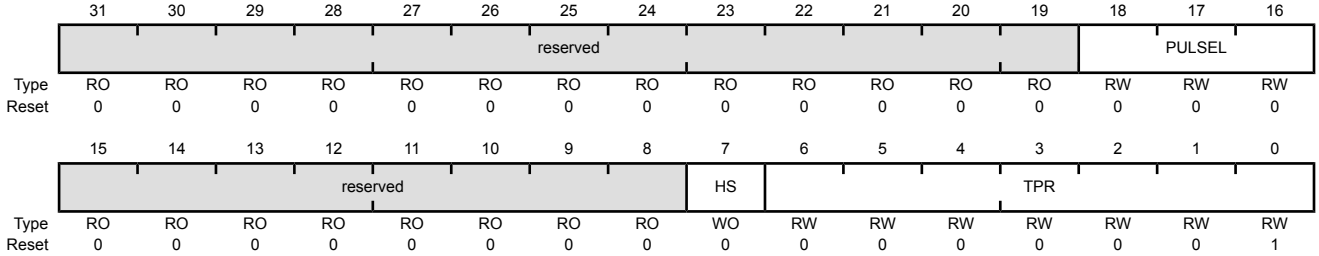

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | RW | 0x00 | This byte contains the data transferred during a transaction. |

## Register 4: I²C Master Timer Period (I2CMTPR), offset 0x00C

This register is programmed to set the timer period for the SCL clock and assign the SCL clock to either standard or high-speed mode.

I2C Master Timer Period (I2CMTPR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x00C
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x0000.00 | 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. |
| 18:16 | PULSEL | RW | 0x0 | Glitch Suppression Pulse Width<br>This field controls the pulse width select for glitch suppression on the SCL and SDA lines. The following values are the glitch suppression values in terms of system clocks.<br><br>Value Description<br>0x0 Bypass<br>0x1 1 clock<br>0x2 2 clocks<br>0x3 3 clocks<br>0x4 4 clocks<br>0x5 8 clocks<br>0x6 16 clocks<br>0x7 31 clocks |
| 15:8 | reserved | RO | 0x0000.00 | 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. |

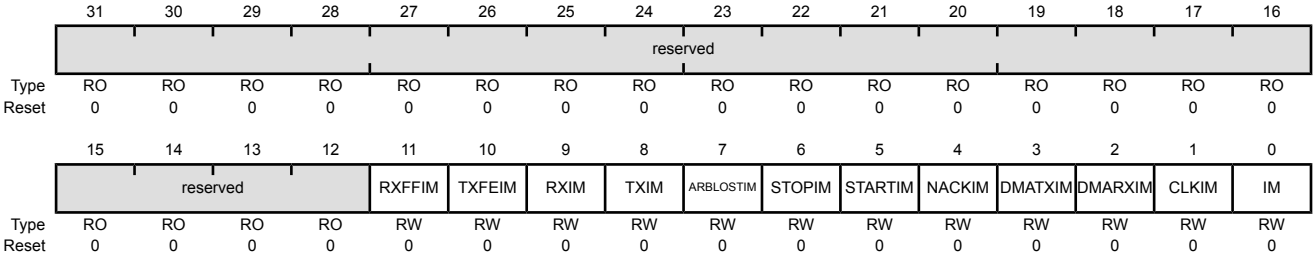
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | HS | WO | 0x0 | High-Speed Enable<br><br>Value Description<br>0 The SCL Clock Period set by TPR applies to Standard mode (100 Kbps), Fast-mode (400 Kbps), or Fast-mode plus (1 Mbps).<br>1 The SCL Clock Period set by TPR applies to High-speed mode (3.33 Mbps). |
| 6:0 | TPR | RW | 0x1 | Timer Period<br><br>This field is used in the equation to configure *SCL_PERIOD*:<br>*SCL_PERIOD* = 2×(1 + TPR)×(*SCL_LP* + *SCL_HP*)×*CLK_PRD*<br>where:<br>*SCL_PRD* is the SCL line period ($I^2C$ clock).<br>TPR is the Timer Period register value (range of 1 to 127).<br>*SCL_LP* is the SCL Low period (fixed at 6).<br>*SCL_HP* is the SCL High period (fixed at 4).<br>*CLK_PRD* is the system clock period in ns. |

## Register 5: I²C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Master Interrupt Mask (I2CMIMR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x010
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0 | 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. |
| 11 | RXFFIM | RW | 0 | Receive FIFO Full Interrupt Mask<br><br>Value Description<br>0 The RXFFRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The Receive FIFO Full interrupt is sent to the interrupt controller when the RXFFRIS bit in the **I2CMRIS** register is set. |
| 10 | TXFEIM | RW | 0 | Transmit FIFO Empty Interrupt Mask<br><br>**Note:** The TXFEIM interrupt mask bit in the **I2CMIMR** register should be clear (masking the TXFE interrupt) when the master is performing an RX Burst from the RXFIFO and should be unmasked before starting a TX FIFO transfers.<br><br>Value Description<br>0 The TXFERIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The Transmit FIFO Empty interrupt is sent to the interrupt controller when the TXFERIS bit in the **I2CMRIS** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RXIM | RW | 0 | Receive FIFO Request Interrupt Mask<br>Value Description<br>0 The RXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The RX FIFO Request interrupt is sent to the interrupt controller when the RXRIS bit in the **I2CMRIS** register is set. |
| 8 | TXIM | RW | 0 | Transmit FIFO Request Interrupt Mask<br>Value Description<br>0 The TXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The TX FIFO Request interrupt is sent to the interrupt controller when the TXRIS bit in the **I2CMRIS** register is set. |
| 7 | ARBLOSTIM | RW | 0 | Arbitration Lost Interrupt Mask<br>Value Description<br>0 The ARBLOSTRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The Arbitration Lost interrupt is sent to the interrupt controller when the ARBLOSTRIS bit in the **I2CMRIS** register is set. |
| 6 | STOPIM | RW | 0 | STOP Detection Interrupt Mask<br>Value Description<br>0 The STOPRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The STOP detection interrupt is sent to the interrupt controller when the STOPRIS bit in the **I2CMRIS** register is set. |
| 5 | STARTIM | RW | 0 | START Detection Interrupt Mask<br>Value Description<br>0 The STARTRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The START detection interrupt is sent to the interrupt controller when the STARTRIS bit in the **I2CMRIS** register is set. |
| 4 | NACKIM | RW | 0 | Address/Data NACK Interrupt Mask<br>Value Description<br>0 The NACKRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The address/data NACK interrupt is sent to the interrupt controller when the NACKRIS bit in the **I2CMRIS** register is set. |

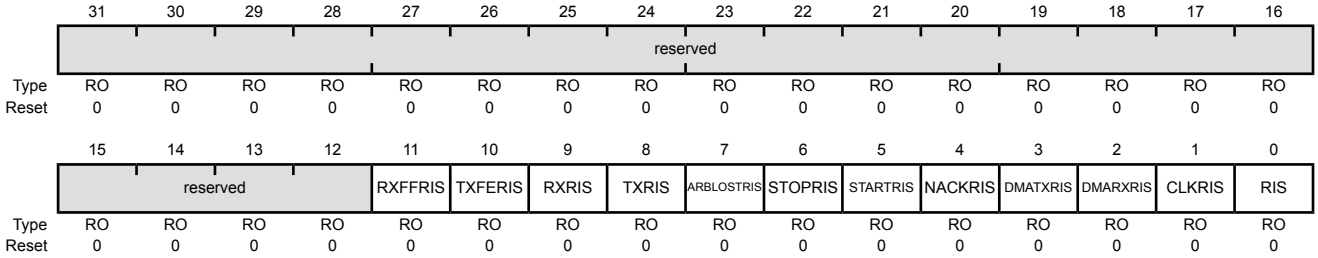
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | DMATXIM | RW | 0 | Transmit DMA Interrupt Mask<br>Value Description<br>0 The DMATXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The transmit DMA complete interrupt is sent to the interrupt controller when the DMATXRIS bit in the **I2CMRIS** register is set. |
| 2 | DMARXIM | RW | 0 | Receive DMA Interrupt Mask<br>Value Description<br>0 The DMARXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The receive DMA complete interrupt is sent to the interrupt controller when the DMARXRIS bit in the **I2CMRIS** register is set. |
| 1 | CLKIM | RW | 0 | Clock Timeout Interrupt Mask<br>Value Description<br>0 The CLKRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The clock timeout interrupt is sent to the interrupt controller when the CLKRIS bit in the **I2CMRIS** register is set. |
| 0 | IM | RW | 0 | Master Interrupt Mask<br>Value Description<br>0 The RIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The master interrupt is sent to the interrupt controller when the RIS bit in the **I2CMRIS** register is set. |

## Register 6: I²C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

I2C Master Raw Interrupt Status (I2CMRIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x014
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0 | 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. |
| 11 | RXFFRIS | RO | 0 | Receive FIFO Full Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The Receive FIFO Full interrupt is pending.<br><br>This bit is cleared by writing a 1 to the RXFFIC bit in the **I2CMICR** register. |
| 10 | TXFERIS | RO | 0 | Transmit FIFO Empty Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The Transmit FIFO Empty interrupt is pending.<br><br>This bit is cleared by writing a 1 to the TXFEIC bit in the **I2CMICR** register.<br><br>Note that if we clear the TXFERIS interrupt (by setting the TXFEIC bit) when the TX FIFO is empty, the TXFERIS interrupt does not reassert even though the TX FIFO remains empty in this situation. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RXRIS | RO | 0 | Receive FIFO Request Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The trigger level for the RX FIFO has been reached or there is data in the FIFO and the burst count is zero. Thus, a RX FIFO request interrupt is pending.<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **I2CMICR** register. |
| 8 | TXRIS | RO | 0 | Transmit Request Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The trigger level for the TX FIFO has been reached and more data is needed to complete the burst. Thus, a TX FIFO request interrupt is pending.<br><br>This bit is cleared by writing a 1 to the TXIC bit in the **I2CMICR** register. |
| 7 | ARBLOSTRIS | RO | 0 | Arbitration Lost Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The Arbitration Lost interrupt is pending.<br><br>This bit is cleared by writing a 1 to the ARBLOSTIC bit in the **I2CMICR** register. |
| 6 | STOPRIS | RO | 0 | STOP Detection Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The STOP Detection interrupt is pending.<br><br>This bit is cleared by writing a 1 to the STOPIC bit in the **I2CMICR** register. |
| 5 | STARTRIS | RO | 0 | START Detection Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The START Detection interrupt is pending.<br><br>This bit is cleared by writing a 1 to the STARTIC bit in the **I2CMICR** register. |
| 4 | NACKRIS | RO | 0 | Address/Data NACK Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The address/data NACK interrupt is pending.<br><br>This bit is cleared by writing a 1 to the NACKIC bit in the **I2CMICR** register. |

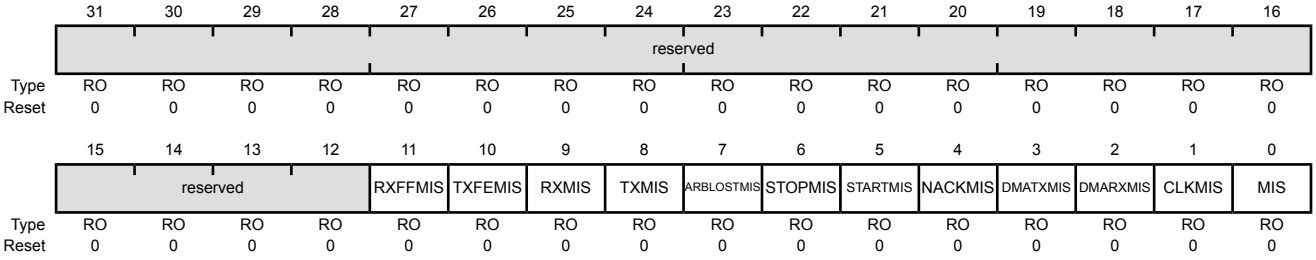
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | DMATXRIS | RO | 0 | Transmit DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The transmit DMA complete interrupt is pending.<br><br>This bit is cleared by writing a 1 to the DMATXIC bit in the **I2CMICR** register. |
| 2 | DMARXRIS | RO | 0 | Receive DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The receive DMA complete interrupt is pending.<br><br>This bit is cleared by writing a 1 to the DMARXIC bit in the **I2CMICR** register. |
| 1 | CLKRIS | RO | 0 | Clock Timeout Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 The clock timeout interrupt is pending.<br><br>This bit is cleared by writing a 1 to the CLKIC bit in the **I2CMICR** register. |
| 0 | RIS | RO | 0 | Master Raw Interrupt Status<br>This interrupt includes:<br>■ Master transaction completed<br>■ Next byte transfer request<br><br>Value Description<br>0 No interrupt.<br>1 A master interrupt is pending.<br><br>This bit is cleared by writing a 1 to the IC bit in the **I2CMICR** register. |

## Register 7: I²C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

I2C Master Masked Interrupt Status (I2CMMIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x018
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0 | 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. |
| 11 | RXFFMIS | RO | 0 | Receive FIFO Full Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Receive FIFO Full interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the RXFFIC bit in the **I2CMICR** register. |
| 10 | TXFEMIS | RO | 0 | Transmit FIFO Empty Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Transmit FIFO Empty interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the TXFEIC bit in the **I2CMICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RXMIS | RO | 0 | Receive FIFO Request Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Receive FIFO Request interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **I2CMICR** register. |
| 8 | TXMIS | RO | 0 | Transmit Request Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Transmit FIFO Request interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the TXIC bit in the **I2CMICR** register. |
| 7 | ARBLOSTMIS | RO | 0 | Arbitration Lost Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Arbitration Lost interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the ARBLOSTIC bit in the **I2CMICR** register. |
| 6 | STOPMIS | RO | 0 | STOP Detection Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked STOP Detection interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the STOPIC bit in the **I2CMICR** register. |
| 5 | STARTMIS | RO | 0 | START Detection Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked START Detection interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the STARTIC bit in the **I2CMICR** register. |

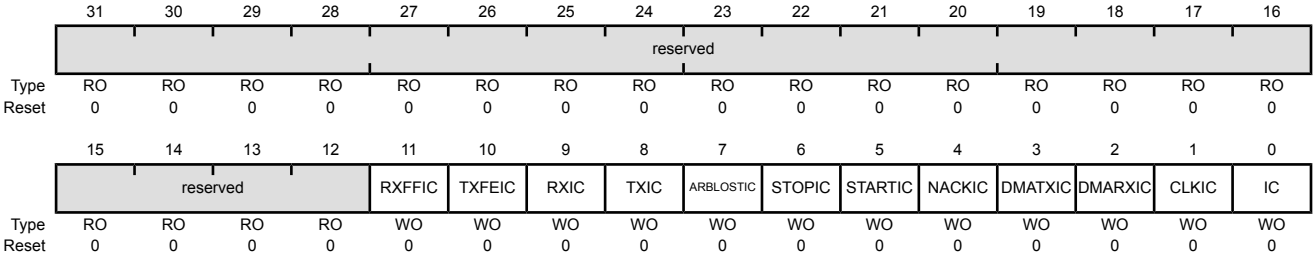
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | NACKMIS | RO | 0 | Address/Data NACK Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Address/Data NACK interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the NACKIC bit in the **I2CMICR** register. |
| 3 | DMATXMIS | RO | 0 | Transmit DMA Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked transmit DMA complete interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the DMATXIC bit in the **I2CMICR** register. |
| 2 | DMARXMIS | RO | 0 | Receive DMA Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked receive DMA complete interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the DMARXIC bit in the **I2CMICR** register. |
| 1 | CLKMIS | RO | 0 | Clock Timeout Masked Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked clock timeout interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the CLKIC bit in the **I2CMICR** register. |
| 0 | MIS | RO | 0 | Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked master interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the IC bit in the **I2CMICR** register. |

## Register 8: I2C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw and masked interrupts.

I2C Master Interrupt Clear (I2CMICR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x01C
Type WO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0 | 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. |
| 11 | RXFFIC | WO | 0 | Receive FIFO Full Interrupt Clear<br>Writing a 1 to this bit clears the RXFFIS bit in the **I2CMRIS** register and the RXFFMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 10 | TXFEIC | WO | 0 | Transmit FIFO Empty Interrupt Clear<br>Writing a 1 to this bit clears the TXFERIS bit in the **I2CMRIS** register and the TXFEMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 9 | RXIC | WO | 0 | Receive FIFO Request Interrupt Clear<br>Writing a 1 to this bit clears the RXRIS bit in the **I2CMRIS** register and the RXMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 8 | TXIC | WO | 0 | Transmit FIFO Request Interrupt Clear<br>Writing a 1 to this bit clears the TXRIS bit in the **I2CMRIS** register and the TXMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 7 | ARBLOSTIC | WO | 0 | Arbitration Lost Interrupt Clear<br>Writing a 1 to this bit clears the ARBLOSTRIS bit in the **I2CMRIS** register and the ARBLOSTMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |

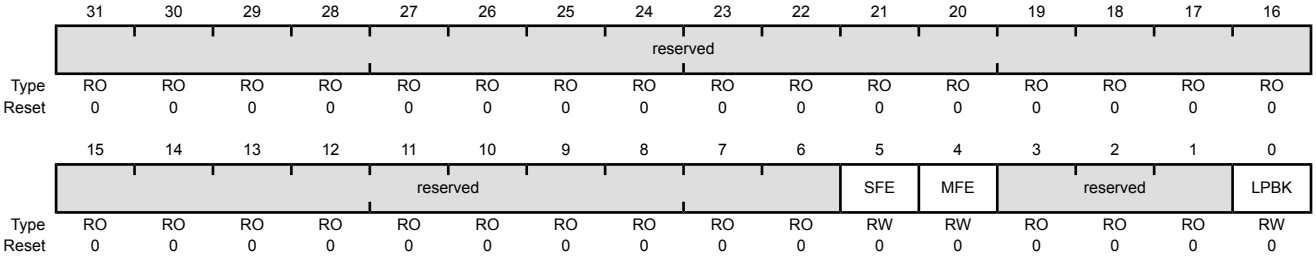
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | STOPIC | WO | 0 | STOP Detection Interrupt Clear<br>Writing a 1 to this bit clears the STOPRIS bit in the **I2CMRIS** register and the STOPMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 5 | STARTIC | WO | 0 | START Detection Interrupt Clear<br>Writing a 1 to this bit clears the STARTRIS bit in the **I2CMRIS** register and the STARTMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 4 | NACKIC | WO | 0 | Address/Data NACK Interrupt Clear<br>Writing a 1 to this bit clears the NACKRIS bit in the **I2CMRIS** register and the NACKMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 3 | DMATXIC | WO | 0 | Transmit DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMATXRIS bit in the **I2CMRIS** register and the DMATXMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 2 | DMARXIC | WO | 0 | Receive DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMARXRIS bit in the **I2CMRIS** register and the DMARXMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 1 | CLKIC | WO | 0 | Clock Timeout Interrupt Clear<br>Writing a 1 to this bit clears the CLKRIS bit in the **I2CMRIS** register and the CLKMIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |
| 0 | IC | WO | 0 | Master Interrupt Clear<br>Writing a 1 to this bit clears the RIS bit in the **I2CMRIS** register and the MIS bit in the **I2CMMIS** register.<br>A read of this register returns no meaningful data. |

## Register 9: I²C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave), and sets the interface for test mode loopback.

I2C Master Configuration (I2CMCR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x020
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.00 | 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 | SFE | RW | 0 | I²C Slave Function Enable<br>Value Description<br>0 Slave mode is disabled.<br>1 Slave mode is enabled. |
| 4 | MFE | RW | 0 | I²C Master Function Enable<br>Value Description<br>0 Master mode is disabled.<br>1 Master mode is enabled. |
| 3:1 | reserved | RO | 0x0 | 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. |
| 0 | LPBK | RW | 0 | I²C Loopback<br>Value Description<br>0 Normal operation.<br>1 The controller in a test mode loopback configuration. |

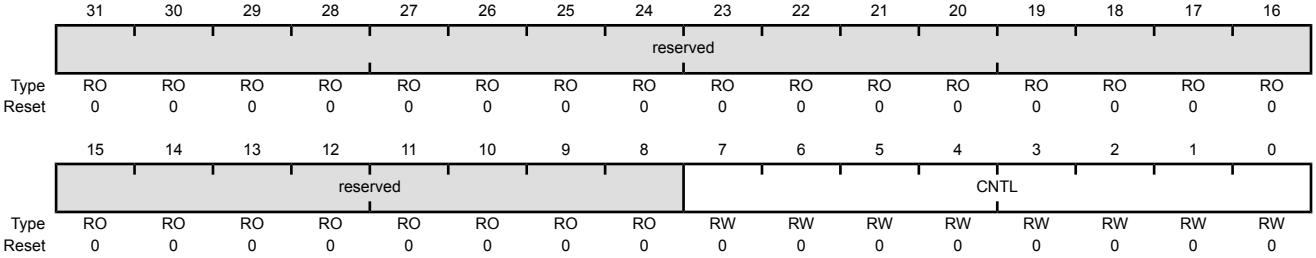
## Register 10: I²C Master Clock Low Timeout Count (I2CMCLKOCNT), offset 0x024

This register contains the upper 8 bits of a 12-bit counter that can be used to keep the timeout limit for clock stretching by a remote slave. The lower four bits of the counter are not user visible and are always 0x0.

**Note:** The Master Clock Low Timeout counter counts for the entire time SCL is held Low continuously. If SCL is deasserted at any point, the Master Clock Low Timeout Counter is reloaded with the value in the **I2CMCLKOCNT** register and begins counting down from this value.

I2C Master Clock Low Timeout Count (I2CMCLKOCNT)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x024
Type RW, reset 0x0000.0000



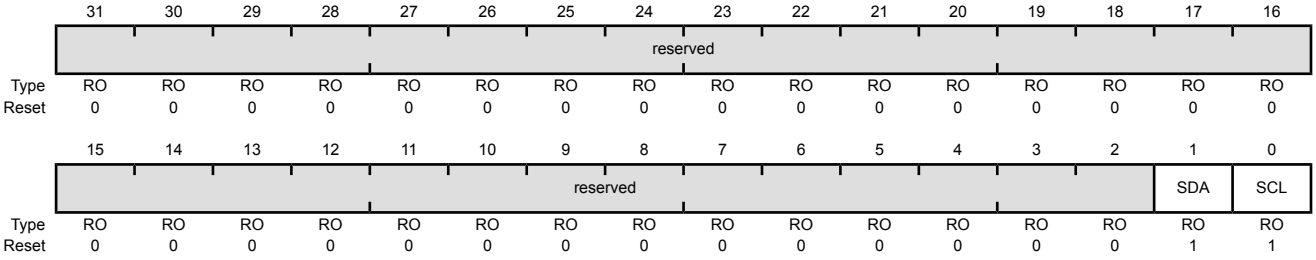

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CNTL | RW | 0 | I²C Master Count<br>This field contains the upper 8 bits of a 12-bit counter for the clock low timeout count.<br>**Note:** The value of CNTL must be greater than 0x1. |

## Register 11: I²C Master Bus Monitor (I2CMBMON), offset 0x02C

This register is used to determine the SCL and SDA signal status.

I2C Master Bus Monitor (I2CMBMON)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x02C
Type RO, reset 0x0000.0003



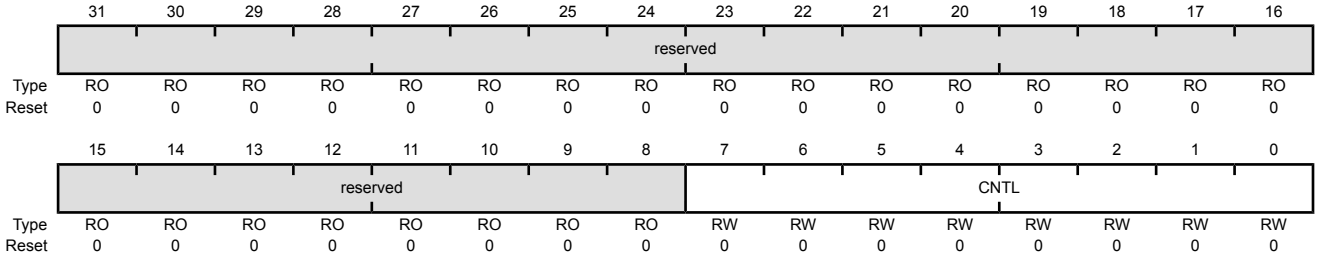

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | SDA | RO | 1 | I²C SDA Status<br>Value Description<br>0 The I2CSDA signal is low.<br>1 The I2CSDA signal is high. |
| 0 | SCL | RO | 1 | I²C SCL Status<br>Value Description<br>0 The I2CSCL signal is low.<br>1 The I2CSCL signal is high. |

## Register 12: I²C Master Burst Length (I2CMBLEN), offset 0x030

This register contains the programmed length of bytes that are transferred during a Burst request.

I2C Master Burst Length (I2CMBLEN)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x030
Type RW, reset 0x0000.0000



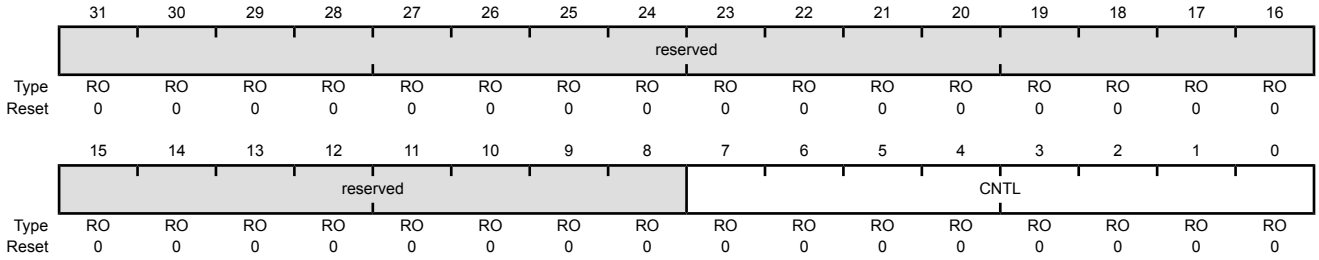

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CNTL | RW | 0 | I²C Burst Length<br>This field contains the programmed length of bytes of the Burst Transaction. If BURST is enabled this register must be set to a non-zero value otherwise an error will occur. |

### Register 13: I2C Master Burst Count (I2CMBCNT), offset 0x034

When BURST is active, the value in the **I2CMBLEN** register is copied into this register and decremented during the BURST transaction. This register can be used to determine the number of transfers that occurred when a BURST terminates early (as a result of a data NACK). When a BURST completes successfully, this register will contain 0.

I2C Master Burst Count (I2CMBCNT)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x034
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | CNTL | RO | 0 | I2C Master Burst Count<br>This field contains the current count-down value of the BURST transaction. |

## 21.7 Register Descriptions (I2C Slave)

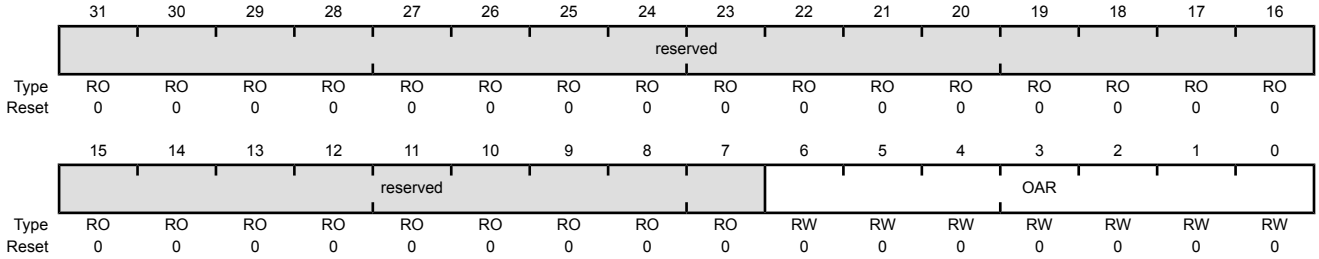
The remainder of this section lists and describes the I2C slave registers, in numerical order by address offset.

## Register 14: I²C Slave Own Address (I2CSOAR), offset 0x800

This register consists of seven address bits that identify the TM4C129XKCZAD I²C device on the I²C bus.

I2C Slave Own Address (I2CSOAR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x800
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0x0000.00 | 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. |
| 6:0 | OAR | RW | 0x00 | I²C Slave Own Address<br>This field specifies bits A6 through A0 of the slave address. |

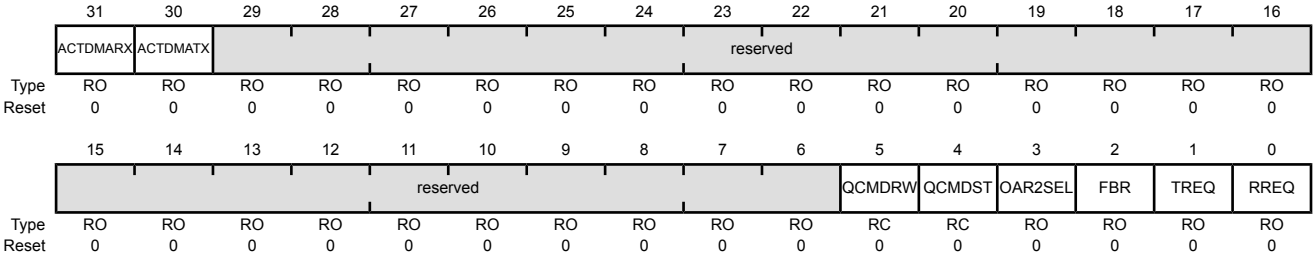
## Register 15: I2C Slave Control/Status (I2CSCSR), offset 0x804

This register functions as a control register when written, and a status register when read.

### Read-Only Status Register

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x804
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ACTDMARX | RO | 0 | DMA RX Active Status<br><br>Value Description<br>0 DMA RX is not active<br>1 DMA RX is active. |
| 30 | ACTDMATX | RO | 0 | DMA TX Active Status<br><br>Value Description<br>0 DMA TX is not active<br>1 DMA TX is active. |
| 29:6 | reserved | RO | 0x0000.000 | 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 | QCMDRW | RC | 0 | Quick Command Read / Write<br><br>Value Description<br>0 Quick command was a write<br>1 Quick command was a read<br><br>This bit only has meaning when the QCMDST bit is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | QCMDST | RC | 0 | Quick Command Status<br><br>Value Description<br>0 The last transaction was a normal transaction or a transaction has not occurred.<br>1 The last transaction was a Quick Command transaction. |
| 3 | OAR2SEL | RO | 0 | OAR2 Address Matched<br><br>Value Description<br>0 Either the address is not matched or the match is in legacy mode.<br>1 OAR2 address matched and ACKed by the slave.<br><br>This bit gets reevaluated after every address comparison. |
| 2 | FBR | RO | 0 | First Byte Received<br><br>Value Description<br>0 The first byte has not been received.<br>1 The first byte following the slave's own address has been received.<br><br>This bit is only valid when the RREQ bit is set and is automatically cleared when data has been read from the **I2CSDR** register.<br><br>**Note:** This bit is not used for slave transmit operations. |
| 1 | TREQ | RO | 0 | Transmit Request<br><br>Value Description<br>0 No outstanding transmit request.<br>1 The $I^2C$ controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the **I2CSDR** register. |
| 0 | RREQ | RO | 0 | Receive Request<br><br>Value Description<br>0 No outstanding receive data.<br>1 The $I^2C$ controller has outstanding receive data from the $I^2C$ master and is using clock stretching to delay the master until the data has been read from the **I2CSDR** register. |

## Write-Only Control Register

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x804
Type WO, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | RXFIFO | TXFIFO | DA |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | WO | WO | WO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0 | 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. |
| 2 | RXFIFO | WO | 0 | RX FIFO Enable<br><br>Value Description<br>0 Disables RX FIFO<br>1 Enables RX FIFO |
| 1 | TXFIFO | WO | 0 | TX FIFO Enable<br><br>Value Description<br>0 Disables TX FIFO<br>1 Enables TX FIFO |
| 0 | DA | WO | 0 | Device Active<br><br>Value Description<br>0 Disables the I²C slave operation.<br>1 Enables the I²C slave operation.<br><br>Once this bit has been set, it should not be set again unless it has been cleared by writing a 0 or by a reset, otherwise transfer failures may occur. |

## Register 16: I²C Slave Data (I2CSDR), offset 0x808

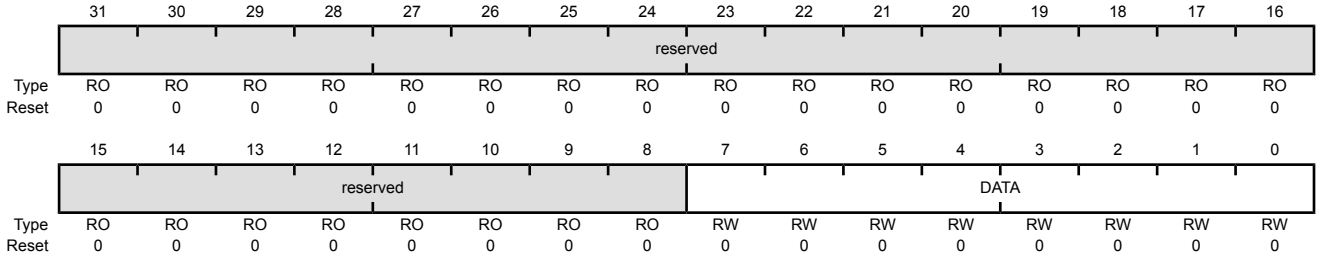
**Important:** This register is read-sensitive. See the register description for details.

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state. If the RXFIFO bit or TXFIFO bit are enabled in the **I2CSCSR** register, then this register is ignored and the data value being transferred from the FIFO is contained in the **I2CFIFODATA** register.

**Note:** Best practice recommends that an application should not switch between the **I2CSDR** register and TX FIFO or vice versa for successive transactions.

I2C Slave Data (I2CSDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x808
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7:0 | DATA | RW | 0x00 | Data for Transfer<br>This field contains the data for transfer during a slave receive or transmit operation. |

## Register 17: I²C Slave Interrupt Mask (I2CSIMR), offset 0x80C

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Slave Interrupt Mask (I2CSIMR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x80C
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | RXFFIM | TXFEIM | RXIM | TXIM | DMATXIM | DMARXIM | STOPIM | STARTIM | DATAIM |
| Type | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | 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. |
| 8 | RXFFIM | RW | 0 | Receive FIFO Full Interrupt Mask<br>Value Description<br>0 The RXFFRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The Receive FIFO Full interrupt is sent to the interrupt controller when the RXFFRIS bit in the **I2CSRIS** register is set. |
| 7 | TXFEIM | RW | 0 | Transmit FIFO Empty Interrupt Mask<br>Value Description<br>0 The TXFERIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The Transmit FIFO Empty interrupt is sent to the interrupt controller when the TXFERIS bit in the **I2CSRIS** register is set. |
| 6 | RXIM | RW | 0 | Receive FIFO Request Interrupt Mask<br>Value Description<br>0 The RXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The RX FIFO Request interrupt is sent to the interrupt controller when the RXRIS bit in the **I2CSRIS** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TXIM | RW | 0 | Transmit FIFO Request Interrupt Mask<br>Value Description<br>0 The TXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The TX FIFO Request interrupt is sent to the interrupt controller when the TXRIS bit in the **I2CSRIS** register is set. |
| 4 | DMATXIM | RW | 0 | Transmit DMA Interrupt Mask<br>Value Description<br>0 The DMATXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The transmit DMA complete interrupt is sent to the interrupt controller when the DMATXRIS bit in the **I2CSRIS** register is set. |
| 3 | DMARXIM | RW | 0 | Receive DMA Interrupt Mask<br>Value Description<br>0 The DMARXRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The receive DMA complete interrupt is sent to the interrupt controller when the DMARXRIS bit in the **I2CSRIS** register is set. |
| 2 | STOPIM | RW | 0 | Stop Condition Interrupt Mask<br>Value Description<br>0 The STOPRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The STOP condition interrupt is sent to the interrupt controller when the STOPRIS bit in the **I2CSRIS** register is set. |
| 1 | STARTIM | RW | 0 | Start Condition Interrupt Mask<br>Value Description<br>0 The STARTRIS interrupt is suppressed and not sent to the interrupt controller.<br>1 The START condition interrupt is sent to the interrupt controller when the STARTRIS bit in the **I2CSRIS** register is set. |
| 0 | DATAIM | RW | 0 | Data Interrupt Mask<br>Value Description<br>0 The DATARIS interrupt is suppressed and not sent to the interrupt controller.<br>1 Data interrupt sent to interrupt controller when DATARIS bit in the **I2CSRIS** register is set. |

## Register 18: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x810

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x810
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | 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. |
| 8 | RXFFRIS | RO | 0 | Receive FIFO Full Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The Receive FIFO Full interrupt is pending.<br><br>This bit is cleared by writing a 1 to the RXFFIC bit in the **I2CSICR** register. |
| 7 | TXFERIS | RO | 0 | Transmit FIFO Empty Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The Transmit FIFO Empty interrupt is pending.<br><br>This bit is cleared by writing a 1 to the TXFEIC bit in the **I2CSICR** register.<br><br>Note that if the TXFERIS interrupt is cleared (by setting the TXFEIC bit) when the TX FIFO is empty, the TXFERIS interrupt does not reassert even though the TX FIFO remains empty in this situation. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | RXRIS | RO | 0 | Receive FIFO Request Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The trigger value for the FIFO has been reached and a RX FIFO Request interrupt is pending.<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **I2CSICR** register. |
| 5 | TXRIS | RO | 0 | Transmit Request Raw Interrupt Status<br><br>Value Description<br>0 No interrupt<br>1 The trigger value for the FIFO has been reached and a TX FIFO Request interrupt is pending.<br><br>This bit is cleared by writing a 1 to the TXIC bit in the **I2CSICR** register. |
| 4 | DMATXRIS | RO | 0 | Transmit DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 A transmit DMA complete interrupt is pending.<br><br>This bit is cleared by writing a 1 to the DMATXIC bit in the **I2CSICR** register. |
| 3 | DMARXRIS | RO | 0 | Receive DMA Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 A receive DMA complete interrupt is pending.<br><br>This bit is cleared by writing a 1 to the DMARXIC bit in the **I2CSICR** register. |
| 2 | STOPRIS | RO | 0 | Stop Condition Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 A STOP condition interrupt is pending.<br><br>This bit is cleared by writing a 1 to the STOPIC bit in the **I2CSICR** register. |
| 1 | STARTRIS | RO | 0 | Start Condition Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 A START condition interrupt is pending.<br><br>This bit is cleared by writing a 1 to the STARTIC bit in the **I2CSICR** register. |

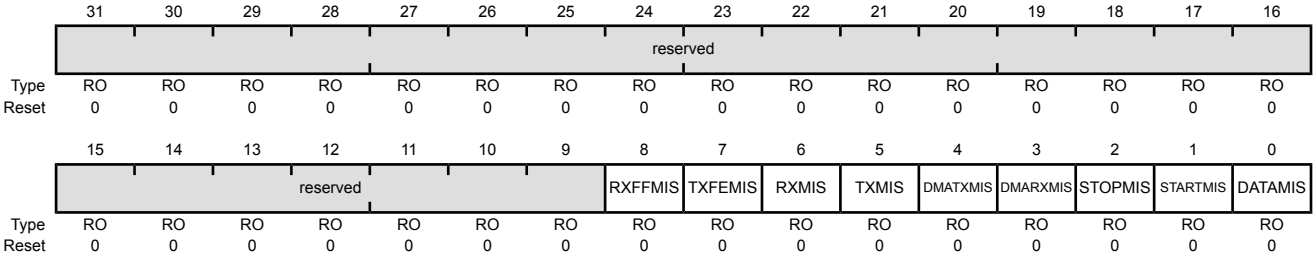
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | DATARIS | RO | 0 | Data Raw Interrupt Status<br>This interrupt encompasses the following:<br>■ Slave transaction received<br>■ Slave transaction requested<br>■ Next byte transfer request<br><br>Value Description<br>0 No interrupt.<br>1 Slave Interrupt is pending.<br><br>This bit is cleared by writing a 1 to the DATAIC bit in the **I2CSICR** register. |

## Register 19: I²C Slave Masked Interrupt Status (I2CSMIS), offset 0x814

This register specifies whether an interrupt was signaled.

I2C Slave Masked Interrupt Status (I2CSMIS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x814
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | 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. |
| 8 | RXFFMIS | RO | 0 | Receive FIFO Full Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Receive FIFO Full interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the RXFFIC bit in the **I2CSICR** register. |
| 7 | TXFEMIS | RO | 0 | Transmit FIFO Empty Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Transmit FIFO Empty interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the TXFEIC bit in the **I2CSICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | RXMIS | RO | 0 | Receive FIFO Request Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Receive FIFO Request interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **I2CSICR** register. |
| 5 | TXMIS | RO | 0 | Transmit FIFO Request Interrupt Mask<br><br>Value Description<br>0 No interrupt.<br>1 An unmasked Transmit FIFO Request interrupt was signaled and is pending.<br><br>This bit is cleared by writing a 1 to the TXIC bit in the **I2CSICR** register. |
| 4 | DMATXMIS | RO | 0 | Transmit DMA Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked transmit DMA complete interrupt was signaled is pending.<br><br>This bit is cleared by writing a 1 to the DMATXIC bit in the **I2CSICR** register. |
| 3 | DMARXMIS | RO | 0 | Receive DMA Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked receive DMA complete interrupt was signaled is pending.<br><br>This bit is cleared by writing a 1 to the DMARXIC bit in the **I2CSICR** register. |
| 2 | STOPMIS | RO | 0 | Stop Condition Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked STOP condition interrupt was signaled is pending.<br><br>This bit is cleared by writing a 1 to the STOPIC bit in the **I2CSICR** register. |

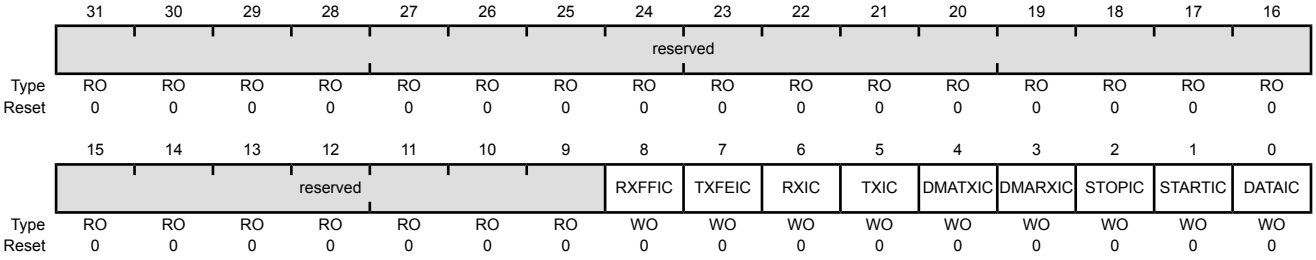
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | STARTMIS | RO | 0 | Start Condition Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked START condition interrupt was signaled is pending.<br><br>This bit is cleared by writing a 1 to the STARTIC bit in the **I2CSICR** register. |
| 0 | DATAMIS | RO | 0 | Data Masked Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred or is masked.<br>1 An unmasked slave data interrupt was signaled is pending.<br><br>This bit is cleared by writing a 1 to the DATAIC bit in the **I2CSICR** register. |

## Register 20: I²C Slave Interrupt Clear (I2CSICR), offset 0x818

This register clears the raw interrupt. A read of this register returns no meaningful data.

I2C Slave Interrupt Clear (I2CSICR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x818
Type WO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0 | 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. |
| 8 | RXFFIC | WO | 0 | Receive FIFO Full Interrupt Mask<br>Writing a 1 to this bit clears the RXFFIS bit in the **I2CSRIS** register and the RXFFMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |
| 7 | TXFEIC | WO | 0 | Transmit FIFO Empty Interrupt Mask<br>Writing a 1 to this bit clears the TXFERIS bit in the **I2CSRIS** register and the TXFEMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |
| 6 | RXIC | WO | 0 | Receive Request Interrupt Mask<br>Writing a 1 to this bit clears the RXRIS bit in the **I2CSRIS** register and the RXMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |
| 5 | TXIC | WO | 0 | Transmit Request Interrupt Mask<br>Writing a 1 to this bit clears the TXRIS bit in the **I2CSRIS** register and the TXMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |
| 4 | DMATXIC | WO | 0 | Transmit DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMATXRIS bit in the **I2CSRIS** register and the DMATXMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |

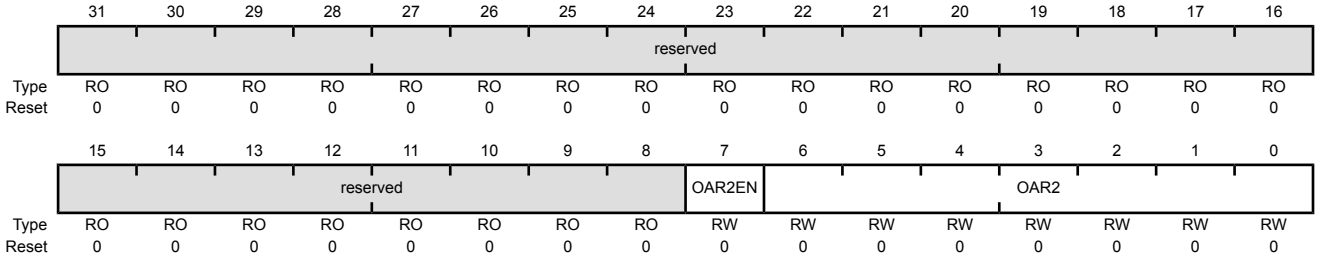
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | DMARXIC | WO | 0 | Receive DMA Interrupt Clear<br>Writing a 1 to this bit clears the DMARXRIS bit in the **I2CSRIS** register and the DMARXMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |
| 2 | STOPIC | WO | 0 | Stop Condition Interrupt Clear<br>Writing a 1 to this bit clears the STOPRIS bit in the **I2CSRIS** register and the STOPMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |
| 1 | STARTIC | WO | 0 | Start Condition Interrupt Clear<br>Writing a 1 to this bit clears the STARTRIS bit in the **I2CSRIS** register and the STARTMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |
| 0 | DATAIC | WO | 0 | Data Interrupt Clear<br>Writing a 1 to this bit clears the DATARIS bit in the **I2CSRIS** register and the DATMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |

## Register 21: I²C Slave Own Address 2 (I2CSOAR2), offset 0x81C

This register consists of seven address bits that identify the alternate address for the I²C device on the I²C bus.

I2C Slave Own Address 2 (I2CSOAR2)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x81C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | OAR2EN | RW | 0 | I²C Slave Own Address 2 Enable<br>Value Description<br>0 The alternate address is disabled.<br>1 Enables the use of the alternate address in the OAR2 field. |
| 6:0 | OAR2 | RW | 0x00 | I²C Slave Own Address 2<br>This field specifies the alternate OAR2 address. |

### Register 22: I²C Slave ACK Control (I2CSACKCTL), offset 0x820

This register enables the I²C slave to NACK for invalid data or command or ACK for valid data or command. The I²C clock is pulled low after the last data bit until this register is written.

I2C Slave ACK Control (I2CSACKCTL)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0x820
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | ACKOVAL | ACKOEN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | 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. |
| 1 | ACKOVAL | RW | 0 | I²C Slave ACK Override Value<br>Value Description<br>0 An ACK is sent indicating valid data or command.<br>1 A NACK is sent indicating invalid data or command. |
| 0 | ACKOEN | RW | 0 | I²C Slave ACK Override Enable<br>Value Description<br>0 A response in not provided.<br>1 An ACK or NACK is sent according to the value written to the ACK0VAL bit. |

## 21.8 Register Descriptions (I²C Status and Control)

The remainder of this section lists and describes the I²C status and control registers, in numerical order by address offset.

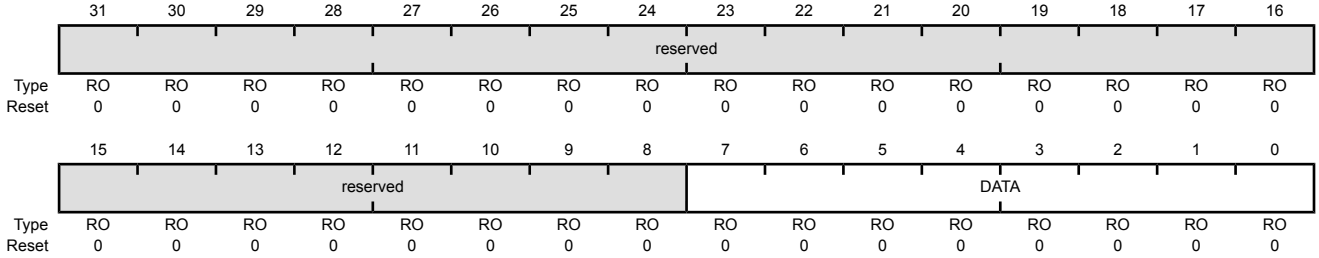
## Register 23: I²C FIFO Data (I2CFIFODATA), offset 0xF00

The **I²C FIFO Data (I2CFIFODATA)** register contains the current value of the top of the RX or TX FIFO stack being used in the a transfer.

### Read-Only Status Register

I2C FIFO Data (I2CFIFODATA)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0xF00
Type RO, reset 0x0000.0000

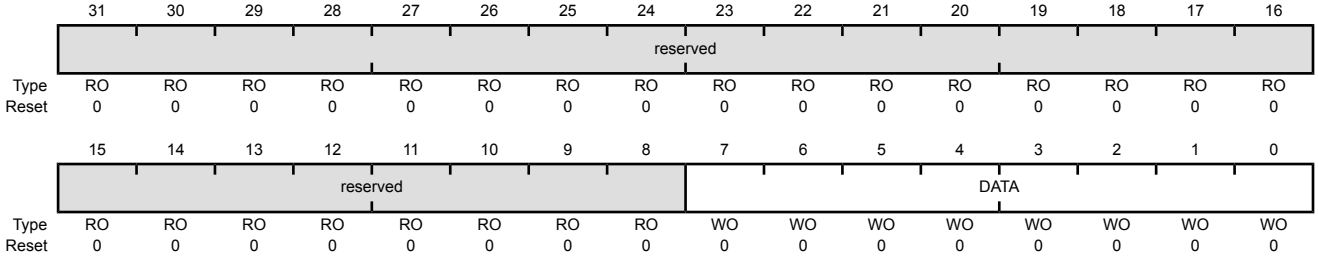



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | RO | 0 | I²C RX FIFO Read Data Byte<br>This field contains the current byte being read in the RX FIFO stack. |

## Write-Only Control Register

### I2C FIFO Data (I2CFIFODATA)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0xF00
Type WO, reset 0x0000.0000



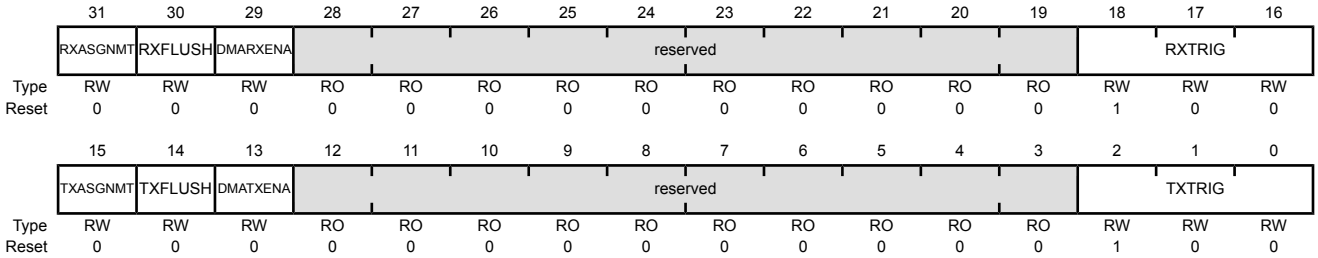

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | WO | 0 | $I^2C$ TX FIFO Write Data Byte<br>This field contains the current byte written to the TX FIFO.<br>For back to back transmit operations, the application should not switch between writing to the **I2CSDR** register and the **I2CFIFODATA**. |

## Register 24: I2C FIFO Control (I2CFIFOCTL), offset 0xF04

The FIFO Control Register can be programmed to control various aspects of the FIFO transaction, such as RX and TX FIFO assignment, byte count value for FIFO triggers and flushing of the FIFOs.

I2C FIFO Control (I2CFIFOCTL)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0xF04
Type RW, reset 0x0004.0004




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RXASGNMT | RW | 0 | RX Control Assignment<br><br>Value Description<br>0 RX FIFO is assigned to Master<br>1 RX FIFO is assigned to Slave |
| 30 | RXFLUSH | RW | 0 | RX FIFO Flush<br>Setting this bit will Flush the RX FIFO. This bit will self-clear when the flush has completed. |
| 29 | DMARXENA | RW | 0 | DMA RX Channel Enable<br><br>Value Description<br>0 DMA RX channel disabled<br>1 DMA RX channel enabled |
| 28:19 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 18:16 | RXTRIG | RW | 0x4 | RX FIFO Trigger<br>Indicates at what fill level the RX FIFO will generate a trigger.<br>**Note:** Programming RXTRIG to 0x0 has no effect since no data is present to transfer out of RX FIFO.<br>Value Description<br>0x0 Trigger when RX FIFO contains no bytes<br>0x1 Trigger when Rx FIFO contains 1 or more bytes<br>0x2 Trigger when Rx FIFO contains 2 or more bytes<br>0x3 Trigger when Rx FIFO contains 3 or more bytes<br>0x4 Trigger when Rx FIFO contains 4 or more bytes<br>0x5 Trigger when Rx FIFO contains 5 or more bytes<br>0x6 Trigger when Rx FIFO contains 6 or more bytes<br>0x7 Trigger when Rx FIFO contains 7 or more bytes. |
| 15 | TXASGNMT | RW | 0 | TX Control Assignment<br>Value Description<br>0 TX FIFO is assigned to Master<br>1 TX FIFO is assigned to Slave |
| 14 | TXFLUSH | RW | 0 | TX FIFO Flush<br>Setting this bit will Flush the TX FIFO. This bit will self-clear when the flush has completed. |
| 13 | DMATXENA | RW | 0 | DMA TX Channel Enable<br>Value Description<br>0 DMA TX channel disabled<br>1 DMA TX channel enabled |
| 12:3 | reserved | RO | 0x000 | 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. |
| 2:0 | TXTRIG | RW | 0x4 | TX FIFO Trigger<br>Indicates at what fill level in the TX FIFO a trigger will be generated.<br>Value Description<br>0x0 Trigger when the TX FIFO is empty.<br>0x1 Trigger when TX FIFO contains ≤ 1 byte<br>0x2 Trigger when TX FIFO contains ≤ 2 bytes<br>0x3 Trigger when TX FIFO ≤ 3 bytes<br>0x4 Trigger when FIFO ≤ 4 bytes<br>0x5 Trigger when FIFO ≤ 5 bytes<br>0x6 Trigger when FIFO ≤ 6 bytes<br>0x7 Trigger when FIFO ≤ 7 bytes |

## Register 25: I²C FIFO Status (I2CFIFOSTATUS), offset 0xF08

This register contains the real-time status of the RX and TX FIFOs.

I2C FIFO Status (I2CFIFOSTATUS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0xF08
Type RO, reset 0x0001.0005

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | RXABVTRIG | RXFF | RXFE |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | TXBLWTRIG | TXFF | TXFE |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x000 | 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. |
| 18 | RXABVTRIG | RO | 0 | RX FIFO Above Trigger Level<br>Value Description<br>0 The number of bytes in RX FIFO is below the trigger level programmed by the RXTRIG bit in the **I2CFIFOCTL** register<br>1 The number of bytes in the RX FIFO is above the trigger level programmed by the RXTRIG bit in the **I2CFIFOCTL** register |
| 17 | RXFF | RO | 0 | RX FIFO Full<br>Value Description<br>0 The RX FIFO is not full.<br>1 The RX FIFO is full. |
| 16 | RXFE | RO | 1 | RX FIFO Empty<br>Value Description<br>0 The RX FIFO is not empty.<br>1 The RX FIFO is empty. |
| 15:3 | reserved | RO | 0x000 | 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. |

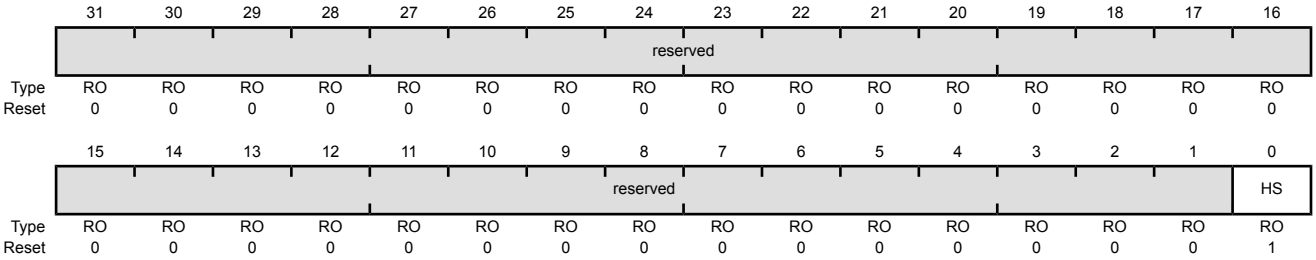
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | TXBLWTRIG | RO | 1 | TX FIFO Below Trigger Level<br><br>Value Description<br>0 The number of bytes in TX FIFO is above the trigger level programmed by the TXTRIG bit in the **I2CFIFOCTL** register<br>1 The number of bytes in the TX FIFO is below the trigger level programmed by the TXTRIG bit in the **I2CFIFOCTL** register |
| 1 | TXFF | RO | 0 | TX FIFO Full<br><br>Value Description<br>0 The TX FIFO is not full.<br>1 The TX FIFO is full. |
| 0 | TXFE | RO | 1 | TX FIFO Empty<br><br>Value Description<br>0 The TX FIFO is not empty.<br>1 The TX FIFO is empty. |

## Register 26: I2C Peripheral Properties (I2CPP), offset 0xFC0

The **I2CPP** register provides information regarding the properties of the I2C module.

I2C Peripheral Properties (I2CPP)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0xFC0
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0 | 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. |
| 0 | HS | RO | 0x1 | High-Speed Capable |

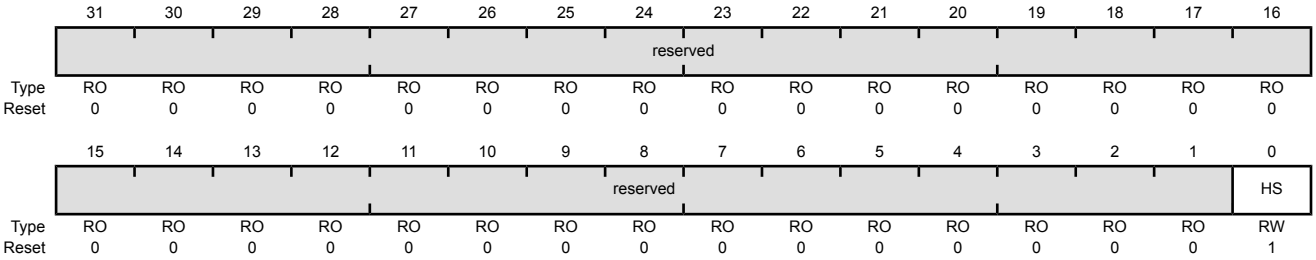
| Value | Description |
|---|---|
| 0 | The interface is capable of Standard, Fast, or Fast mode plus operation. |
| 1 | The interface is capable of High-Speed operation. |

## Register 27: I²C Peripheral Configuration (I2CPC), offset 0xFC4

The **I2CPC** register allows software to enable features present in the I²C module.

I2C Peripheral Configuration (I2CPC)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
I2C 2 base: 0x4002.2000
I2C 3 base: 0x4002.3000
I2C 4 base: 0x400C.0000
I2C 5 base: 0x400C.1000
I2C 6 base: 0x400C.2000
I2C 7 base: 0x400C.3000
I2C 8 base: 0x400B.8000
I2C 9 base: 0x400B.9000
Offset 0xFC4
Type RO, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | HS | RW | 1 | High-Speed Capable |

| Value | Description |
|---|---|
| 0 | The interface is set to Standard, Fast or Fast mode plus operation. |
| 1 | The interface is set to High-Speed operation. Note that this encoding may only be used if the HS bit in the **I2CPP** register is set. Otherwise, this encoding is not available. |

# 22 1-Wire Master Module

The 1-Wire Master module is a bi-directional serial communication interface that implements the protocol functions of the Dallas Semiconductor 1-Wire protocol and provides both power and data over a single wire. The 1-Wire Master module can interface with a multiple variety of slaves such as thermometers, mixed-signal devices, memory, and authentication devices.

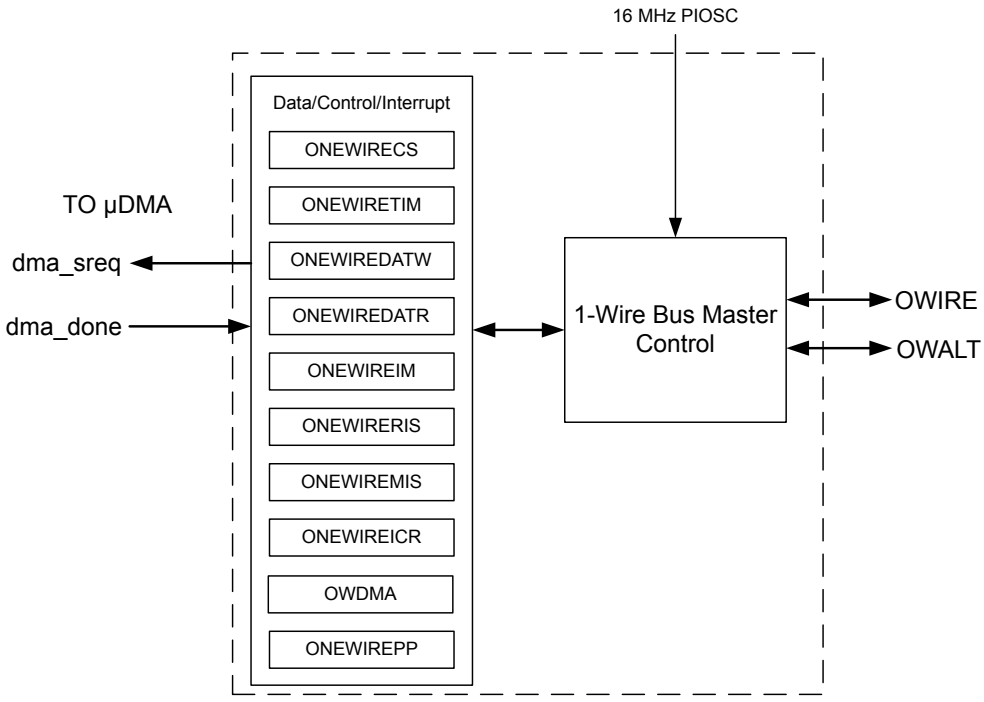
The TM4C129XKCZAD microcontroller includes a 1-Wire Master module with the following features:

- Support for standard and overdrive speeds, including a late-sample mechanism
- Allows transfers of send, receive and bi-directional bits
- Data size transfers of 1, 2, 3, or 4 bytes with sub-byte search and enumeration support
- Interrupt capability for transaction pacing and line error
- Optional two-wire support for isolated lines and high voltage use
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)

## 22.1 Block Diagram

Figure 22-1 on page 1508 details the components of the 1-wire Master Module. The clock source for the 1-Wire Master module is the 16-MHz precision internal oscillator (PIOSC).

**Figure 22-1. 1-Wire Block Diagram**



## 22.2 Signal Description

The following table lists the external signals of the 1-Wire module and describes the function of each. The 1-Wire module signals are alternate functions for GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placements for the 1-Wire signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the 1-Wire function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOCTL)** register to assign the 1-Wire signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 22-1. 1-Wire Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| 0WALT | K15<br>B12 | PG5 (5)<br>PP5 (4) | I/O | TTL | 1-Wire Optional 2nd signal to be used as output. |
| 0WIRE | G2<br>K17<br>V12<br>U14<br>D8<br>A8 | PE3 (5)<br>PG4 (5)<br>PG6 (5)<br>PG7 (5)<br>PP4 (4)<br>PP7 (5) | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |

## 22.3 Functional Description

1-Wire is a simple single-wire communication interface comprising a wire protocol, transport protocol, and base session protocols (as well as some basic commands). The wire protocol generates 0s and 1s by varying the time the line is held low (it is pulled up by a resistor). The TM4C129XKCZAD microcontroller is the master and controls the line at all times. A pull-up keeps the line parked high and the microcontroller driver is assumed to be normal open-drain. No special GPIO configuration is needed here since the open-drain support is taken care of outside of the microcontroller. See Table 10-4 on page 786 for pin configuration information. Data can be both read and written on the same 1-Wire pin. The 1-Wire module supports the most basic aspects of the protocol, including wire protocol, byte transport control and line reset. Software is expected to handle the session protocol, including selection of a slave (when more than one is on the same line) and higher level commands.

**Note:** All time values by the 1-Wire Master are stated in µs, but the actual time on the wire may be less by 62.5 ns.

### 22.3.1 1-Wire Protocol

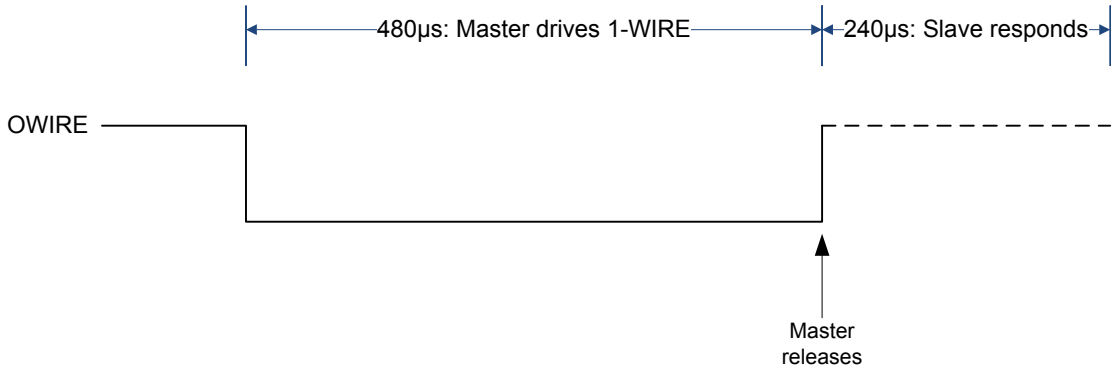
The 1-Wire protocol signals 1s and 0s by holding the line low for varying lengths of time as described below. For details on the operation of this device, see “Overdrive” on page 1513 Before a command is sent, a reset is issued on the line. This is a two-part operation:

- The 1-Wire Master module drives and holds line low for >480 µs.
- The controller waits for an answer-to-reset from one or more slaves. A slave signals a reset by pulling the line low for 60 µs to 240 µs. If the line is not sampled low, there is no slave on the bus and the NOATR bit is set in the **1-Wire Control and Status (ONEWIRECS)** register. An interrupt mask can also be set to trigger an interrupt on this condition. If the line is sampled low, it indicates there are one or more slaves on the 1-Wire bus. The Master samples the line some period after releasing the line, taking into consideration the time needed for the slave to respond. The time from reset release to first sample is programmed through the ATRSAM bit in the **1-Wire**

**Timing Override (ONEWIRETIM)** register. For example, the Master could sample 10 µs after releasing the reset for 240 µs. Caution should be exercised to make sure the line has been pulled high (by a pull-up) before the Master samples to avert a bogus answer-to-reset. Because the slave may hold the line low for a longer duration than the sample time, the Master must wait for the line to go high before starting a new command. This reset protocol is used to ensure that all slaves are in a known state.

Figure 22-2 on page 1510 shows the details of the 1-wire reset.

**Figure 22-2. 1-Wire Reset Protocol**




If the Master is transmitting data to the slave, the signalling is as follows:

- A 1 is signaled by the master driving and holding the line low for <15 µs. Generally, about 6 µs is used for normal mode. The slave samples and measures the signal from falling edge and checks the line 15 µs later (or more). If line has reverted to high, the slave registers a 1 value.
- A 0 is signaled by the Master driving and holding the line low for 60 µs or more. Although the slave reads just past 15 µs, the normal mode requires the line to be low for 60 µs. If the line is still low after 15 µs, a value of 0 is registered.

Figure 22-3 on page 1511 depicts a 1-Wire Master transmitting a 1 to a slave. Figure 22-4 on page 1511 shows a 1-wire Master transmitting a 0 to a slave.

**Figure 22-3. 1-Wire Master Transmitting a 1**

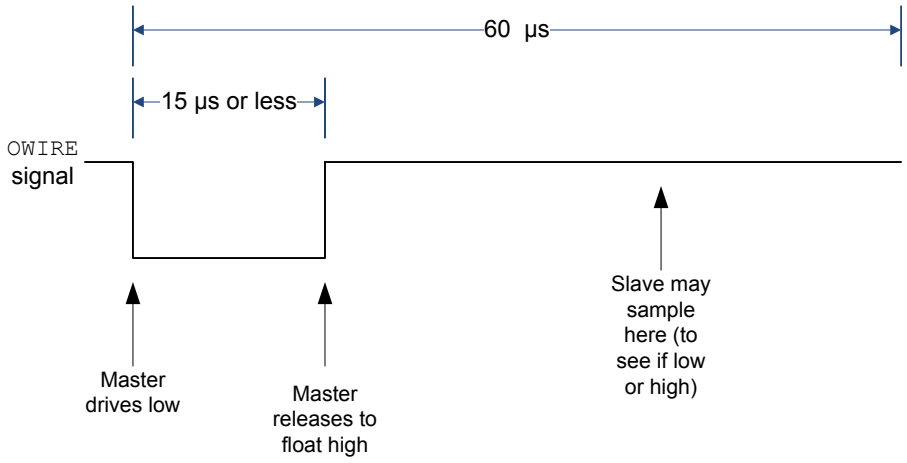
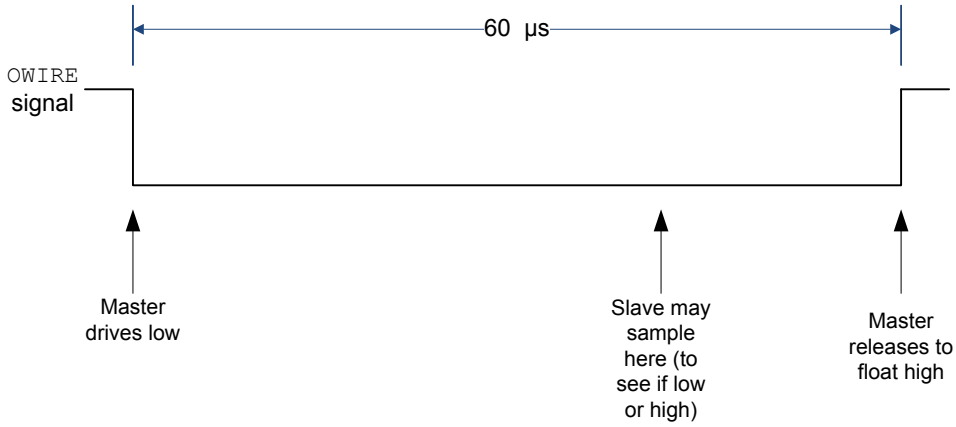



**Figure 22-4. 1-Wire Master Transmitting a 0'**




For a read from the slave, the Master drives and holds the line low for at least 1µs (but less than 15 µs) and then releases. A typical hold time is 6 µs; in other words, it uses a transmitted 1. The read is performed as:

- A 1 is signaled by the slave when "does nothing." Thus, the slave just allows the line to come back up when the master releases it. The master samples the line at 15µs after the low edge and the line is still high, signaling a 1.
- If the slave holds the line down for more than 5 µs (and no more than 60 µs), a 0 is signaled. The master samples the line at 15 µs after the low edge, and because the line is low still, it is treated as a 0.

Figure 22-5 on page 1512 depicts a Master receiving a 1 value from a slave. Figure 22-6 on page 1512 shows the slave sending a 0 value to a Master.

**Figure 22-5. 1-Wire Master Receiving a 1 Signal from a Slave**

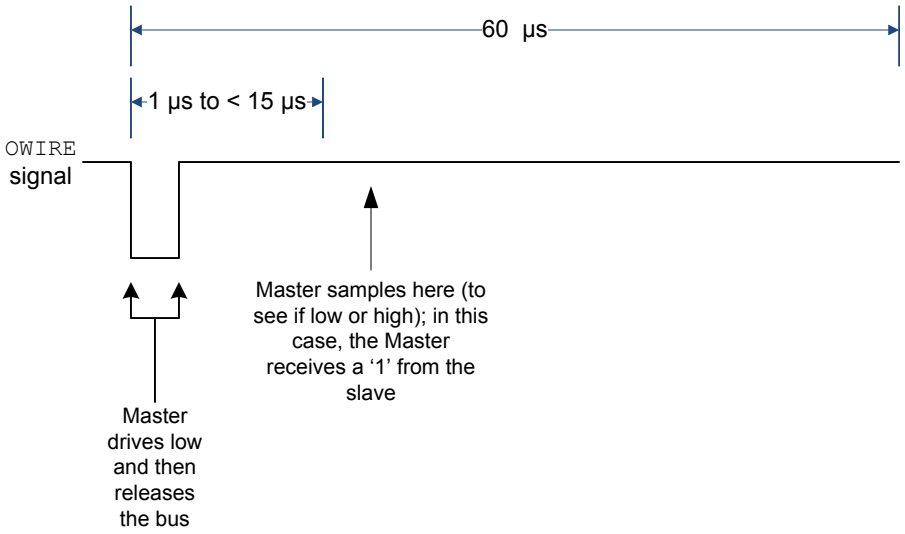
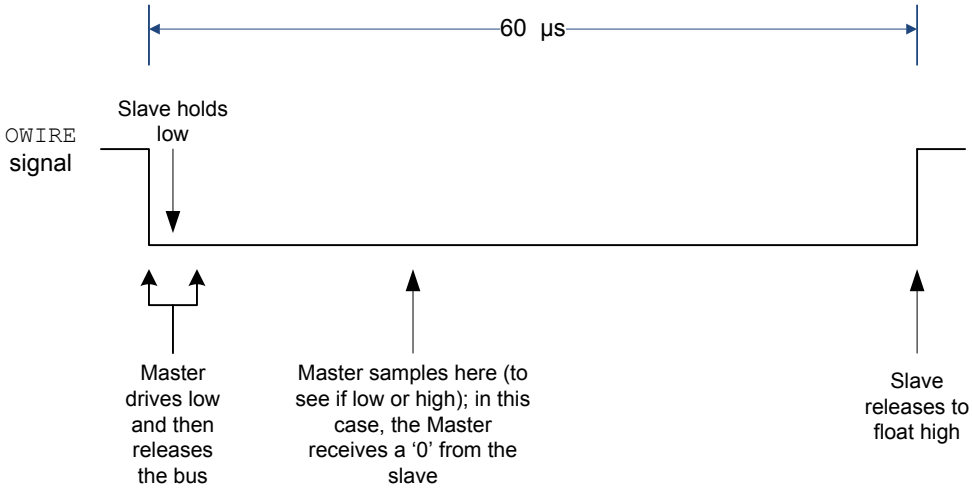



**Figure 22-6. 1-Wire Master Receiving a 0 Signal from a Slave**




## 22.3.2 Transport Protocol

The transport level protocol follows a simple model, in part based on treating most 1-wire devices as memory like (IDs, ROM, EEPROM, etc) even if some are sensors or actuators. All commands and data are in 8-bit (byte) quantities and the post-reset commands are pre-defined. The basic protocol is:

- Reset the line by driving and holding the OWIRE signal low for >480 µs.
- Wait for answer-to-reset from one or more slaves. Answer-to-reset (ATR) is defined as each slave pulling the line low for 60 µs to 240 µs. If the line does not go low, there is no slave on the bus. If it does, there is one or more.
- At this point the Master either figures out the ID of the slaves on the bus using the enumeration scheme, or selects a slave to issue a command. The "select" command to use is based on

whether there is one slave or more than one. If there is only one slave device, the Master skips the selection process. If more than one are present, the Master selects a slave by sending out the ID for the selected slave, indicating to the remaining slaves that any further commands should be ignored until after the next reset.

- The Master sends a command followed by a sending or receiving of data. While most commands are implemented using bytes read or written, some commands use bytes where some bits are written and some read. Since a read looks like a write 1, if the line is held and driven low for less than 15 µs and the slave does not know it is supposed to reply, the line remains untouched and is read as a 1 (and written as a 1).
- Usually at the end of one or more commands to that slave, a reset is issued again.

The 1-Wire module supports reset and answer-to-reset on request (although answer-to-reset can be suppressed). It also supports byte operations. The software chooses which read, write, or mixed operation to use. Partial byte (bit or multi-bit) is also supported.

### 22.3.3 Overdrive

The 1-Wire Module also allows for an "overdrive" speed to be selected for shorter line lengths. Overdrive is about ten times faster because it uses smaller time periods. It is agreed by both master and slave before enabling. Overdrive is selected using the ODRV bit of the **1-Wire Control and Status (ONEWIRECS)** register.

In overdrive mode, the time periods shrink. Table 22-2 on page 1513 shows the active time for 1-Wire operations in overdrive mode. For example, the Master typically drives and holds the line low for 15 µs for a write 1, but in overdrive mode this is 1 µs.

**Table 22-2. Active Time Periods in Overdrive Mode**

| Operation | Time | Notes |
|---|---|---|
| Write 1 | 1 µs low | Less than 2 µs low |
| Write 0 | 7.5 µs low | Up to 10 µs low |
| Read 1 | 1 µs low | Master samples at 2 µs |
| Read 0 | 9 µs low | Slave holds at least 7 µs |
| Reset | 70 µs low | Note 2 µs delay before reset for safety |
| Answer to Reset | 4 µs to 48 µs | Master samples at 8.5 µs |

### 22.3.4 Timing Override

The 1-Wire module can manually override timing for any main action. Each non-zero value in the **1-Wire Timing Override (ONEWIRETIM)** register replaces the default normal or overdrive time rule.

The time factors that can be controlled:

**Table 22-3. Bit Field Definitions for 1-Wire Timing and Override (ONEWIRETIM) Register**

| Name | Meaning | Range (if used) | Scale | Default (normal) | Default (overdrive) |
|---|---|---|---|---|---|
| W1TIM | Amount of time to drive and hold line low for a write 1 | 1 to 15 µs | 1 µs units | 6 µs | 1 µs |
| W0TIM | Amount of time to drive and hold line low for write 0[a] | 4 to 124 µs | 4 µs units | 60 µs | 7.5 µs |

**Table 22-3. Bit Field Definitions for 1-Wire Timing and Override (ONEWIRETIM) Register *(continued)***

| Name | Meaning | Range (if used) | Scale | Default (normal) | Default (overdrive) |
|---|---|---|---|---|---|
| W0REST | Amount of time to hold line high after a write 0 (rest) [a] | 1 to 15 µs | 1 µs units | 10 µs | 2.5 µs |
| W1SAM | Amount of time after 1-Wire has released the line to sample slave holding line | 1 to 15 µs | 1 µs units | 10 µs | 1 µs |
| RSTTIM | Amount of time to drive and hold line low for a reset. Allow for an additional 2-µs pre-reset period | 32 to 2016 µs | 32 µs units | 480 µs | 70 µs |
| ATRTIM | Period of answer-to-reset or rest. The ATRSAM bit value must be considered when programming this field. ATRTIM must be greater than ATRSAM to avoid a STUCK fault generation | 16 to 496 µs | 16 µs units | 240 µs | 48 µs |
| ATRSAM | Period in which first sample occurs in answer-to-reset (after line goes high). If the SKATR bit is set in the **ONEWIRECS** register, this value indicates the rest period after reset | 8 to 120 µs | 8 µs units | 10 µs[b] | 2 µs |

a. If W0TIM has been programmed to a non-default value, W0REST must be programmed for correct functionality.

b. ATRSAM is the first sample. If the line is high, it continues sampling for the rest of ATRTIM period to see if goes low. This subsequent sampling is every 62.5 ns. Thus, this value must be long enough to be sure the line has floated high, but does not have to be after the time the slaves have driven low. This rule is different than a single sample model, which can be too late or miss on long lines.

## 22.3.5 Command Protocol

The standard 1-Wire protocol has commands to be used after a reset. These commands normally include slave selection and detection. However, many 1-Wire buses only have one slave, and in that case, those mechanisms can be skipped. If slave management is needed, a small software stack can be used to handle these operations. Other protocols can also be easily supported, whether byte-oriented (as normal) or bit-oriented. To support an alternative protocol, it is usually necessary to override some or all of the timing and may be necessary to disable the answer-to-reset mechanism using the SKATR bit in the **ONEWIRECS** register. The module is designed to support the most common uses of 1-Wire, including a mechanism to allow a reset followed by an operation of 1, 2, 3, or 4 bytes through the SZ bit in the **ONEWIRECS** register. Additionally, the OP bit supports a write/read model to allow a mix of write and read in one transaction. Although nominally mixed mode allows for a mix of write and read bytes, it can be used down to the bit level.

The following represents some typical operation flows:

1. To only send a reset, the RST bit of the **ONEWIRECS** register is set, performing a reset and triggering an interrupt if enabled. If there is no answer-to-reset, the NOATR status bit is set and the corresponding interrupt bit can be set as well.

2. The SZ and OP fields of the **ONEWIRECS** register are used to perform a write, a read, or a combination. For small transactions, the SZ encoding can be programmed for processing of up to 4 bytes in one transaction. For larger transactions, the OP field can be used multiple times. The OPC raw interrupt bit is used to notify when a transaction is complete.

3. To perform a reset and then a transaction, both the RST and OP/SZ fields of the **ONEWIRECS** register are configured. The reset runs to completion and then the transaction starts. The interrupt is used to notify when both are done.

To setup a read, write or mixed transaction, the following configuration needs to occur:

- For writes, the **1-Wire Data Write (ONEWIREDATW)** register must first be written with one, two, three or four bytes to send (as a word). Next the **ONEWIRECS** register is configured as described above.
- For reads, the **ONEWIRECS** register is written as described above. Then, when the transaction is complete, the **1-Wire Data Read (ONEWIREDATR)** register is read to collect the one, two, three or four bytes that were received (as a word).
- For write/read combination, the **ONEWIREDATW** register is written with one, two, three or four bytes to write or read. Then, the **ONEWIRECS** register is configured as described above. As the data is transferred from the **ONEWIREDATW** register to the 1-Wire line, any 1 that is transmitted creates a read that is registered in the **ONEWIREDATR** register. When the entire transaction is complete, the **ONEWIREDATR** register can be read to collect the bytes that were received by the 1-Wire Master. A **ONEWIREDATR** register bit contains a 0 for any bit that was written as a 0, a 1 for any bit that was written as a 1 or any bit not modified by the slave (may have been intentional 1 or ignored) and a 0 for any bit written as a 1 but read back as a '0.'

### 22.3.6 Search (Enumeration) and Sub-Byte

Although 1-Wire is a byte protocol, there are times when less than a byte must to be written or read. By programming the BSIZE field of the **ONEWIRECS** register, the number of bits to write and or read in the last byte can be configured. The BSIZE field can be used for a 1-Wire search (enumeration) to allow detect and select processing. Enumeration begins by executing the standard SEARCH (0xF0) command (and if available, the ALARM SEARCH (0xEC)) and then reading two bits for the ID, which is the first bit and its complement. Subsequently, a bit is emitted to select devices with that first matching bit followed by another read of two bits. This process is repeated for up to all 64 bits. To start over, a reset is used. In this way, a tree of IDs can be broken down until one unique ID is left. This can be repeated for each device. The BSIZE field can also be used for non-standard devices which do not fully follow protocol.

### 22.3.7 Interrupts

The interrupt model for the 1-Wire module allows interrupt service routines to drive continuations of operations, whether reading the results or processing more bytes. Most command sequences are small, so the module supports transactions of one, two, three, and four bytes at once. Because most commands start with a line reset, the operational model allows for a reset followed by the transaction, with an interrupt occurring when the transaction is complete. There is also support for enabling interrupts for reset complete (RST), no answer-to-reset (NOATR) and line-hold-low error detected (STUCK) in the **1-Wire Interrupt Mask (ONEWIREIM)** register.

### 22.3.8 DMA

The µDMA model is designed to support both normal µDMA operation on write or read, as well as scatter-gather operation. The normal mode of operation allows for continued writes of byte, half-word, or word (three-byte size is not supported by µDMA for repeated operation) as well as continued reads by byte, half-word, or word. Additionally, the first transaction may be started with a reset.

The **1-Wire DMA Control (ONEWIREDMA)** register is used to configure the µDMA. To use µDMA with 1-Wire transmits:

1. Select 1-Wire to use µDMA by programming the **DMA Channel Map Select n (DMACHMAPn)** register in the µDMA. See “Channel Assignments” on page 710 for more information.

2. Configure the µDMA to transfer bytes, half-words, or words from memory to the 1-Wire **ONEWIREDATW** register or from the **ONEWIREDATR** register through the **DMA Channel Control Word (DMACHCTL)** register (see page 735).

3. Set the SZ field of the **ONEWIRECS** register to byte, half-word, or word (0, 1, or 3). This is the only field programmed in the **ONEWIRECS** register.

4. Enable the DMA completion interrupt in the **ONEWIREIM** register. It is also recommended that the ERR and NOATR interrupts should be enabled when using the µDMA to ensure that the application is notified if the reset or transaction failed.

5. Write the DMAOP field of the **ONEWIREDMA** register with the encoding 0x2 to enable the 1-Wire module assert a µDMA request when **ONEWIREDATW** is empty. The RST bit in the **ONEWIREDMA** register can be set for reset to be asserted first before requesting a µDMA access.

6. When the µDMA transfer is complete, the DMA bit in the **1-Wire Raw Interrupt Status (ONEWIRERIS)** register is set and the **ONEWIREDMA** register is cleared.

A 1-Wire µDMA receive configuration is as follows:

1. Select 1-Wire to use µDMA by programming the **DMA Channel Map Sellect n (DMACHMAPn)** register in the µDMA. See “Channel Assignments” on page 710 for more information.

2. Configure the µDMA to transfer bytes, half-words, or words from the 1-Wire **ONEWIREDATR** register through the **DMA Channel Control Word (DMACHCTL)** register (see page 735).

3. Enable the DMA completion interrupt in the **ONEWIREIM** register. It is also recommended that the ERR and NOATR interrupts should be enabled when using the µDMA to ensure that the application is notified if the reset or transaction failed.

4. Write the DMAOP field of the **ONEWIREDMA** register with the encoding 0x1 or 0x3 to enable a read or read multiple by the µDMA. The RST bit in the **ONEWIREDMA** register can be set for reset to be asserted first before requesting a µDMA access.

5. Write the **ONEWIREDATW** register with 0xFFFF.FFFF to prime the read operations. If the RST bit is not set in the **ONEWIREDMA** register, the transaction begins. If RST was set, the read starts when reset is complete. At the end of the read operation, the µDMA receives a request to transfer the data from the **ONEWIREDATR** register. If the DMAOP bit was configured for a multiple read (0x3), then the next read transaction will automatically start. If not, then no further µDMA requests occur unless **ONEWIREDATR** is written to.

6. When a µDMA receive is complete, the DMA bit in the **1-Wire Raw Interrupt Status (ONEWIRERIS)** register is set and the **ONEWIREDMA** register is cleared.

The normal flow for writing/reading in one operation is:

1. Configure the DMAOP bit in the **ONEWIREDMA** register for a read, and optionally set the RST bit. If RST is selected, the reset is performed first, else the module waits for the next step.

2. Write the **ONEWIREDATW** register with one, two, or four bytes containing the mix of writes and reads. For example, to write 0x46, 0x20, and then read 2 bytes, set size to word (SZ=3) and write **ONEWIREDATW** with 0xFFFF2046 to send the 0x46 and 0x20 and then read the last two bytes. On completion, the **ONEWIREDATR** contains the read values in the upper two bytes. If

RST is not set in the **ONEWIREDMA** register, the transaction starts. If RST is set, the transaction starts when reset completes. At the end of the operation, the µDMA is requested to transfer the data from the **ONEWIREDATR** register (read).

3. When the µDMA is done, the DMA bit in the **ONEWIRERIS** register is set, allowing interrupt on completion by µDMA, and the **ONEWIREDMA** register is cleared.

It is recommended to enable trigger interrupts for ERR and NOATR when using µDMA so application is notified if reset or the transaction failed.

The natural model for using scatter-gather is:

1. Enable peripheral-scatter-gather in the µDMA module by programming the **DMA Channel Control Word (DMACHCTL)** register.

2. Set the DMAOP to read (not read multiple) and enable scatter gather in the 1-Wire module by setting the SG bit to 0x1 in the **ONEWIREDMA** register. This allows use of **ONEWDATW** register to push through transactions, with µDMA request being triggered off completion, except for the first transaction.

3. Scatter gather can then use a combination of write and read as needed.

### 22.3.9 1-Wire Timing

The 1-Wire module wire control operates using the Precision Internal Oscillator (PIOSC) to provide a reliable frequency with no need for baud dividers. The register interface operates using the system clock, which may be a different clock source. However, there is no visible impact from this separate clock model. Normal rules apply to use of the registers to ensure correct operation. In particular:

- The control register is used live (not buffered) by the communication engine. Once an operation is started, it must be allowed to finish before changes are made, such as starting a new operation or changing speed. The status bits in the **ONEWIRECS** register notify the application if the transaction is busy or if a line-hold-low error is detected. When the last write, read or read/write has occurred, the OPC bit is set in the **ONEWIRERIS** register.

- Data read by the 1-Wire Master is not valid or meaningful until the 1-Wire module signals that the data is ready (via interrupt, DMA, and status).

- Data written should be written before the operation is started and then should not be re-written until the 1-Wire module signals the data has been written (via interrupt, DMA, and status).

- The **1-Wire Timing Override (ONEWIRETIM)** register should not be changed while a transaction is taking place.

The **ONEWIREIM**, **ONEWIRERIS** and **ONEWIREMIS** registers may be written and read at any time with no risk.

## 22.4 Initialization and Configuration

The whole block may be enabled or disabled using the system control registers. The following example shows how to enable, force a reset and send a command.

1. Enable the 1-Wire Module by writing a value of 0x0000.0001 to the **1-Wire Run Mode Clock Gating Control (RCGCOWIRE)** register in the System Control Module, see page 420.

2. Enable the clock to the appropriate GPIO port that is used for the 1-Wire signals using the **General-Purpose Input/Output Run Mode Clock gating Control (RCGCGPIO)** in the System Control Module; see page 399. To find out which GPIO port to enable, refer to Table 31-5 on page 2081.
3. In the GPIO module, enable the appropriate pin for its alternate function using the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 803).
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the 1-Wire signals to the appropriate pins. See page 820 and Table 31-5 on page 2081.
5. Optionally enable interrupts to be notified when the transactions completes.
6. Write to the **ONEWIREDATW** register with the data to write (1, 2, 3, or 4 bytes).
7. Write to the **ONEWIRECS** register to set the RST bit and configure the OP (operation) and SZ fields. When configuring all three bit fields, the reset is performed first, followed by the operation (unless an error occurs) and then any interrupt triggers.
8. If normal interrupts are generated (for example, OPC and RST), clear these interrupts and read the **ONEWIREDATR** register if a read or read/write transaction occurred.
9. If the 1-Wire master has more data to write and/or read for the same command, repeat the previous two steps but without setting the RST bit. If a new command is required, repeat the previous two steps including the reset enable.

## 22.5 Register Map

Table 22-4 on page 1518 lists the 1-Wire Master Module registers. The offset listed is a hexadecimal increment to the register's address, relative to the 1-Wire Master module's base address 0x400B.6000.

**Table 22-4. OWIRE Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | ONEWIRECS | RW | 0x0000.0000 | 1-Wire Control and Status | 1520 |
| 0x004 | ONEWIRETIM | RW | 0x0000.0000 | 1-Wire Timing Override | 1524 |
| 0x008 | ONEWIREDATW | RW | 0x0000.0000 | 1-Wire Data Write | 1525 |
| 0x00C | ONEWIREDATR | RW | 0x0000.0000 | 1-Wire Data Read | 1526 |
| 0x100 | ONEWIREIM | RW | 0x0000.0000 | 1-Wire Interrupt Mask | 1527 |
| 0x104 | ONEWIRERIS | RO | 0x0000.0000 | 1-Wire Raw Interrupt Status | 1529 |
| 0x108 | ONEWIREMIS | RO | 0x0000.0000 | 1-Wire Masked Interrupt Status | 1531 |
| 0x10C | ONEWIREICR | RW1C | 0x0000.0000 | 1-Wire Interrupt Clear | 1533 |
| 0x120 | ONEWIREDMA | RW | 0x0000.0000 | 1-Wire µDMA Control | 1534 |
| 0xFC0 | ONEWIREPP | RO | 0x0000.0011 | 1-Wire Peripheral Properties | 1536 |

## 22.6 1-Wire Master Register Descriptions

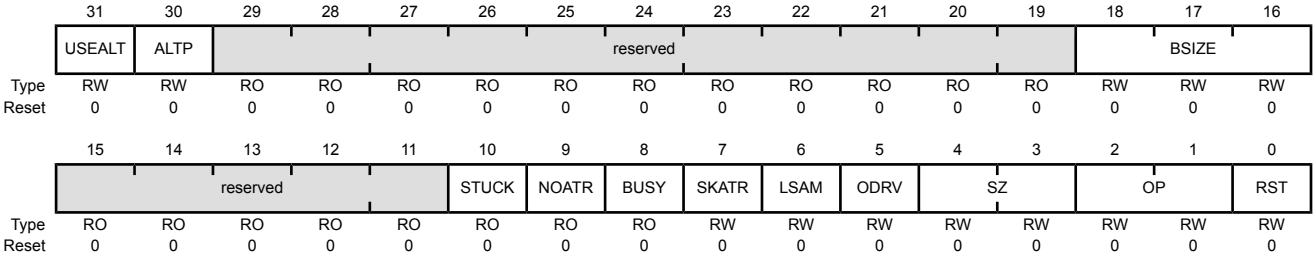
This section lists and describes the 1-Wire Master Module registers, in numerical order by address offset.

## Register 1: 1-Wire Control and Status (ONEWIRECS), offset 0x000

The **1-Wire Control and Status (ONEWIRECS)** register contains the control and status bits for the 1-Wire bus, including active state and passive detection. The OP field is written to start an operation. If executing a write or a write/read, the Bn field of the **1-Wire Data Write (ONEWIREDATW)** register must be written first before programming the OP field.

1-Wire Control and Status (ONEWIRECS)

Base 0x400B.6000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | USEALT | RW | 0 | Two Wire Enable<br><br>Value Description<br>0 Function in standard 1-Wire mode, using only the OWIRE signal for both input and output.<br>1 Use the OWIRE signal for the input and the OWALT signal for the output. |
| 30 | ALTP | RW | 0 | Alternate Polarity Enable<br><br>Value Description<br>0 Output pin is driven low to drive 1-wire line low.<br>1 Output pin is driven high to drive 1-Wire line low (that is, the output pin is connected to an NFET). |
| 29:19 | reserved | RO | 0x000 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 18:16 | BSIZE | RW | 0 | Last Byte Size<br>This field indicates the bit-size of the last byte. These bits are sent and received least significant bit first.<br><br>Value Description<br>0x0 8 bits (1 byte)<br>0x1 1 bit<br>0x2 2 bits<br>0x3 3 bits<br>0x4 4 bits<br>0x5 5 bits<br>0x6 6 bits<br>0x7 7 bits |
| 15:11 | reserved | RO | 0x00 | 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. |
| 10 | STUCK | RO | 0 | STUCK Status<br><br>Value Description<br>0 STUCK status is not detected.<br>1 Indicates line is being held low (other than in normal operation). The STUCK bit in the **ONEWIRERIS** is asserted when a line-hold-low error is detected. |
| 9 | NOATR | RO | 0 | Answer-to-Reset Status<br>This bit is disabled if the SKATR bit is set.<br><br>Value Description<br>0 Answer-to-reset behavior was as expected.<br>1 No answer-to-reset was detected after last reset (RST=1), which indicates no slave may be present on the bus. |
| 8 | BUSY | RO | 0 | Busy Status<br><br>Value Description<br>0 No activity on the bus.<br>1 Activity is taking place on the bus as a result of a reset, read, write or read/write transaction.<br><br>**Note:** Because this bit is on the PIOSC clock domain, it does not set immediately when a read, write, read/write or reset transaction begins. The OPC and RST bits in the **ONEWIRERIS** register should be used to verify when these operations have completed. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | SKATR | RW | 0 | Skip Answer-to-Reset Enable<br><br>Value Description<br>0 No effect.<br>1 Transaction goes from reset to first byte transfer without an answer-to-reset (presence detect) after programmed rest period has passed. |
| 6 | LSAM | RW | 0 | Late Sample Enable<br>This bit is used for long distance lines or when the slave cannot pull low quickly.<br><br>Value Description<br>0 Late sample disabled.<br>1 1-Wire module samples late in the read. The sample point moves to 50 µs for normal and 7µs for overdrive (versus 16 µs and 2 µs in normal operation) |
| 5 | ODRV | RW | 0 | Overdrive Enable<br><br>Value Description<br>0 Overdrive mode disabled.<br>1 Overdrive mode is enabled. |
| 4:3 | SZ | RW | 0 | Data Operation Size<br>This field is used to program the size in bytes of data operations transferred to and from the **ONEWIREDATn** registers per request.<br><br>Value Description<br>0x0 1 byte<br>0x1 2 bytes<br>0x2 3 bytes<br>0x3 4 bytes |

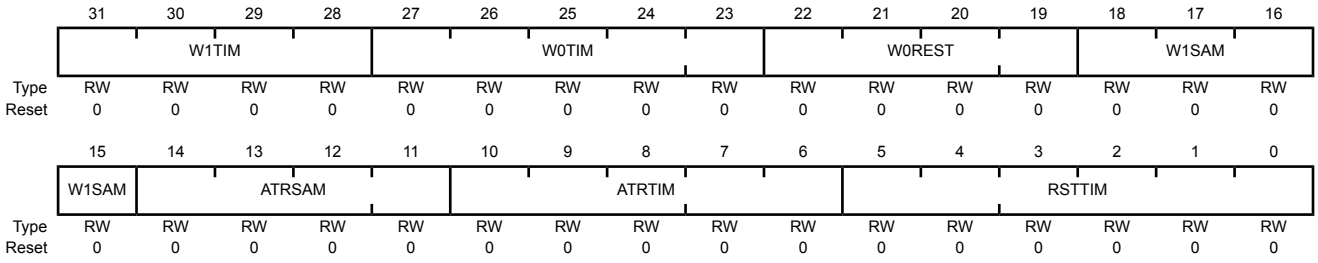
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2:1 | OP | RW | 0 | Operation Request<br>**Note:** If written with a non-zero value, this field requests data operations on the 1-Wire bus for number of bytes configured by the SZ field.<br>This field does not clear until the operation completes.<br>Value Description<br>0x0 No operation<br>0x1 Read<br>0x2 Write<br>0x3 Write/Read<br>The operation starts immediately. If using write or write/read, the **ONEWIREDATW** register must be written first.<br>**Note:** µDMA operations are handled by the **ONEWIREDMA** register. |
| 0 | RST | RW | 0 | Reset Request<br>If RST = 1, a reset operation begins. This bit clears when the reset operation is complete and the NOATR bit can be read to verify that a slave responded.<br>If this bit is set at the same time as the OP bit is set, the reset runs first and the operation starts when reset has completed. If this bit is written when an operation is already in progress, it cancels the operation and starts the reset.<br>Value Description<br>0 No effect.<br>1 Reset operation request. |

## Register 2: 1-Wire Timing Override (ONEWIRETIM), offset 0x004

The **1-Wire Timing Override (ONEWIRETIM)** register allows for overriding of the timing rules for the 1-Wire interface. If a field is 0, the default is used (normal or overdrive).

1-Wire Timing Override (ONEWIRETIM)

Base 0x400B.6000
Offset 0x004
Type RW, reset 0x0000.0000



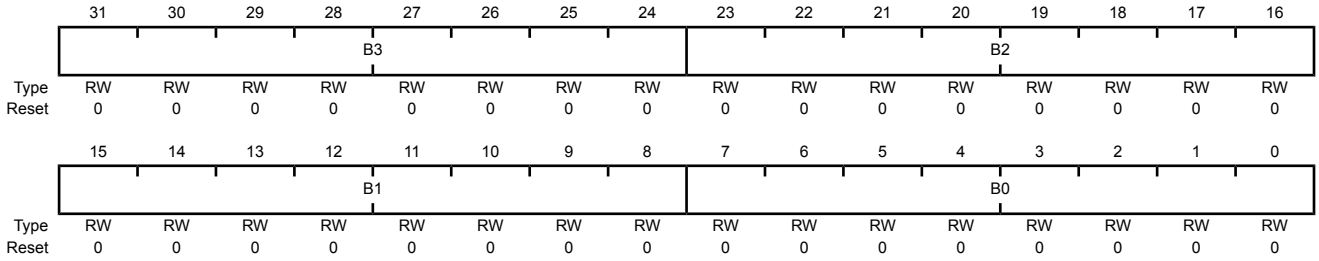

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | W1TIM | RW | 0 | Value 1 Timing<br>This field indicates the time low for a 1 in µs units. |
| 27:23 | W0TIM | RW | 0 | Value 0 Timing<br>Indicates time low for a 0 in 4 µs units.<br>**Note:** If `W0TIM` has been programmed to a non-default value, `W0REST` must be programmed for correct functionality. |
| 22:19 | W0REST | RW | 0 | Rest Time<br>This field indicates time to rest after a 0 in 1 µs units.<br>**Note:** If `W0TIM` has been programmed to a non-default value, `W0REST` must be programmed for correct functionality. |
| 18:15 | W1SAM | RW | 0 | Sample Time<br>Time to sample for a 1 vs 0 in µs after the release of the line by 1-Wire. The `LSAM` bit is ignored if this bit is set. |
| 14:11 | ATRSAM | RW | 0 | Answer-to-Reset Sample<br>Time to first sample for answer-to-reset in 8 µs units. Sampling continues until time indicated by `ATRTIM` bit.<br>**Note:** When programming `ATRTIM` and `ATRSAM`, the value of `ATRTIM` must be greater than `ATRSAM` to prevent the line from being flagged as STUCK. |
| 10:6 | ATRTIM | RW | 0 | Answer-to-Reset/Rest Period<br>This field indicates the answer-to-reset period in 16 µs units.<br>If the `SKATR` bit is set in the **ONEWIRECS** register, this value indicates the rest period after reset.<br>**Note:** When programming `ATRTIM` and `ATRSAM`, the value of `ATRTIM` must be greater than `ATRSAM` to prevent the line from being flagged as STUCK. |
| 5:0 | RSTTIM | RW | 0 | Reset Low Time<br>Indicates the time reset is low in 32 µs units. |

## Register 3: 1-Wire Data Write (ONEWIREDATW), offset 0x008

The **1-Wire Data Write (ONEWIREDATW)** and **1-Wire Data Read (ONEWIREDATR)** registers are used to transmit or receive data. Data is processed LSB first, which means the lowest (and possibly only) byte is transferred first. Note that for normal processor use of writing and write/read, the **ONEWIREDATW** register must be written before the OP field of the **ONEWIRECS** register is programmed. If reading (or write/read), the **ONEWIREDATR** register is read after completion (when OP returns to value 0). When doing write/read (OP=0x3), either register can be written. Because bits written as a 1 in the **ONEWIREDATR** register also act as a read, writing 0xFF is the same as a read of two bytes. Writing 0xF0 allows writing 0 for the lower nibble and reading the upper nibble. If the slave does not know it is a read, each written 1 reads back as 1.

1-Wire Data Write (ONEWIREDATW)

Base 0x400B.6000
Offset 0x008
Type RW, reset 0x0000.0000



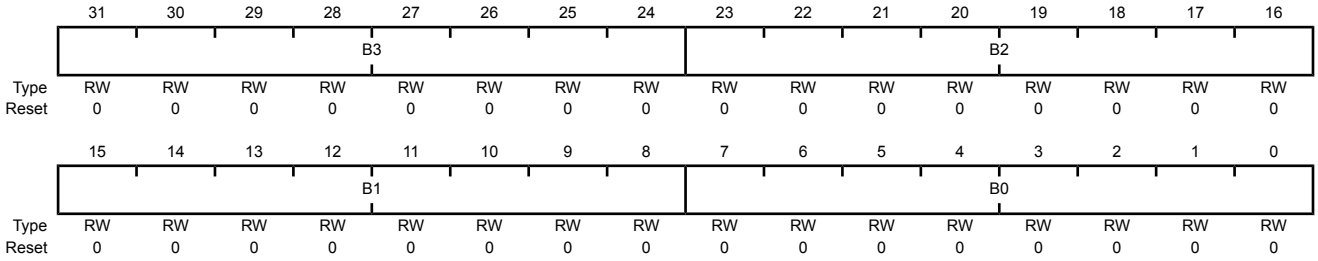

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | B3 | RW | 0x00 | Upper Data Byte<br>This data byte is used when SZ field = 3. |
| 23:16 | B2 | RW | 0x00 | Upper Middle Data Byte<br>Contains upper middle data byte and is used when SZ field = 2 or 3. |
| 15:8 | B1 | RW | 0x00 | Lower Middle Data Byte<br>Contains lower middle data byte and is used when SZ field = 1, 2, or 3. |
| 7:0 | B0 | RW | 0x00 | Lowest Data Byte<br>Contains byte data being read or written (depending on the operation). This byte is always used. |

## Register 4: 1-Wire Data Read (ONEWIREDATR), offset 0x00C

The **1-Wire Data Write (ONEWIREDATW)** and **1-Wire Data Read (ONEWIREDATR)** registers are used to transmit or receive data. Data is processed LSB first, which means the lowest (and possibly only) byte is transferred first. Note that for normal processor use of writing and write/read, the **ONEWIREDATW** register must be written before the OP field of the **ONEWIRECS** register is programmed. If reading (or write/read), the **ONEWIREDATR** register is read after completion (when OP returns to value 0). When doing write/read (OP=0x3), either register can be written. Because bits written as a 1 in the **ONEWIREDATR** register also act as a read, writing 0xFF is the same as a read of two bytes. Writing 0xF0 allows writing 0 for the lower nibble and reading the upper nibble. If the slave does not know it is a read, each written 1 reads back as 1.

1-Wire Data Read (ONEWIREDATR)

Base 0x400B.6000
Offset 0x00C
Type RW, reset 0x0000.0000



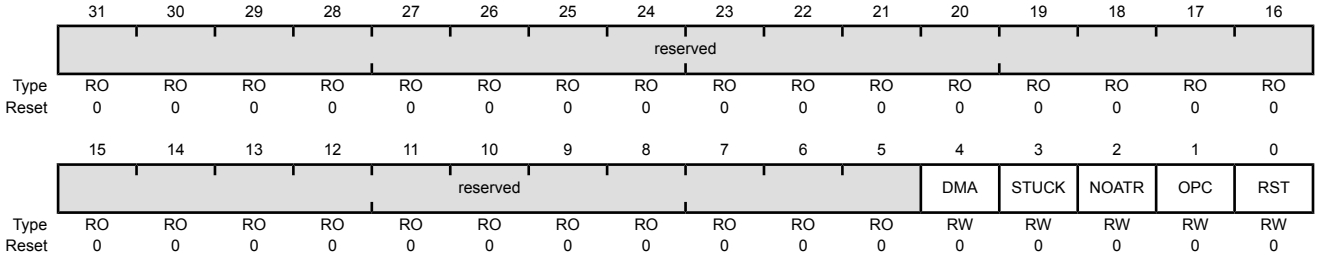

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | B3 | RW | 0x00 | Upper Data Byte<br>This data byte is used when SZ field = 3. |
| 23:16 | B2 | RW | 0x00 | Upper Middle Data Byte<br>Contains upper middle data byte and is used when SZ field = 2 or 3. |
| 15:8 | B1 | RW | 0x00 | Lower Middle Data Byte<br>Contains lower middle data byte and is used when SZ field = 1, 2, or 3. |
| 7:0 | B0 | RW | 0x00 | Lowest Data Byte<br>Contains byte data being read or written (depending on the operation). This byte is always used. |

## Register 5: 1-Wire Interrupt Mask (ONEWIREIM), offset 0x100

The **1-Wire Interrupt Mask (ONEWIREIM)** register enables the interrupt triggers for the 1-Wire Module.

1-Wire Interrupt Mask (ONEWIREIM)

Base 0x400B.6000
Offset 0x100
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMA | RW | 0 | DMA Done Interrupt Mask<br>Value Description<br>0 The DMA Done interrupt is suppressed and not sent to the interrupt controller.<br>1 The DMA done interrupt is sent to the interrupt controller when the DMA bit in the **ONEWIRERIS** register is set. |
| 3 | STUCK | RW | 0 | Stuck Status Interrupt Mask<br>When unmasked, this interrupt indicates a line-hold-low error is detected.<br>Value Description<br>0 The Stuck interrupt is suppressed and not sent to the interrupt controller.<br>1 The Stuck status interrupt is sent to the interrupt controller when the STUCK bit in the **ONEWIRERIS** register is set. |
| 2 | NOATR | RW | 0 | No Answer-to-Reset Interrupt Mask<br>Value Description<br>0 The No Answer-to-Reset interrupt is suppressed and not sent to the interrupt controller.<br>1 The No Answer-to-Reset interrupt is sent to the interrupt controller when the NOATR bit in the **ONEWIRERIS** register is set. |

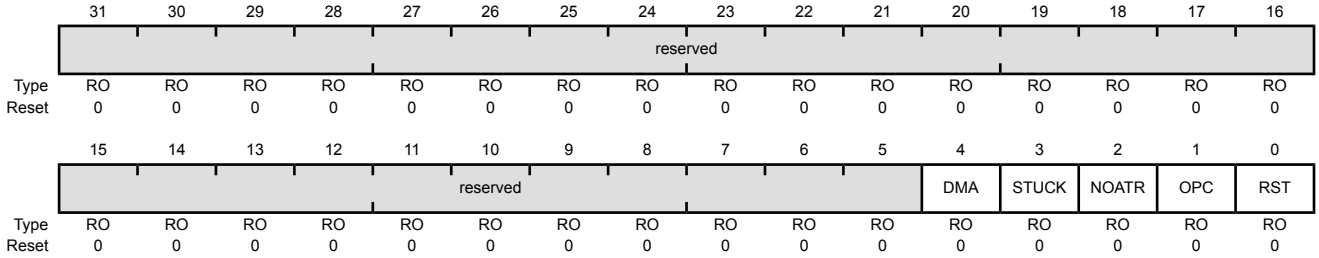
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | OPC | RW | 0 | Operation Complete Interrupt Mask<br>If this bit is set to a 1, an interrupt is sent when a write, read, or write/read completes. If a read or read/write transfer has occurred then the read data is ready to be accessed when this bit is set in the **ONEWIRERIS** register.<br>Value Description<br>0 The operation complete interrupt is suppressed and not sent to the interrupt controller.<br>1 The operation complete interrupt is sent to the interrupt controller when the OPC bit in the **ONEWIRERIS** register is set. |
| 0 | RST | RW | 0 | Reset Interrupt Mask<br>Value Description<br>0 The reset interrupt is suppressed and not sent to the interrupt controller.<br>1 The reset interrupt is sent to the interrupt controller when the RST bit in the **ONEWIRERIS** register is set. |

## Register 6: 1-Wire Raw Interrupt Status (ONEWIRERIS), offset 0x104

The **1-Wire Raw Interrupt Status (ONEWIRERIS)** register contains the raw interrupt status. If any of these bits read 1, the processor is interrupted if the corresponding masked interrupt status bit is set.

1-Wire Raw Interrupt Status (ONEWIRERIS)

Base 0x400B.6000
Offset 0x104
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMA | RO | 0 | DMA Done Raw Interrupt Status<br>Value Description<br>0 No interrupt<br>1 DMA transfer complete and an interrupt is pending. |
| 3 | STUCK | RO | 0 | Stuck Status Raw Interrupt Status<br>When unmasked, this interrupt indicates a line-hold-low error is detected.<br>Value Description<br>0 No interrupt<br>1 Stuck status (line-hold-error) is detected and an interrupt is pending. |
| 2 | NOATR | RO | 0 | No Answer-to-Reset Raw Interrupt Status<br>Value Description<br>0 No interrupt<br>1 A No Answer-to-Reset is detected from the last reset and an interrupt is pending. |

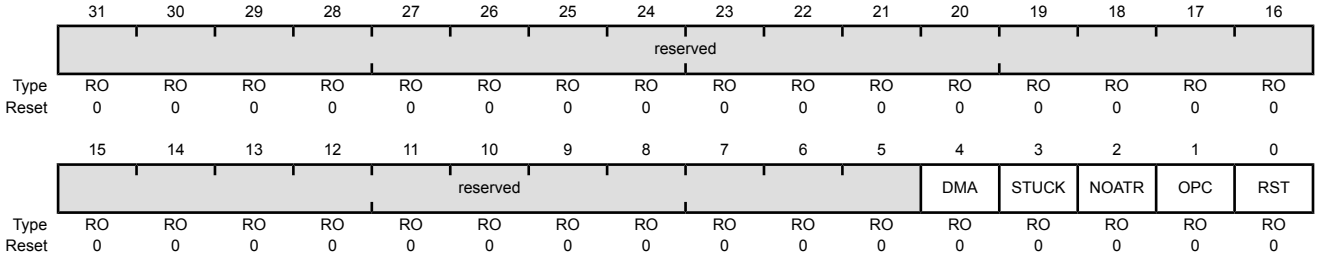
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | OPC | RO | 0 | Operation Complete Raw Interrupt Status<br>This bit indicates when a read, write or read/write operation has completed.<br>If a read or read/write transfer has occurred then the read data is ready to be accessed when this bit is set.<br>Value Description<br>0 No interrupt.<br>1 The last write, read, or write/read has completed and an interrupt is pending. |
| 0 | RST | RO | 0 | Reset Raw Interrupt Status<br>Value Description<br>0 No interrupt.<br>1 The last reset completed and an interrupt is pending. |

## Register 7: 1-Wire Masked Interrupt Status (ONEWIREMIS), offset 0x108

The **1-Wire Masked Interrupt Status (ONEWIREMIS)** register indicates when an unmasked interrupt has occurred.

1-Wire Masked Interrupt Status (ONEWIREMIS)

Base 0x400B.6000
Offset 0x108
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMA | RO | 0 | DMA Done Masked Interrupt Status<br>Value Description<br>0 No interrupt<br>1 A DMA transfer is complete and an unmasked interrupt is pending. |
| 3 | STUCK | RO | 0 | Stuck Status Masked Interrupt Status<br>When unmasked, this interrupt indicates a line-hold-low error is detected.<br>Value Description<br>0 No interrupt<br>1 An unmasked Stuck status (line-hold-error) interrupt is detected and pending. |
| 2 | NOATR | RO | 0 | No Answer-to-Reset Masked Interrupt Status<br>Value Description<br>0 No interrupt<br>1 A No Answer-to-Reset unmasked interrupt is detected from the last reset and pending. |
| 1 | OPC | RO | 0 | Operation Complete Masked Interrupt Status<br>Value Description<br>0 No Interrupt<br>1 An unmasked operation complete interrupt is pending. |

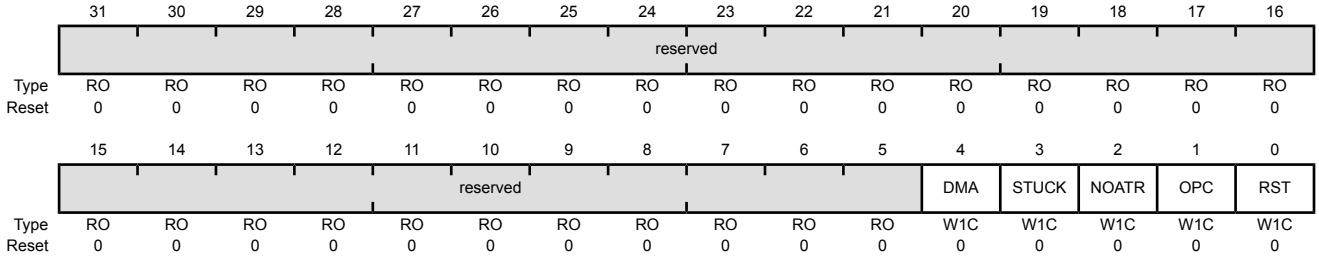
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RST | RO | 0 | Reset Interrupt Mask<br><br>Value Description<br>0 No Interrupt.<br>1 The unmasked reset interrupt is pending. |

## Register 8: 1-Wire Interrupt Clear (ONEWIREICR), offset 0x10C

The **1-Wire Interrupt Clear (ONEWIREICR)** register is used to clear the **ONEWIRERIS** register (and by extension, the **ONEWIREMIS** register).When read, this register contains the same value as the **ONEWIRERIS** register. To clear current interrupts, read this register and write the results back.

1-Wire Interrupt Clear (ONEWIREICR)

Base 0x400B.6000
Offset 0x10C
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x000 | 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. |
| 4 | DMA | W1C | 0 | DMA Done Interrupt Clear<br>Writing a 1 to this bit clears the DMA bit in the **ONEWIRERIS** register and the DMA bit in the **ONEWIREMIS** register. |
| 3 | STUCK | W1C | 0 | Stuck Status Interrupt Clear<br>Writing a 1 to this bit clears the STUCK bit in the **ONEWIRERIS** register and the STUCK bit in the **ONEWIREMIS** register. |
| 2 | NOATR | W1C | 0 | No Answer-to-Reset Interrupt Clear<br>Writing a 1 to this bit clears the NOATR bit in the **ONEWIRERIS** register and the NOATR bit in the **ONEWIREMIS** register. |
| 1 | OPC | W1C | 0 | Operation Complete Interrupt Clear<br>Writing a 1 to this bit clears the OPC bit in the **ONEWIRERIS** register and the OPC bit in the **ONEWIREMIS** register. |
| 0 | RST | W1C | 0 | Reset Interrupt Clear<br>Writing a 1 to this bit clears the RST bit in the **ONEWIRERIS** register and the RST bit in the **ONEWIREMIS** register. |

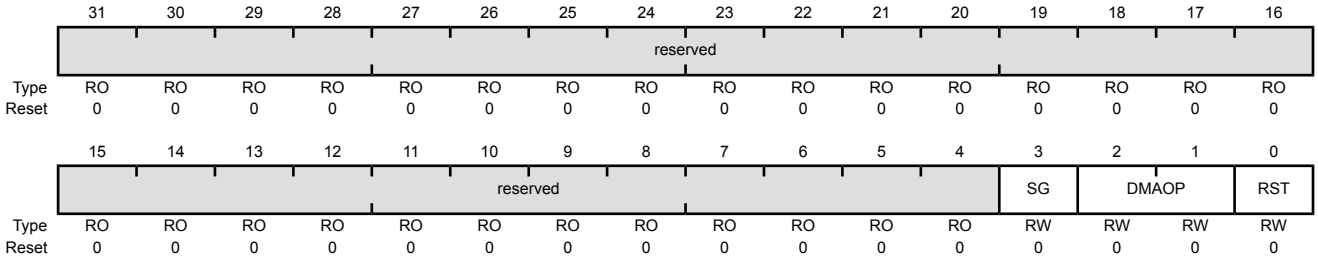
## Register 9: 1-Wire µDMA Control (ONEWIREDMA), offset 0x120

The **1-Wire DMA Control (ONEWIREDMA)** register is used to configure the µDMA operation for 1-Wire. This mechanism supports both µDMA write operations, µDMA read operations, small write/read operations, and scatter-gather support of mixed operations.

**Note:** This register is cleared when the dma_done signal to the 1-Wire module is asserted by the µDMA and the DMA bit in the **ONEWIRERIS** register is set.

1-Wire µDMA Control (ONEWIREDMA)

Base 0x400B.6000
Offset 0x120
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | SG | RW | 0 | Scatter-Gather Enable<br>This bit should be enabled when DMAOP = 0x1 and the scatter-gather method is being used.<br>This bit self clears after it has performed the µDMA request.<br>Value Description<br>0 No effect<br>1 µDMA is requested at start of operation and all requests after are on completion of transactions. |
| 2:1 | DMAOP | RW | 0x0 | µDMA Operation<br>The programmed operation starts when reset completes if the RST bit in the **ONEWIRECS** register is set. If RST is not set, then write requests the µDMA immediately and read starts on a write to the **ONEWIREDATW** register.<br>Value Description<br>0x0 µDMA disabled<br>0x1 µDMA single read: 1-Wire requests µDMA to read **ONEWIREDATR** register after each read transaction.<br>0x2 µDMA multiple write: 1-Wire requests µDMA to load whenever the **ONEWIREDATW** register is empty.<br>0x3 µDMA multiple read: An initial read occurs and subsequent reads start after µDMA has read the **ONEWIREDATR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RST | RW | 0 | µDMA Reset |

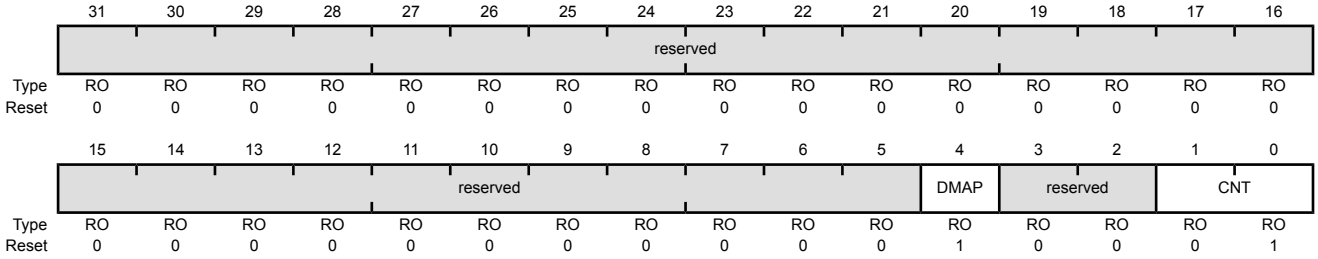
| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | A reset is issued and no reads and writes should be started until reset is done. Setting this bit sets the RST bit in the **ONEWIRECS** register. This bit is self-clearing upon reset completion. |

## Register 10: 1-Wire Peripheral Properties (ONEWIREPP), offset 0xFC0

The **1-Wire Peripheral Properties (ONEWIREPP)** register contains peripheral properties for the 1-Wire module.

1-Wire Peripheral Properties (ONEWIREPP)

Base 0x400B.6000
Offset 0xFC0
Type RO, reset 0x0000.0011




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x0 | 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. |
| 4 | DMAP | RO | 0x1 | µDMA Present<br>Indicates whether µDMA is available.<br>Value Description<br>0 No µDMA<br>1 µDMA is available to 1-Wire Module. |
| 3:2 | reserved | RO | 0x0 | 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. |
| 1:0 | CNT | RO | 0x1 | 1-Wire Bus Count<br>Indicates number of 1-Wire buses available.<br>Value Description<br>0x0 None<br>0x1 One 1-Wire bus.<br>0x2-0x3 Reserved |

# 23 Controller Area Network (CAN) Module

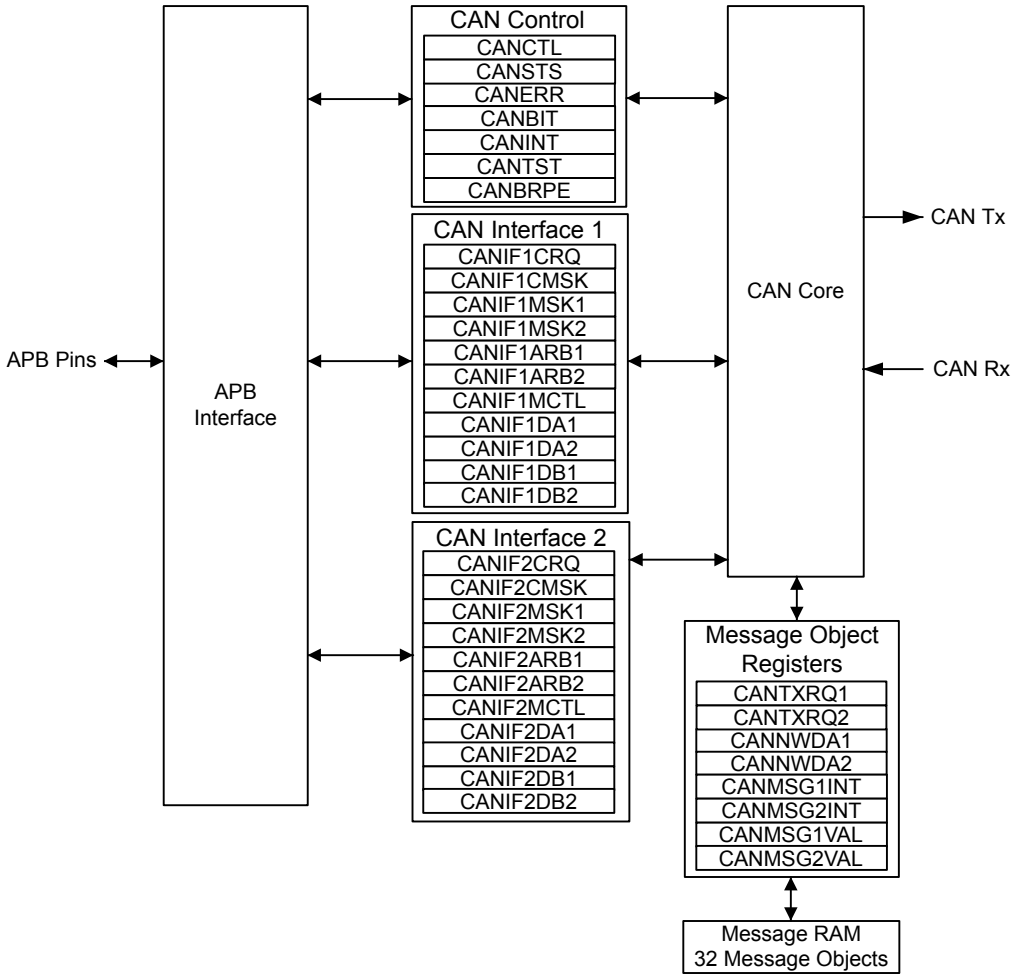
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally created for automotive purposes, it is also used in many embedded control applications (such as industrial and medical). Bit rates up to 1 Mbps are possible at network lengths less than 40 meters. Decreased bit rates allow longer network distances (for example, 125 Kbps at 500 meters).

The TM4C129XKCZAD microcontroller includes two CAN units with the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the CANnTX and CANnRX signals

## 23.1 Block Diagram

**Figure 23-1. CAN Controller Block Diagram**




## 23.2 Signal Description

The following table lists the external signals of the CAN controller and describes the function of each. The CAN controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the CAN signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the CAN controller function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the CAN signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 23-1. Controller Area Network Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| CAN0Rx | V3<br>W10 | PA0 (7)<br>PT0 (7) | I | TTL | CAN module 0 receive. |
| CAN0Tx | W3<br>V10 | PA1 (7)<br>PT1 (7) | O | TTL | CAN module 0 transmit. |
| CAN1Rx | A16<br>E18 | PB0 (7)<br>PT2 (7) | I | TTL | CAN module 1 receive. |
| CAN1Tx | B16<br>F17 | PB1 (7)<br>PT3 (7) | O | TTL | CAN module 1 transmit. |

## 23.3 Functional Description

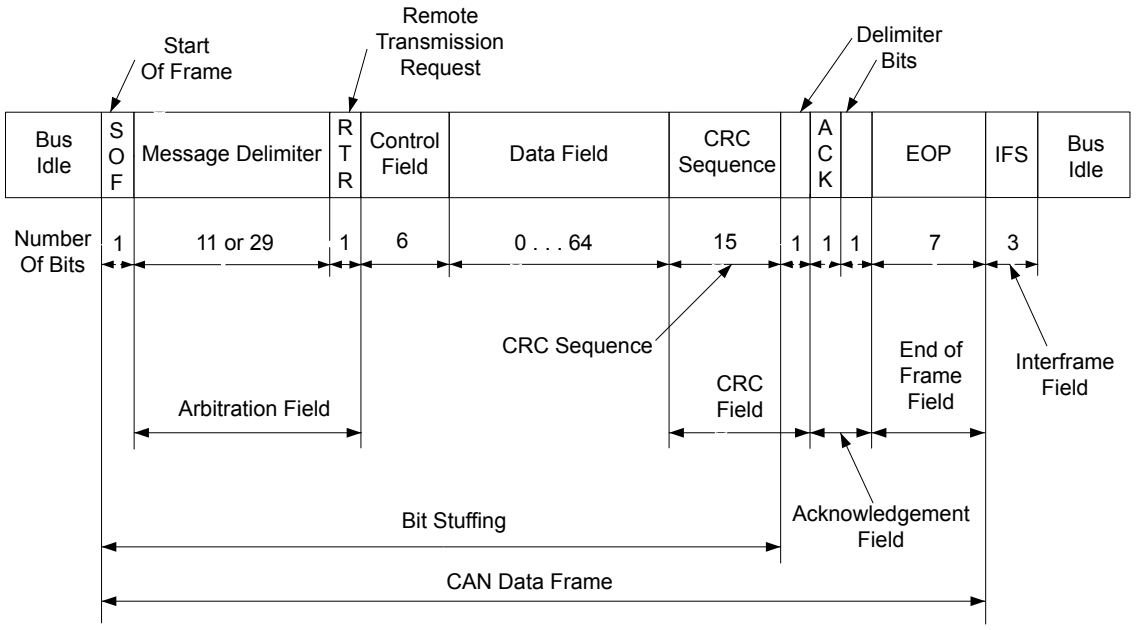
The TM4C129XKCZAD CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

A data frame contains data for transmission, whereas a remote frame contains no data and is used to request the transmission of a specific message object. The CAN data/remote frame is constructed as shown in Figure 23-2.

**Figure 23-2. CAN Data/Remote Frame**



The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These memory blocks are accessed via either of the CAN message object register interfaces.

The message memory is not directly accessible in the TM4C129XKCZAD memory map, so the TM4C129XKCZAD CAN controller provides an interface to communicate with the message memory via two CAN interface register sets for communicating with the message objects. These two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that must be processed. In general, one interface is used for transmit data and one for receive data.

### 23.3.1 Initialization

To use the CAN controller, the peripheral clock must be enabled using the **RCGC0** register (see page 412). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register (see page 412). To find out which GPIO port to enable, refer to Table 31-4 on page 2064. Set the GPIO AFSEL bits for the appropriate pins (see page 803). Configure the PMCn fields in the **GPIOPCTL** register to assign the CAN signals to the appropriate pins. See page 820 and Table 31-5 on page 2081.

Software initialization is started by setting the INIT bit in the **CAN Control (CANCTL)** register (with software or by a hardware reset) or by going bus-off, which occurs when the transmitter's error counter exceeds a count of 255. While INIT is set, all message transfers to and from the CAN bus are stopped and the CANnTX signal is held High. Entering the initialization state does not change the configuration of the CAN controller, the message objects, or the error counters. However, some configuration registers are only accessible while in the initialization state.

To initialize the CAN controller, set the **CAN Bit Timing (CANBIT)** register and configure each message object. If a message object is not needed, label it as not valid by clearing the MSGVAL bit in the **CAN IFn Arbitration 2 (CANIFnARB2)** register. Otherwise, the whole message object must be initialized, as the fields of the message object may not have valid information, causing unexpected results. Both the INIT and CCE bits in the **CANCTL** register must be set in order to access the **CANBIT** register and the **CAN Baud Rate Prescaler Extension (CANBRPE)** register to configure the bit timing. To leave the initialization state, the INIT bit must be cleared. Afterwards, the internal Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (indicating a bus idle condition) before it takes part in bus activities and starts message transfers. Message object initialization does not require the CAN to be in the initialization state and can be done on the fly. However, message objects should all be configured to particular identifiers or set to not valid before message transfer starts. To change the configuration of a message object during normal operation, clear the MSGVAL bit in the **CANIFnARB2** register to indicate that the message object is not valid during the change. When the configuration is completed, set the MSGVAL bit again to indicate that the message object is once again valid.

### 23.3.2 Operation

Two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**) are used to access the message objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The two sets are independent and identical and can be used to

queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

Once the CAN module is initialized and the INIT bit in the **CANCTL** register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the message handler's filtering process, and if it passes through the filter, is stored in the message object specified by the MNUM bit in the **CAN IFn Command Request (CANIFnCRQ)** register. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the MSK bits in the **CAN IFn Mask 1** and **CAN IFn Mask 2 (CANIFnMSKn)** registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time via the CAN Interface Registers. The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects is under the control of the software that is managing the CAN hardware. Message objects can be used for one-time data transfers or can be permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. At the start of transmission, the appropriate TXRQST bit in the **CAN Transmission Request n (CANTXRQn)** register and the NEWDAT bit in the **CAN New Data n (CANNWDAn)** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier (MNUM) for the message object, with 1 being the highest priority and 32 being the lowest priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the RMTEN bit in the **CAN IFn Message Control (CANIFnMCTL)** register. A matching received remote frame causes the TXRQST bit to be set, and the message object automatically transfers its data or generates an interrupt indicating a remote frame was requested. A remote frame can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are identified as remote frame requests. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are identified as a remote frame request. The MXTD bit in the **CANIFnMSK2** register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

### 23.3.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if a data transfer is not occurring between the CAN Interface Registers and message RAM, the valid message object with the highest priority that has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's NEWDAT bit in the **CANNWDAn** register is cleared. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the TXRQST bit in the **CANTXRQn** register is cleared. If the CAN controller is configured to interrupt on a successful transmission of a message object, (the TXIE bit in the **CAN IFn Message Control (CANIFnMCTL)** register is set), the INTPND bit in the **CANIFnMCTL** register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is

re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

### 23.3.4 Configuring a Transmit Message Object

The following steps illustrate how to configure a transmit message object.

1. In the **CAN IFn Command Mask (CANIFnCMASK)** register:
   - Set the WRNRD bit to specify a write to the **CANIFnCMASK** register; specify whether to transfer the IDMASK, DIR, and MXTD of the message object into the **CAN IFn** registers using the MASK bit
   - Specify whether to transfer the ID, DIR, XTD, and MSGVAL of the message object into the interface registers using the ARB bit
   - Specify whether to transfer the control bits into the interface registers using the CONTROL bit
   - Specify whether to clear the INTPND bit in the **CANIFnMCTL** register using the CLRINTPND bit
   - Specify whether to clear the NEWDAT bit in the **CANNWDAn** register using the NEWDAT bit
   - Specify which bits to transfer using the DATAA and DATAB bits
2. In the **CANIFnMSK1** register, use the MSK[15:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[15:0] in this register are used for bits [15:0] of the 29-bit message identifier and are not used for an 11-bit identifier. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.
3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.
4. For a 29-bit identifier, configure ID[15:0] in the **CANIFnARB1** register for bits [15:0] of the message identifier and ID[12:0] in the **CANIFnARB2** register for bits [28:16] of the message identifier. Set the XTD bit to indicate an extended identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.
5. For an 11-bit identifier, disregard the **CANIFnARB1** register and configure ID[12:2] in the **CANIFnARB2** register for bits [10:0] of the message identifier. Clear the XTD bit to indicate a standard identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.
6. In the **CANIFnMCTL** register:

- Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
- Optionally set the TXIE bit to enable the INTPND bit to be set after a successful transmission
- Optionally set the RMTEN bit to enable the TXRQST bit to be set on the reception of a matching remote frame allowing automatic transmission
- Set the EOB bit for a single message object
- Configure the DLC[3:0] field to specify the size of the data frame. Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

7. Load the data to be transmitted into the **CAN IFn Data (CANIFnDA1, CANIFnDA2, CANIFnDB1, CANIFnDB2)** registers. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register.
8. Program the number of the message object to be transmitted in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register.
9. When everything is properly configured, set the TXRQST bit in the **CANIFnMCTL** register. Once this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Note that setting the RMTEN bit in the **CANIFnMCTL** register can also start message transmission if a matching remote frame has been received.

### 23.3.5 Updating a Transmit Message Object

The CPU may update the data bytes of a Transmit Message Object any time via the CAN Interface Registers and neither the MSGVAL bit in the **CANIFnARB2** register nor the TXRQST bits in the **CANIFnMCTL** register have to be cleared before the update.

Even if only some of the data bytes are to be updated, all four bytes of the corresponding **CANIFnDAn**/**CANIFnDBn** register have to be valid before the content of that register is transferred to the message object. Either the CPU must write all four bytes into the **CANIFnDAn**/**CANIFnDBn** register or the message object is transferred to the **CANIFnDAn**/**CANIFnDBn** register before the CPU writes the new data bytes.

In order to only update the data in a message object, the WRNRD, DATAA and DATAB bits in the **CANIFnMSKn** register are set, followed by writing the updated data into **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** registers, and then the number of the message object is written to the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. To begin transmission of the new data as soon as possible, set the TXRQST bit in the **CANIFnMSKn** register.

To prevent the clearing of the TXRQST bit in the **CANIFnMCTL** register at the end of a transmission that may already be in progress while the data is updated, the NEWDAT and TXRQST bits have to be set at the same time in the **CANIFnMCTL** register. When these bits are set at the same time, NEWDAT is cleared as soon as the new transmission has started.

### 23.3.6 Accepting Received Message Objects

When the arbitration and control field (the ID and XTD bits in the **CANIFnARB2** and the RMTEN and DLC[3:0] bits of the **CANIFnMCTL** register) of an incoming message is completely shifted into the CAN controller, the message handling capability of the controller starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the controller uses the acceptance filtering programmed through the mask bits in the **CANIFnMSKn** register and enabled using the UMASK bit in the **CANIFnMCTL** register. Each valid

message object, starting with object 1, is compared with the incoming message to locate a matching message object in the message RAM. If a match occurs, the scanning is stopped and the message handler proceeds depending on whether it is a data frame or remote frame that was received.

### 23.3.7 Receiving a Data Frame

The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the DLC bits are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used. The NEWDAT bit of the **CANIFnMCTL** register is set to indicate that new data has been received. The CPU should clear this bit when it reads the message object to indicate to the controller that the message has been received, and the buffer is free to receive more messages. If the CAN controller receives a message and the NEWDAT bit is already set, the MSGLST bit in the **CANIFnMCTL** register is set to indicate that the previous data was lost. If the system requires an interrupt on successful reception of a frame, the RXIE bit of the **CANIFnMCTL** register should be set. In this case, the INTPND bit of the same register is set, causing the **CANINT** register to point to the message object that just received a message. The TXRQST bit of this message object should be cleared to prevent the transmission of a remote frame.

### 23.3.8 Receiving a Remote Frame

A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object have to be considered:

**Table 23-2. Message Object Configurations**

| Configuration in CANIFnMCTL | Description |
|---|---|
| ■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register<br>■ RMTEN = 1 (set the TXRQST bit of the **CANIFnMCTL** register at reception of the frame to enable transmission)<br>■ UMASK = 1 or 0 | At the reception of a matching remote frame, the TXRQST bit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible. |
| ■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register<br>■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame)<br>■ UMASK = 0 (ignore mask in the **CANIFnMSKn** register) | At the reception of a matching remote frame, the TXRQST bit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and nothing indicates that the remote frame ever happened. |
| ■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register<br>■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame)<br>■ UMASK = 1 (use mask (MSK, MXTD, and MDIR in the **CANIFnMSKn** register) for acceptance filtering) | At the reception of a matching remote frame, the TXRQST bit of this message object is cleared. The arbitration and control field (ID + XTD + RMTEN + DLC) from the shift register is stored into the message object in the message RAM, and the NEWDAT bit of this message object is set. The data field of the message object remains unchanged; the remote frame is treated similar to a received data frame. This mode is useful for a remote data request from another CAN device for which the TM4C129XKCZAD controller does not have readily available data. The software must fill the data and answer the frame manually. |

### 23.3.9 Receive/Transmit Priority

The receive/transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This prioritization is separate from that of the message identifier which is enforced by the CAN bus. As a result, if message object 1 and message object 2 both have valid messages to be transmitted, message object 1 is always transmitted first regardless of the message identifier in the message object itself.

### 23.3.10 Configuring a Receive Message Object

The following steps illustrate how to configure a receive message object.

1. Program the **CAN IFn Command Mask (CANIFnCMASK)** register as described in the "Configuring a Transmit Message Object" on page 1542 section, except that the WRNRD bit is set to specify a write to the message RAM.

2. Program the **CANIFnMSK1**and **CANIFnMSK2** registers as described in the "Configuring a Transmit Message Object" on page 1542 section to configure which bits are used for acceptance filtering. Note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

4. Program the **CANIFnARB1** and **CANIFnARB2** registers as described in the "Configuring a Transmit Message Object" on page 1542 section to program XTD and ID bits for the message identifier to be received; set the MSGVAL bit to indicate a valid message; and clear the DIR bit to specify receive.

5. In the **CANIFnMCTL** register:

   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
   - Optionally set the RXIE bit to enable the INTPND bit to be set after a successful reception
   - Clear the RMTEN bit to leave the TXRQST bit unchanged
   - Set the EOB bit for a single message object
   - Configure the DLC[3:0] field to specify the size of the data frame

   Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

6. Program the number of the message object to be received in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. Reception of the message object begins as soon as a matching frame is available on the CAN bus.

When the message handler stores a data frame in the message object, it stores the received Data Length Code and eight data bytes in the **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** register. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by unspecified values.

The CAN mask registers can be used to allow groups of data frames to be received by a message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are received by a message object. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are received. The MXTD bit in the **CANIFnMSK2** register should be set if only 29-bit extended identifiers are expected by this message object.

## 23.3.11 Handling of Received Message Objects

The CPU may read a received message any time via the CAN Interface registers because the data consistency is guaranteed by the message handler state machine.

Typically, the CPU first writes 0x007F to the **CANIFnCMSK** register and then writes the number of the message object to the **CANIFnCRQ** register. That combination transfers the whole received message from the message RAM into the Message Buffer registers (**CANIFnMSKn**, **CANIFnARBn**, and **CANIFnMCTL**). Additionally, the NEWDAT and INTPND bits are cleared in the message RAM, acknowledging that the message has been read and clearing the pending interrupt generated by this message object.

If the message object uses masks for acceptance filtering, the **CANIFnARBn** registers show the full, unmasked ID for the received message.

The NEWDAT bit in the **CANIFnMCTL** register shows whether a new message has been received since the last time this message object was read. The MSGLST bit in the **CANIFnMCTL** register shows whether more than one message has been received since the last time this message object was read. MSGLST is not automatically cleared, and should be cleared by software after reading its status.

Using a remote frame, the CPU may request new data from another CAN node on the CAN bus. Setting the TXRQST bit of a receive object causes the transmission of a remote frame with the receive object's identifier. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TXRQST bit is automatically reset. This prevents the possible loss of data when the other device on the CAN bus has already transmitted the data slightly earlier than expected.

### 23.3.11.1 Configuration of a FIFO Buffer

With the exception of the EOB bit in the **CANIFnMCTL** register, the configuration of receive message objects belonging to a FIFO buffer is the same as the configuration of a single receive message object (see “Configuring a Receive Message Object” on page 1545). To concatenate two or more message objects into a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest message object number is the first message object in a FIFO buffer. The EOB bit of all message objects of a FIFO buffer except the last one must be cleared. The EOB bit of the last message object of a FIFO buffer is set, indicating it is the last entry in the buffer.

### 23.3.11.2 Reception of Messages with FIFO Buffers

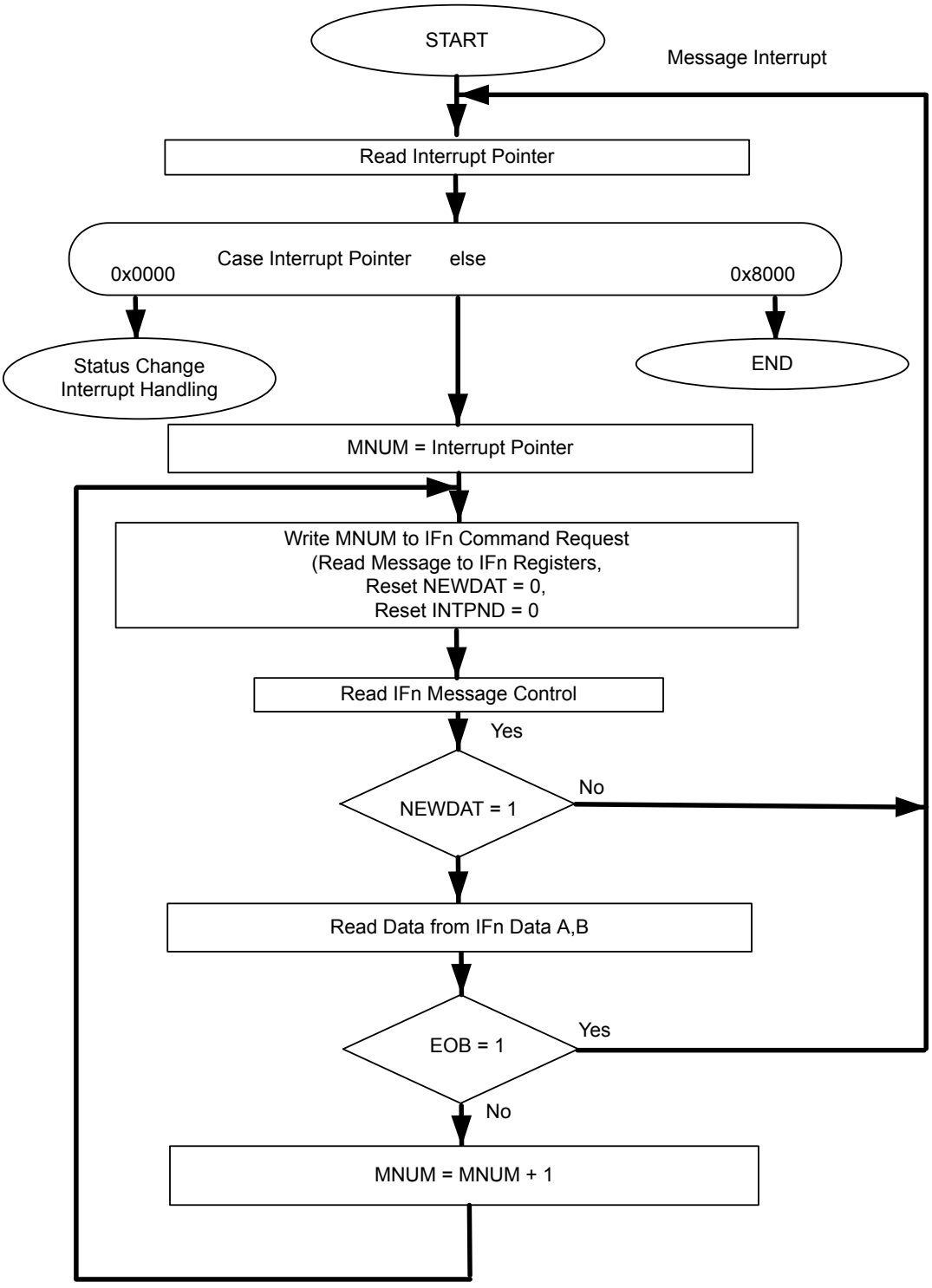
Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NEWDAT of the **CANIFnMCTL** register bit of this message object is set. By setting

NEWDAT while EOB is clear, the message object is locked and cannot be written to by the message handler until the CPU has cleared the NEWDAT bit. Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. Until all of the preceding message objects have been released by clearing the NEWDAT bit, all further messages for this FIFO buffer are written into the last message object of the FIFO buffer and therefore overwrite previous messages.

#### 23.3.11.3 Reading from a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the **CANIFnCRQ** register, the TXRQST and CLRINTPND bits in the **CANIFnCMSK** register should be set such that the NEWDAT and INTPEND bits in the **CANIFnMCTL** register are cleared after the read. The values of these bits in the **CANIFnMCTL** register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. When reading from the FIFO buffer, the user should be aware that a new received message is placed in the message object with the lowest message number for which the NEWDAT bit of the **CANIFnMCTL** register is clear. As a result, the order of the received messages in the FIFO is not guaranteed. Figure 23-3 on page 1548 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

**Figure 23-3. Message Objects in a FIFO Buffer**




## 23.3.12 Handling of Interrupts

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. The status interrupt has the highest

priority. Among the message interrupts, the message object's interrupt with the lowest message number has the highest priority. A message interrupt is cleared by clearing the message object's INTPND bit in the **CANIFnMCTL** register or by reading the **CAN Status (CANSTS)** register. The status Interrupt is cleared by reading the **CANSTS** register.

The interrupt identifier INTID in the **CANINT** register indicates the cause of the interrupt. When no interrupt is pending, the register reads as 0x0000. If the value of the INTID field is different from 0, then an interrupt is pending. If the IE bit is set in the **CANCTL** register, the interrupt line to the interrupt controller is active. The interrupt line remains active until the INTID field is 0, meaning that all interrupt sources have been cleared (the cause of the interrupt is reset), or until IE is cleared, which disables interrupts from the CAN controller.

The INTID field of the **CANINT** register points to the pending message interrupt with the highest interrupt priority. The SIE bit in the **CANCTL** register controls whether a change of the RXOK, TXOK, and LEC bits in the **CANSTS** register can cause an interrupt. The EIE bit in the **CANCTL**register controls whether a change of the BOFF and EWARN bits in the **CANSTS** register can cause an interrupt. The IE bit in the **CANCTL** register controls whether any interrupt from the CAN controller actually generates an interrupt to the interrupt controller. The **CANINT** register is updated even when the IE bit in the **CANCTL** register is clear, but the interrupt is not indicated to the CPU.

A value of 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed, the **CANSTS** register, indicating that either an error or status interrupt has been generated. A write access to the **CANSTS** register can clear the RXOK, TXOK, and LEC bits in that same register; however, the only way to clear the source of a status interrupt is to read the **CANSTS** register.

The source of an interrupt can be determined in two ways during interrupt handling. The first is to read the INTID bit in the **CANINT** register to determine the highest priority interrupt that is pending, and the second is to read the **CAN Message Interrupt Pending (CANMSGnINT)** register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and clear the message object's INTPND bit at the same time by setting the CLRINTPND bit in the **CANIFnCMSK** register. Once the INTPND bit has been cleared, the **CANINT** register contains the message number for the next message object with a pending interrupt.

### 23.3.13 Test Mode

A Test Mode is provided which allows various diagnostics to be performed. Test Mode is entered by setting the TEST bit in the **CANCTL** register. Once in Test Mode, the TX[1:0], LBACK, SILENT and BASIC bits in the **CAN Test (CANTST)** register can be used to put the CAN controller into the various diagnostic modes. The RX bit in the **CANTST** register allows monitoring of the CANnRX signal. All **CANTST** register functions are disabled when the TEST bit is cleared.

#### 23.3.13.1 Silent Mode

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The CAN Controller is put in Silent Mode setting the SILENT bit in the **CANTST** register. In Silent Mode, the CAN controller is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus remains in recessive state.

### 23.3.13.2 Loopback Mode

Loopback mode is useful for self-test functions. In Loopback Mode, the CAN Controller internally routes the CANnTX signal on to the CANnRX signal and treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into the message buffer. The CAN Controller is put in Loopback Mode by setting the LBACK bit in the **CANTST** register. To be independent from external stimulation, the CAN Controller ignores acknowledge errors (a recessive bit sampled in the acknowledge slot of a data/remote frame) in Loopback Mode. The actual value of the CANnRX signal is disregarded by the CAN Controller. The transmitted messages can be monitored on the CANnTX signal.

### 23.3.13.3 Loopback Combined with Silent Mode

Loopback Mode and Silent Mode can be combined to allow the CAN Controller to be tested without affecting a running CAN system connected to the CANnTX and CANnRX signals. In this mode, the CANnRX signal is disconnected from the CAN Controller and the CANnTX signal is held recessive. This mode is enabled by setting both the LBACK and SILENT bits in the **CANTST** register.

### 23.3.13.4 Basic Mode

Basic Mode allows the CAN Controller to be operated without the Message RAM. In Basic Mode, The CANIF1 registers are used as the transmit buffer. The transmission of the contents of the IF1 registers is requested by setting the BUSY bit of the **CANIF1CRQ** register. The CANIF1 registers are locked while the BUSY bit is set. The BUSY bit indicates that a transmission is pending. As soon the CAN bus is idle, the CANIF1 registers are loaded into the shift register of the CAN Controller and transmission is started. When the transmission has completed, the BUSY bit is cleared and the locked CANIF1 registers are released. A pending transmission can be aborted at any time by clearing the BUSY bit in the **CANIF1CRQ** register while the CANIF1 registers are locked. If the CPU has cleared the BUSY bit, a possible retransmission in case of lost arbitration or an error is disabled.

The CANIF2 Registers are used as a receive buffer. After the reception of a message, the contents of the shift register are stored in the CANIF2 registers, without any acceptance filtering. Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read message object is initiated by setting the BUSY bit of the **CANIF2CRQ** register, the contents of the shift register are stored into the CANIF2 registers.

In Basic Mode, all message-object-related control and status bits and of the control bits of the **CANIFnCMSK** registers are not evaluated. The message number of the **CANIFnCRQ** registers is also not evaluated. In the **CANIF2MCTL** register, the NEWDAT and MSGLST bits retain their function, the DLC[3:0] field shows the received DLC, the other control bits are cleared.

Basic Mode is enabled by setting the BASIC bit in the **CANTST** register.

### 23.3.13.5 Transmit Control

Software can directly override control of the CANnTX signal in four different ways.

- CANnTX is controlled by the CAN Controller
- The sample point is driven on the CANnTX signal to monitor the bit timing
- CANnTX drives a low value
- CANnTX drives a high value

The last two functions, combined with the readable CAN receive pin CANnRX, can be used to check the physical layer of the CAN bus.

The Transmit Control function is enabled by programming the TX[1:0] field in the **CANTST** register. The three test functions for the CANnTX signal interfere with all CAN protocol functions. TX[1:0] must be cleared when CAN message transfer or Loopback Mode, Silent Mode, or Basic Mode are selected.

### 23.3.14 Bit Timing Configuration Error Considerations

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronization amends a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration, however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive. The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

### 23.3.15 Bit Time and Bit Rate

The CAN system supports bit rates in the range of lower than 1 Kbps up to 1000 Kbps. Each member of the CAN network has its own clock generator. The timing parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods may be different.

Because of small variations in frequency caused by changes in temperature or voltage and by deteriorating components, these oscillators are not absolutely stable. As long as the variations remain inside a specific oscillator's tolerance range, the CAN nodes are able to compensate for the different bit rates by periodically resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 23-4 on page 1552): the Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 23-3 on page 1552). The length of the time quantum ($t_q$), which is the basic time unit of the bit time, is defined by the CAN controller's input clock (fsys) and the Baud Rate Prescaler (BRP):

$t_q$ = BRP / fsys

The fsys input clock is the system clock frequency as configured by the **RSCLKCFG** register (see page 287).

The Synchronization Segment Sync is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync and the Sync is called the phase error of that edge.

The Propagation Time Segment Prop is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase1 and Phase2 surround the Sample Point.

The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

**Figure 23-4. CAN Bit Time**

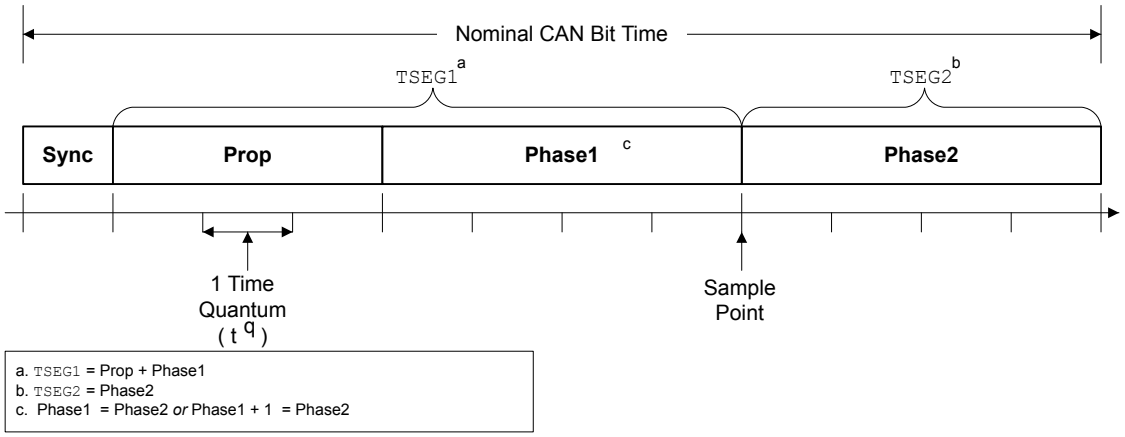



**Table 23-3. CAN Protocol Ranges**[a]

| Parameter | Range | Remark |
|---|---|---|
| BRP | [1 .. 64] | Defines the length of the time quantum $t_q$. The **CANBRPE** register can be used to extend the range to 1024. |
| Sync | 1 $t_q$ | Fixed length, synchronization of bus input to system clock |
| Prop | [1 .. 8] $t_q$ | Compensates for the physical delay times |
| Phase1 | [1 .. 8] $t_q$ | May be lengthened temporarily by synchronization |
| Phase2 | [1 .. 8] $t_q$ | May be shortened temporarily by synchronization |
| SJW | [1 .. 4] $t_q$ | May not be longer than either Phase Buffer Segment |

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. In the **CANBIT** register, the four components TSEG2, TSEG1, SJW, and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, for example, SJW (functional range of [1..4]) is represented by only two bits in the SJW bit field. Table 23-4 shows the relationship between the **CANBIT** register values and the parameters.

**Table 23-4. CANBIT Register Values**

| CANBIT Register Field | Setting |
|---|---|
| TSEG2 | Phase2 - 1 |
| TSEG1 | Prop + Phase1 - 1 |
| SJW | SJW - 1 |
| BRP | BRP |

Therefore, the length of the bit time is (programmed values):

$[TSEG1 + TSEG2 + 3] \times t_q$

or (functional values):

$[Sync + Prop + Phase1 + Phase2] \times t_q$

The data in the **CANBIT** register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) defines the length of the time quantum, the basic time

unit of the bit time; the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. In addition, the controller generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. The bit value is received or transmitted at the sample point. The information processing time (IPT) is the time after the sample point needed to calculate the next bit to be transmitted on the CAN bus. The IPT includes any of the following: retrieving the next data bit, handling a CRC bit, determining if bit stuffing is required, generating an error flag or simply going idle.

The IPT is application-specific but may not be longer than 2 $t_q$; the CAN's IPT is 0 $t_q$. Its length is the lower limit of the programmed length of Phase2. In case of synchronization, Phase2 may be shortened to a value less than IPT, which does not affect bus timing.

### 23.3.16 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a required bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the required bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is Prop. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop is converted into time quanta (rounded up to the nearest integer multiple of $t_q$).

Sync is 1 $t_q$ long (fixed), which leaves (bit time - Prop - 1) $t_q$ for the two Phase Buffer Segments. If the number of remaining $t_q$ is even, the Phase Buffer Segments have the same length, that is, Phase2 = Phase1, else Phase2 = Phase1 + 1.

The minimum nominal length of Phase2 has to be regarded as well. Phase2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of [0..2] $t_q$.

The length of the synchronization jump width is set to the least of 4, Phase1 or Phase2.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times fnom \leq fosc \leq (1 + df) \times fnom$$

where:

- df = Maximum tolerance of oscillator frequency
- fosc = Actual oscillator frequency
- fnom = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(Phase_seg1, Phase_seg2)\,\min}{2 \times (13 \times tbit - Phase_Seg2)}$$

$$df\ \max = 2 \times df \times fnom$$

where:

- Phase1 and Phase2 are from Table 23-3 on page 1552
- tbit = Bit Time
- dfmax = Maximum difference between two oscillators

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol-compliant configuration of the CAN bit timing.

### 23.3.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 25 MHz, and the bit rate is 1 Mbps.

```
bit time = 1 µs = n * tq = 5 * tq
tq = 200 ns
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 200E-9 * 25E6 = 5

tSync = 1 * tq = 200 ns                \\fixed at 1 time quanta

delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 400 ns = 2 * tq                  \\400 is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 5 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (5 * tq) - (1 * tq) - (2 * tq)
tPhase 1 + tPhase2 = 2 * tq
tPhase1 = 1 * tq
tPhase2 = 1 * tq                       \\tPhase2 = tPhase1
```

```
tTSeg1 = tProp + tPhase1
tTSeg1 = (2 * tq) + (1 * tq)
tTSeg1 = 3 * tq

tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 1) * tq
tTSeg2 = 1 * tq                          \\Assumes IPT=0

tSJW = 1 * tq                           \\Least of 4, Phase1 and Phase2
```

In the above example, the bit field values for the **CANBIT** register are:

| | |
|---|---|
| TSEG2 | = TSeg2 -1<br>= 1-1<br>= 0 |
| TSEG1 | = TSeg1 -1<br>= 3-1<br>= 2 |
| SJW | = SJW -1<br>= 1-1<br>= 0 |
| BRP | = Baud rate prescaler - 1<br>= 5-1<br>=4 |

The final value programmed into the **CANBIT** register = 0x0204.

### 23.3.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 50 MHz, and the bit rate is 100 Kbps.

```
bit time = 10 µs = n * tq = 10 * tq
tq = 1 µs
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 1E-6 * 50E6 = 50

tSync = 1 * tq = 1 µs                \\fixed at 1 time quanta

delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 1 µs = 1 * tq                  \\1 µs is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 10 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (10 * tq) - (1 * tq)  - (1 * tq)
tPhase 1 + tPhase2 = 8 * tq
tPhase1 = 4 * tq
tPhase2 = 4 * tq                     \\tPhase1 = tPhase2
```

```
tTSeg1 = tProp + tPhase1
tTSeg1 = (1 * tq) + (4 * tq)
tTSeg1 = 5 * tq
tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 4) × tq
tTSeg2 = 4 * tq                          \\Assumes IPT=0

tSJW = 4 * tq                            \\Least of 4, Phase1, and Phase2
```

| | |
|---|---|
| TSEG2 | = TSeg2 -1<br>= 4-1<br>= 3 |
| TSEG1 | = TSeg1 -1<br>= 5-1<br>= 4 |
| SJW | = SJW -1<br>= 4-1<br>= 3 |
| BRP | = Baud rate prescaler - 1<br>= 50-1<br>=49 |

The final value programmed into the **CANBIT** register = 0x34F1.

## 23.4 Register Map

Table 23-5 on page 1556 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000
- CAN1: 0x4004.1000

Note that the CAN controller clock must be enabled before the registers can be programmed (see page 412). There must be a delay of 3 system clocks after the CAN module clock is enabled before any CAN module registers are accessed.

**Table 23-5. CAN Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | CANCTL | RW | 0x0000.0001 | CAN Control | 1559 |
| 0x004 | CANSTS | RW | 0x0000.0000 | CAN Status | 1561 |
| 0x008 | CANERR | RO | 0x0000.0000 | CAN Error Counter | 1564 |
| 0x00C | CANBIT | RW | 0x0000.2301 | CAN Bit Timing | 1565 |
| 0x010 | CANINT | RO | 0x0000.0000 | CAN Interrupt | 1566 |
| 0x014 | CANTST | RW | 0x0000.0000 | CAN Test | 1567 |
| 0x018 | CANBRPE | RW | 0x0000.0000 | CAN Baud Rate Prescaler Extension | 1569 |
| 0x020 | CANIF1CRQ | RW | 0x0000.0001 | CAN IF1 Command Request | 1570 |

**Table 23-5. CAN Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x024 | CANIF1CMSK | RW | 0x0000.0000 | CAN IF1 Command Mask | 1571 |
| 0x028 | CANIF1MSK1 | RW | 0x0000.FFFF | CAN IF1 Mask 1 | 1574 |
| 0x02C | CANIF1MSK2 | RW | 0x0000.FFFF | CAN IF1 Mask 2 | 1575 |
| 0x030 | CANIF1ARB1 | RW | 0x0000.0000 | CAN IF1 Arbitration 1 | 1577 |
| 0x034 | CANIF1ARB2 | RW | 0x0000.0000 | CAN IF1 Arbitration 2 | 1578 |
| 0x038 | CANIF1MCTL | RW | 0x0000.0000 | CAN IF1 Message Control | 1580 |
| 0x03C | CANIF1DA1 | RW | 0x0000.0000 | CAN IF1 Data A1 | 1583 |
| 0x040 | CANIF1DA2 | RW | 0x0000.0000 | CAN IF1 Data A2 | 1583 |
| 0x044 | CANIF1DB1 | RW | 0x0000.0000 | CAN IF1 Data B1 | 1583 |
| 0x048 | CANIF1DB2 | RW | 0x0000.0000 | CAN IF1 Data B2 | 1583 |
| 0x080 | CANIF2CRQ | RW | 0x0000.0001 | CAN IF2 Command Request | 1570 |
| 0x084 | CANIF2CMSK | RW | 0x0000.0000 | CAN IF2 Command Mask | 1571 |
| 0x088 | CANIF2MSK1 | RW | 0x0000.FFFF | CAN IF2 Mask 1 | 1574 |
| 0x08C | CANIF2MSK2 | RW | 0x0000.FFFF | CAN IF2 Mask 2 | 1575 |
| 0x090 | CANIF2ARB1 | RW | 0x0000.0000 | CAN IF2 Arbitration 1 | 1577 |
| 0x094 | CANIF2ARB2 | RW | 0x0000.0000 | CAN IF2 Arbitration 2 | 1578 |
| 0x098 | CANIF2MCTL | RW | 0x0000.0000 | CAN IF2 Message Control | 1580 |
| 0x09C | CANIF2DA1 | RW | 0x0000.0000 | CAN IF2 Data A1 | 1583 |
| 0x0A0 | CANIF2DA2 | RW | 0x0000.0000 | CAN IF2 Data A2 | 1583 |
| 0x0A4 | CANIF2DB1 | RW | 0x0000.0000 | CAN IF2 Data B1 | 1583 |
| 0x0A8 | CANIF2DB2 | RW | 0x0000.0000 | CAN IF2 Data B2 | 1583 |
| 0x100 | CANTXRQ1 | RO | 0x0000.0000 | CAN Transmission Request 1 | 1584 |
| 0x104 | CANTXRQ2 | RO | 0x0000.0000 | CAN Transmission Request 2 | 1584 |
| 0x120 | CANNWDA1 | RO | 0x0000.0000 | CAN New Data 1 | 1585 |
| 0x124 | CANNWDA2 | RO | 0x0000.0000 | CAN New Data 2 | 1585 |
| 0x140 | CANMSG1INT | RO | 0x0000.0000 | CAN Message 1 Interrupt Pending | 1586 |
| 0x144 | CANMSG2INT | RO | 0x0000.0000 | CAN Message 2 Interrupt Pending | 1586 |
| 0x160 | CANMSG1VAL | RO | 0x0000.0000 | CAN Message 1 Valid | 1587 |
| 0x164 | CANMSG2VAL | RO | 0x0000.0000 | CAN Message 2 Valid | 1587 |

## 23.5 CAN Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers that are used to access the Message Objects in

the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

## Register 1: CAN Control (CANCTL), offset 0x000

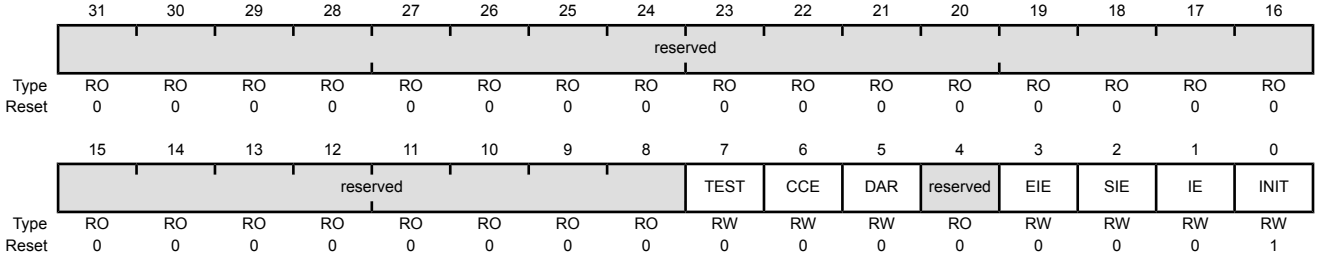
This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or clearing INIT. If the device goes bus-off, it sets INIT, stopping all bus activities. Once INIT has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 * 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after INIT is cleared, each time a sequence of 11 High bits has been monitored, a BITERROR0 code is written to the **CANSTS** register (the LEC field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

CAN Control (CANCTL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x000
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | TEST | RW | 0 | Test Mode Enable<br>Value: 0 – The CAN controller is operating normally.<br>1 – The CAN controller is in test mode. |
| 6 | CCE | RW | 0 | Configuration Change Enable<br>Value: 0 – Write accesses to the **CANBIT** register are not allowed.<br>1 – Write accesses to the **CANBIT** register are allowed if the INIT bit is 1. |
| 5 | DAR | RW | 0 | Disable Automatic-Retransmission<br>Value: 0 – Auto-retransmission of disturbed messages is enabled.<br>1 – Auto-retransmission is disabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | reserved | RO | 0 | 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. |
| 3 | EIE | RW | 0 | Error Interrupt Enable<br>Value: Description<br>0: No error status interrupt is generated.<br>1: A change in the BOFF or EWARN bits in the **CANSTS** register generates an interrupt. |
| 2 | SIE | RW | 0 | Status Interrupt Enable<br>Value: Description<br>0: No status interrupt is generated.<br>1: An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the **CANSTS** register generates an interrupt. |
| 1 | IE | RW | 0 | CAN Interrupt Enable<br>Value: Description<br>0: Interrupts disabled.<br>1: Interrupts enabled. |
| 0 | INIT | RW | 1 | Initialization<br>Value: Description<br>0: Normal operation.<br>1: Initialization started. |

## Register 2: CAN Status (CANSTS), offset 0x004

**Important:** This register is read-sensitive. See the register description for details.

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

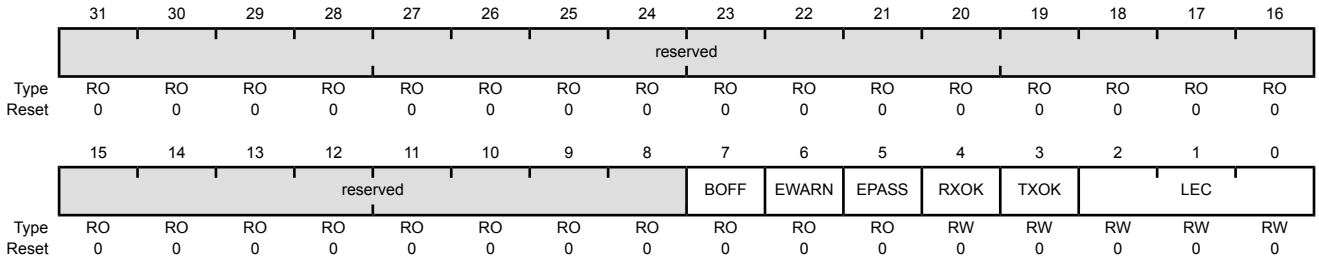
The LEC field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error. The unused error code 0x7 may be written by the CPU to manually set this field to an invalid error so that it can be checked for a change later.

An error interrupt is generated by the BOFF and EWARN bits, and a status interrupt is generated by the RXOK, TXOK, and LEC bits, if the corresponding enable bits in the **CAN Control (CANCTL)** register are set. A change of the EPASS bit or a write to the RXOK, TXOK, or LEC bits does not generate an interrupt.

Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Status (CANSTS)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x004
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | BOFF | RO | 0 | Bus-Off Status |

| Value | Description |
|---|---|
| 0 | The CAN controller is not in bus-off state. |
| 1 | The CAN controller is in bus-off state. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | EWARN | RO | 0 | Warning Status |

| Value | Description |
|---|---|
| 0 | Both error counters are below the error warning limit of 96. |
| 1 | At least one of the error counters has reached the error warning limit of 96. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | EPASS | RO | 0 | Error Passive<br><br>Value: Description<br>0: The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127.<br>1: The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127. |
| 4 | RXOK | RW | 0 | Received a Message Successfully<br><br>Value: Description<br>0: Since this bit was last cleared, no message has been successfully received.<br>1: Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering.<br><br>This bit must be cleared by writing a 0 to it. |
| 3 | TXOK | RW | 0 | Transmitted a Message Successfully<br><br>Value: Description<br>0: Since this bit was last cleared, no message has been successfully transmitted.<br>1: Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node.<br><br>This bit must be cleared by writing a 0 to it. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2:0 | LEC | RW | 0x0 | Last Error Code<br>This is the type of the last error to occur on the CAN bus. |

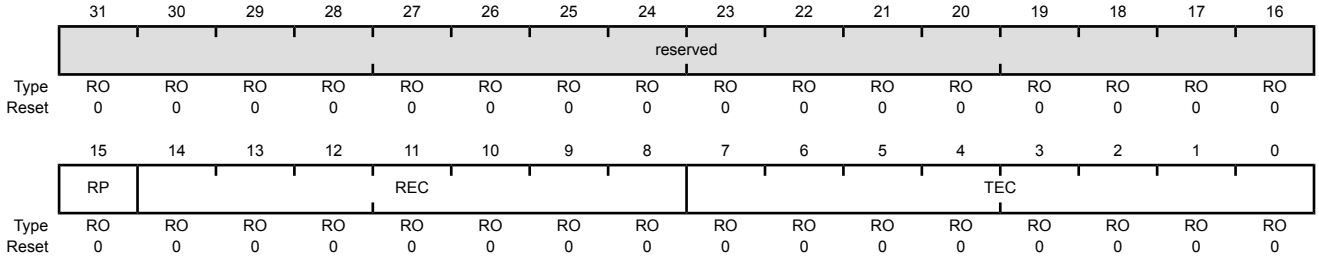
| Value | Description |
|---|---|
| 0x0 | No Error |
| 0x1 | Stuff Error<br>More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. |
| 0x2 | Format Error<br>A fixed format part of the received frame has the wrong format. |
| 0x3 | ACK Error<br>The message transmitted was not acknowledged by another node. |
| 0x4 | Bit 1 Error<br>When a message is transmitted, the CAN controller monitors the data lines to detect any conflicts. When the arbitration field is transmitted, data conflicts are a part of the arbitration protocol. When other frame fields are transmitted, data conflicts are considered errors.<br>A Bit 1 Error indicates that the device wanted to send a High level (logical 1) but the monitored bus value was Low (logical 0). |
| 0x5 | Bit 0 Error<br>A Bit 0 Error indicates that the device wanted to send a Low level (logical 0), but the monitored bus value was High (logical 1).<br>During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. By checking for this status, software can monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus. |
| 0x6 | CRC Error<br>The CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data. |
| 0x7 | No Event<br>When the LEC bit shows this value, no CAN bus event was detected since this value was written to the LEC field. |

## Register 3: CAN Error Counter (CANERR), offset 0x008

This register contains the error counter values, which can be used to analyze the cause of an error.

CAN Error Counter (CANERR)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x008
Type RO, reset 0x0000.0000



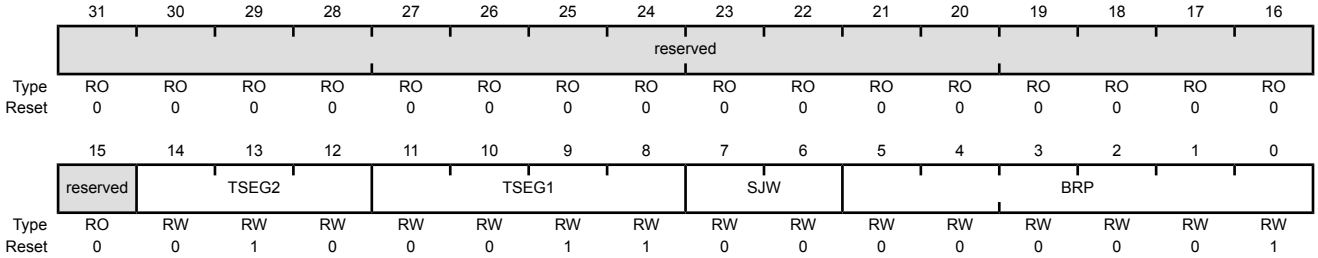

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15 | RP | RO | 0 | Received Error Passive<br>Value: 0 – The Receive Error counter is below the Error Passive level (127 or less).<br>Value: 1 – The Receive Error counter has reached the Error Passive level (128 or greater). |
| 14:8 | REC | RO | 0x00 | Receive Error Counter<br>This field contains the state of the receiver error counter (0 to 127). |
| 7:0 | TEC | RO | 0x00 | Transmit Error Counter<br>This field contains the state of the transmit error counter (0 to 255). |

## Register 4: CAN Bit Timing (CANBIT), offset 0x00C

This register is used to program the bit width and bit quantum. Values are programmed to the system clock frequency. This register is write-enabled by setting the CCE and INIT bits in the **CANCTL** register. See “Bit Time and Bit Rate” on page 1551 for more information.

CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x00C
Type RW, reset 0x0000.2301




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:15 | reserved | RO | 0x0000 | 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. |
| 14:12 | TSEG2 | RW | 0x2 | Time Segment after Sample Point<br>0x00-0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>So, for example, the reset value of 0x2 means that 3 (2+1) bit time quanta are defined for Phase2 (see Figure 23-4 on page 1552). The bit time quanta is defined by the BRP field. |
| 11:8 | TSEG1 | RW | 0x3 | Time Segment Before Sample Point<br>0x00-0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>So, for example, the reset value of 0x3 means that 4 (3+1) bit time quanta are defined for Phase1 (see Figure 23-4 on page 1552). The bit time quanta is defined by the BRP field. |
| 7:6 | SJW | RW | 0x0 | (Re)Synchronization Jump Width<br>0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of TSEG2 or TSEG1 by the value in SJW. So the reset value of 0 adjusts the length by 1 bit time quanta. |
| 5:0 | BRP | RW | 0x1 | Baud Rate Prescaler<br>The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum.<br>0x00-0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.<br>BRP defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1).<br>The **CANBRPE** register can be used to further divide the bit time. |

## Register 5: CAN Interrupt (CANINT), offset 0x010

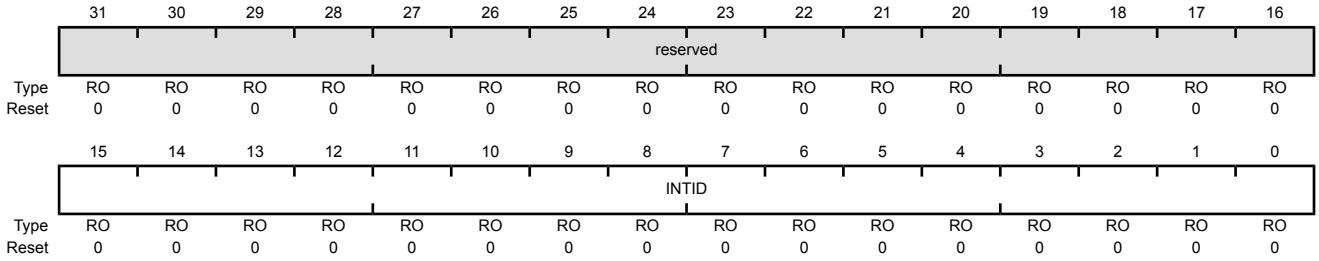
This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the `INTID` field is not 0x0000 (the default) and the `IE` bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the `INTID` field is cleared by reading the **CANSTS** register, or until the `IE` bit in the **CANCTL** register is cleared.

**Note:** Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x010
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | INTID | RO | 0x0000 | Interrupt Identifier<br>The number in this field indicates the source of the interrupt. |

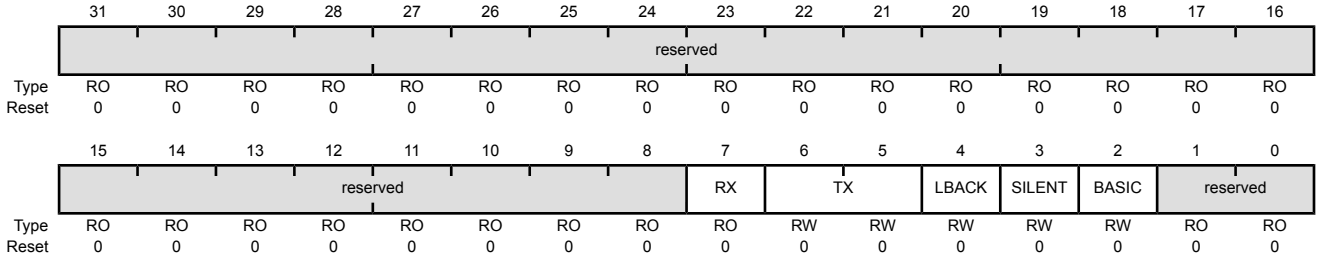
| Value | Description |
|---|---|
| 0x0000 | No interrupt pending |
| 0x0001-0x0020 | Number of the message object that caused the interrupt |
| 0x0021-0x7FFF | Reserved |
| 0x8000 | Status Interrupt |
| 0x8001-0xFFFF | Reserved |

## Register 6: CAN Test (CANTST), offset 0x014

This register is used for self-test and external pin access. It is write-enabled by setting the TEST bit in the **CANCTL** register. Different test functions may be combined, however, CAN transfers are affected if the TX bits in this register are not zero.

CAN Test (CANTST)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x014
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | RX | RO | 0 | Receive Observation<br>Value: 0 — The CANnRx pin is low.<br>Value: 1 — The CANnRx pin is high. |
| 6:5 | TX | RW | 0x0 | Transmit Control<br>Overrides control of the CANnTx pin.<br>Value: 0x0 — CAN Module Control. CANnTx is controlled by the CAN module; default operation<br>Value: 0x1 — Sample Point. The sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing.<br>Value: 0x2 — Driven Low. CANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus.<br>Value: 0x3 — Driven High. CANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus. |

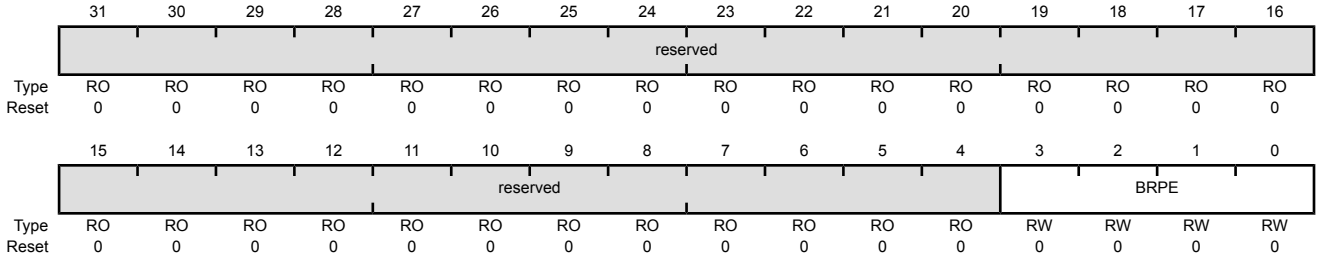
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | LBACK | RW | 0 | Loopback Mode<br>Value: 0 — Loopback mode is disabled.<br>1 — Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored. |
| 3 | SILENT | RW | 0 | Silent Mode<br>Value: 0 — Silent mode is disabled.<br>1 — Silent mode is enabled. In silent mode, the CAN controller does not transmit data but instead monitors the bus. This mode is also known as Bus Monitor mode. |
| 2 | BASIC | RW | 0 | Basic Mode<br>Value: 0 — Basic mode is disabled.<br>1 — Basic mode is enabled. In basic mode, software should use the **CANIF1** registers as the transmit buffer and use the **CANIF2** registers as the receive buffer. |
| 1:0 | reserved | RO | 0x0 | 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. |

## Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018

This register is used to further divide the bit time set with the BRP bit in the **CANBIT** register. It is write-enabled by setting the CCE bit in the **CANCTL** register.

CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x018
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3:0 | BRPE | RW | 0x0 | Baud Rate Prescaler Extension<br>0x00-0x0F: Extend the BRP bit in the **CANBIT** register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by BRPE (MSBs) and BRP (LSBs). |

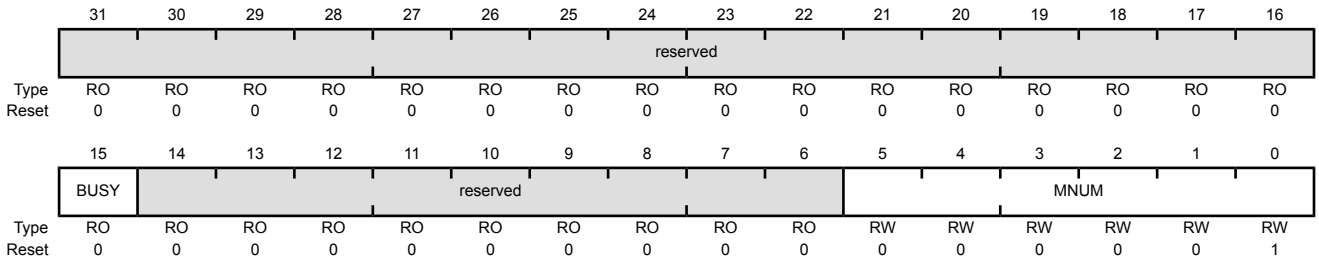
## Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

## Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080

A message transfer is started as soon as there is a write of the message object number to the MNUM field when the TXRQST bit in the **CANIF1MCTL** register is set. With this write operation, the BUSY bit is automatically set to indicate that a transfer between the CAN Interface Registers and the internal message RAM is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the interface register and the message RAM completes, which then clears the BUSY bit.

CAN IFn Command Request (CANIFnCRQ)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x020
Type RW, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15 | BUSY | RO | 0 | Busy Flag |

| Value | Description |
|---|---|
| 0 | This bit is cleared when read/write action has finished. |
| 1 | This bit is set when a write occurs to the message number in this register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 14:6 | reserved | RO | 0x00 | 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. |
| 5:0 | MNUM | RW | 0x01 | Message Number<br>Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32. |

| Value | Description |
|---|---|
| 0x00 | Reserved<br>0 is not a valid message number; it is interpreted as 0x20, or object 32. |
| 0x01-0x20 | Message Number<br>Indicates specified message object 1 to 32. |
| 0x21-0x3F | Reserved<br>Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F. |

## Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024

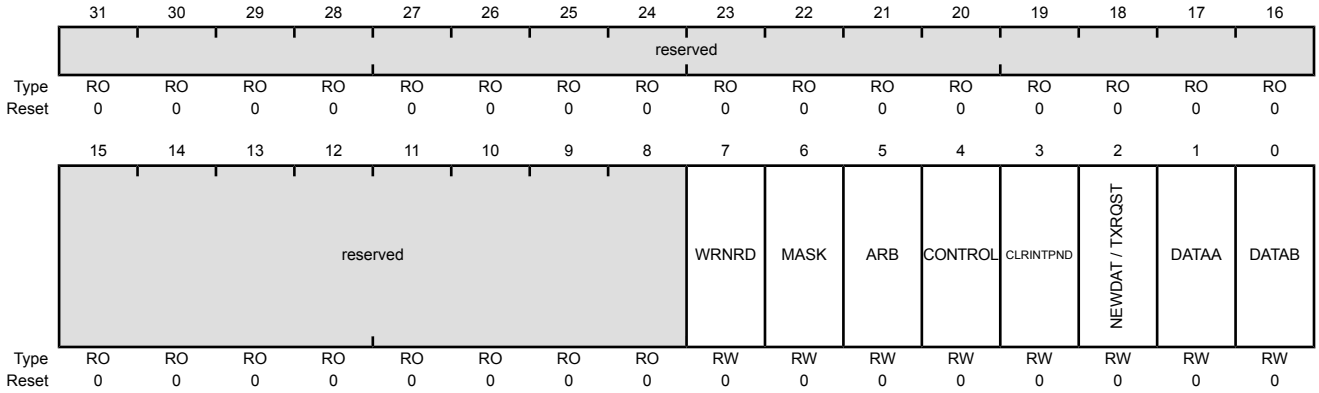
## Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084

Reading the Command Mask registers provides status for various functions. Writing to the Command Mask registers specifies the transfer direction and selects which buffer registers are the source or target of the data transfer.

Note that when a read from the message object buffer occurs when the WRNRD bit is clear and the CLRINTPND and/or NEWDAT bits are set, the interrupt pending and/or new data flags in the message object buffer are cleared.

CAN IFn Command Mask (CANIFnCMSK)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x024
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | WRNRD | RW | 0 | Write, Not Read |

| Value | Description |
|---|---|
| 0 | Transfer the data in the CAN message object specified by the MNUM field in the **CANIFnCRQ** register into the **CANIFn** registers. |
| 1 | Transfer the data in the **CANIFn** registers to the CAN message object specified by the MNUM field in the **CAN Command Request (CANIFnCRQ)**. |

**Note:** Interrupt pending and new data conditions in the message buffer can be cleared by reading from the buffer (WRNRD = 0) when the CLRINTPND and/or NEWDAT bits are set.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | MASK | RW | 0 | Access Mask Bits |

| Value | Description |
|---|---|
| 0 | Mask bits unchanged. |
| 1 | Transfer IDMASK + DIR + MXTD of the message object into the Interface registers. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | ARB | RW | 0 | Access Arbitration Bits<br>Value: 0 – Arbitration bits unchanged.<br>1 – Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers. |
| 4 | CONTROL | RW | 0 | Access Control Bits<br>Value: 0 – Control bits unchanged.<br>1 – Transfer control bits from the **CANIFnMCTL** register into the Interface registers. |
| 3 | CLRINTPND | RW | 0 | Clear Interrupt Pending Bit<br>The function of this bit depends on the configuration of the WRNRD bit.<br>Value: 0 – If WRNRD is clear, the interrupt pending status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is set, the INTPND bit in the message object remains unchanged.<br>1 – If WRNRD is clear, the interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing. If WRNRD is set, the INTPND bit is cleared in the message object. |
| 2 | NEWDAT / TXRQST | RW | 0 | NEWDAT / TXRQST Bit<br>The function of this bit depends on the configuration of the WRNRD bit.<br>Value: 0 – If WRNRD is clear, the value of the new data status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is set, a transmission is not requested.<br>1 – If WRNRD is clear, the new data status is cleared in the message buffer. Note the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing. If WRNRD is set, a transmission is requested. Note that when this bit is set, the TXRQST bit in the **CANIFnMCTL** register is ignored. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | DATAA | RW | 0 | Access Data Byte 0 to 3<br>The function of this bit depends on the configuration of the WRNRD bit.<br><br>Value — Description<br>0 — Data bytes 0-3 are unchanged.<br>1 — If WRNRD is clear, transfer data bytes 0-3 in **CANIFnDA1** and **CANIFnDA2** to the message object. If WRNRD is set, transfer data bytes 0-3 in message object to **CANIFnDA1** and **CANIFnDA2**. |
| 0 | DATAB | RW | 0 | Access Data Byte 4 to 7<br>The function of this bit depends on the configuration of the WRNRD bit as follows:<br><br>Value — Description<br>0 — Data bytes 4-7 are unchanged.<br>1 — If WRNRD is clear, transfer data bytes 4-7 in **CANIFnDA1** and **CANIFnDA2** to the message object. If WRNRD is set, transfer data bytes 4-7 in message object to **CANIFnDA1** and **CANIFnDA2**. |

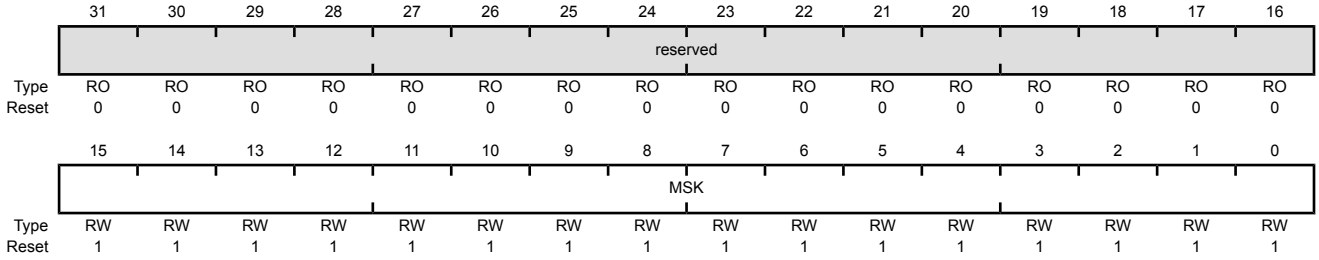
## Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

## Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the ID bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

CAN IFn Mask 1 (CANIFnMSK1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x028
Type RW, reset 0x0000.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | MSK | RW | 0xFFFF | Identifier Mask<br>When using a 29-bit identifier, these bits are used for bits [15:0] of the ID. The MSK field in the **CANIFnMSK2** register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored. |

| Value | Description |
|---|---|
| 0 | The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering. |
| 1 | The corresponding identifier field (ID) is used for acceptance filtering. |

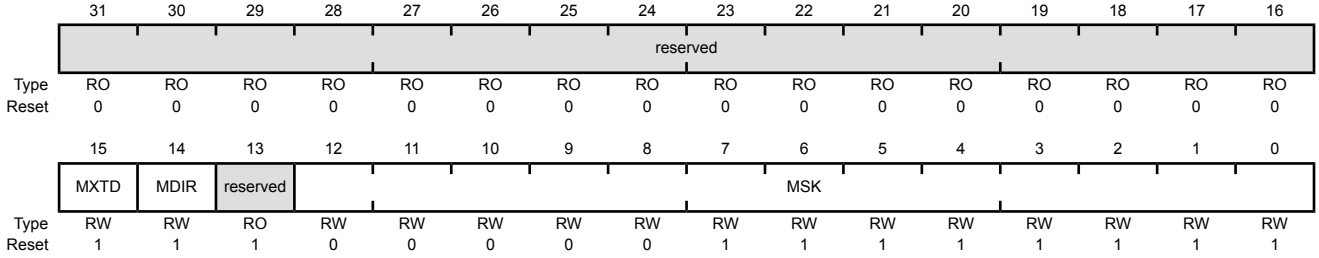
## Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C

## Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C

This register holds extended mask information that accompanies the **CANIFnMSK1** register.

CAN IFn Mask 2 (CANIFnMSK2)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x02C
Type RW, reset 0x0000.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15 | MXTD | RW | 1 | Mask Extended Identifier<br><br>Value Description<br>0 The extended identifier bit (XTD in the **CANIFnARB2** register) has no effect on the acceptance filtering.<br>1 The extended identifier bit XTD is used for acceptance filtering. |
| 14 | MDIR | RW | 1 | Mask Message Direction<br><br>Value Description<br>0 The message direction bit (DIR in the **CANIFnARB2** register) has no effect for acceptance filtering.<br>1 The message direction bit DIR is used for acceptance filtering. |
| 13 | reserved | RO | 1 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12:0 | MSK | RW | 0xFF | Identifier Mask<br>When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The MSK field in the **CANIFnMSK1** register are used for bits [15:0] of the ID. When using an 11-bit identifier, MSK[12:2] are used for bits [10:0] of the ID.<br>Value: 0 — The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering.<br>Value: 1 — The corresponding identifier field (ID) is used for acceptance filtering. |

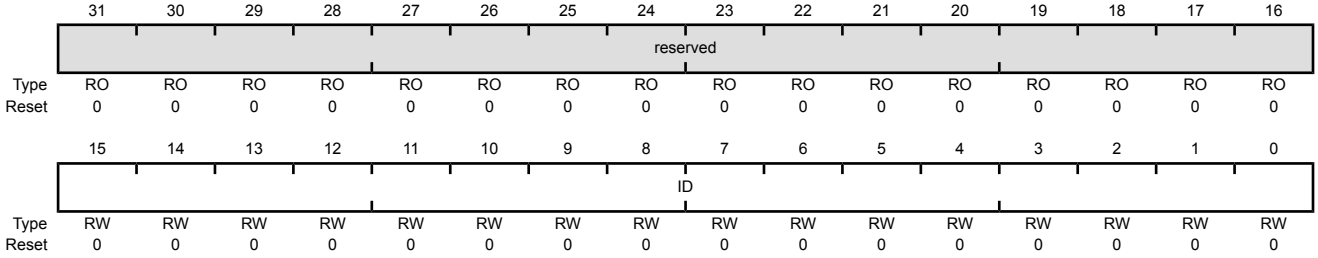
## Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030

## Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090

These registers hold the identifiers for acceptance filtering.

CAN IFn Arbitration 1 (CANIFnARB1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x030
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | ID | RW | 0x0000 | Message Identifier<br>This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier.<br>When using a 29-bit identifier, bits 15:0 of the **CANIFnARB1** register are [15:0] of the ID, while bits 12:0 of the **CANIFnARB2** register are [28:16] of the ID.<br>When using an 11-bit identifier, these bits are not used. |

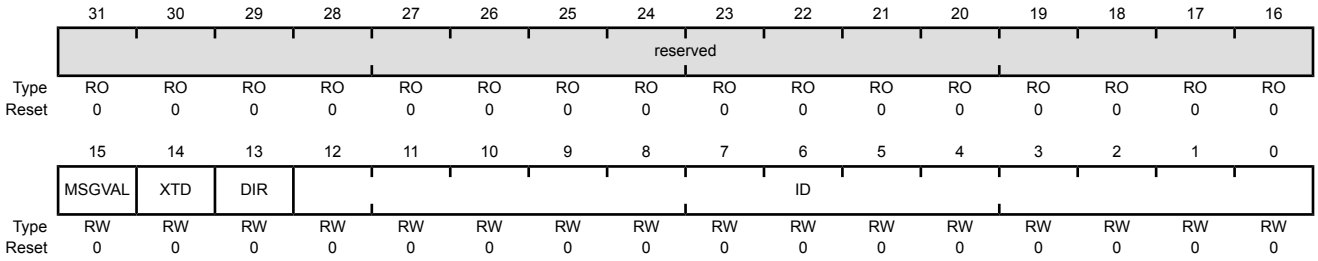
## Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034

## Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094

These registers hold information for acceptance filtering.

CAN IFn Arbitration 2 (CANIFnARB2)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x034
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15 | MSGVAL | RW | 0 | Message Valid<br><br>Value / Description<br>0 – The message object is ignored by the message handler.<br>1 – The message object is configured and ready to be considered by the message handler within the CAN controller.<br><br>All unused message objects should have this bit cleared during initialization and before clearing the `INIT` bit in the **CANCTL** register. The `MSGVAL` bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the `ID` fields in the **CANIFnARBn** registers, the `XTD` and `DIR` bits in the **CANIFnARB2** register, or the `DLC` field in the **CANIFnMCTL** register. |
| 14 | XTD | RW | 0 | Extended Identifier<br><br>Value / Description<br>0 – An 11-bit Standard Identifier is used for this message object.<br>1 – A 29-bit Extended Identifier is used for this message object. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | DIR | RW | 0 | Message Direction<br><br>Value Description<br>0 Receive. When the TXRQST bit in the **CANIFnMCTL** register is set, a remote frame with the identifier of this message object is received. On reception of a data frame with matching identifier, that message is stored in this message object.<br>1 Transmit. When the TXRQST bit in the **CANIFnMCTL** register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TXRQST bit of this message object is set (if RMTEN=1). |
| 12:0 | ID | RW | 0x000 | Message Identifier<br><br>This bit field is used with the ID field in the **CANIFnARB2** register to create the message identifier.<br><br>When using a 29-bit identifier, ID[15:0] of the **CANIFnARB1** register are [15:0] of the ID, while these bits, ID[12:0], are [28:16] of the ID.<br><br>When using an 11-bit identifier, ID[12:2] are used for bits [10:0] of the ID. The ID field in the **CANIFnARB1** register is ignored. |

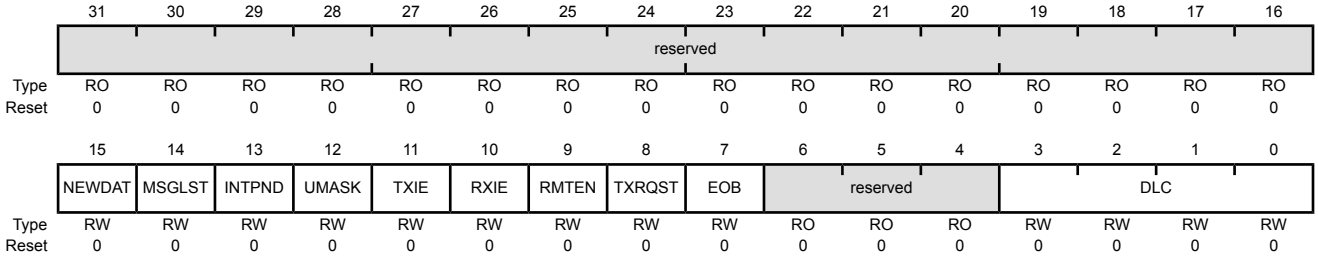
## Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038

## Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098

This register holds the control information associated with the message object to be sent to the Message RAM.

CAN IFn Message Control (CANIFnMCTL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x038
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15 | NEWDAT | RW | 0 | New Data<br>Value: 0 – No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU.<br>1 – The message handler or the CPU has written new data into the data portion of this message object. |
| 14 | MSGLST | RW | 0 | Message Lost<br>Value: 0 – No message was lost since the last time this bit was cleared by the CPU.<br>1 – The message handler stored a new message into this object when NEWDAT was set; the CPU has lost a message.<br>This bit is only valid for message objects when the DIR bit in the **CANIFnARB2** register is clear (receive). |
| 13 | INTPND | RW | 0 | Interrupt Pending<br>Value: 0 – This message object is not the source of an interrupt.<br>1 – This message object is the source of an interrupt. The interrupt identifier in the **CANINT** register points to this message object if there is not another interrupt source with a higher priority. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | UMASK | RW | 0 | Use Acceptance Mask |

| Value | Description |
|---|---|
| 0 | Mask is ignored. |
| 1 | Use mask (MSK, MXTD, and MDIR bits in the **CANIFnMSKn** registers) for acceptance filtering. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | TXIE | RW | 0 | Transmit Interrupt Enable |

| Value | Description |
|---|---|
| 0 | The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful transmission of a frame. |
| 1 | The INTPND bit in the **CANIFnMCTL** register is set after a successful transmission of a frame. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | RXIE | RW | 0 | Receive Interrupt Enable |

| Value | Description |
|---|---|
| 0 | The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful reception of a frame. |
| 1 | The INTPND bit in the **CANIFnMCTL** register is set after a successful reception of a frame. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RMTEN | RW | 0 | Remote Enable |

| Value | Description |
|---|---|
| 0 | At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is left unchanged. |
| 1 | At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TXRQST | RW | 0 | Transmit Request |

| Value | Description |
|---|---|
| 0 | This message object is not waiting for transmission. |
| 1 | The transmission of this message object is requested and is not yet done. |

**Note:** If the WRNRD and TXRQST bits in the **CANIFnCMSK** register are set, this bit is ignored.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | EOB | RW | 0 | End of Buffer<br><br>Value: 0 — Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.<br>Value: 1 — Single message object or last message object of a FIFO Buffer.<br><br>This bit is used to concatenate two or more message objects (up to 32) to build a FIFO buffer. For a single message object (thus not belonging to a FIFO buffer), this bit must be set. |
| 6:4 | reserved | RO | 0x0 | 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. |
| 3:0 | DLC | RW | 0x0 | Data Length Code<br><br>Value: 0x0-0x8 — Specifies the number of bytes in the data frame.<br>Value: 0x9-0xF — Defaults to a data frame with 8 bytes.<br><br>The DLC field in the **CANIFnMCTL** register of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it writes DLC to the value given by the received message. |

### Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C

### Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040

### Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044

### Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048

### Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C

### Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0

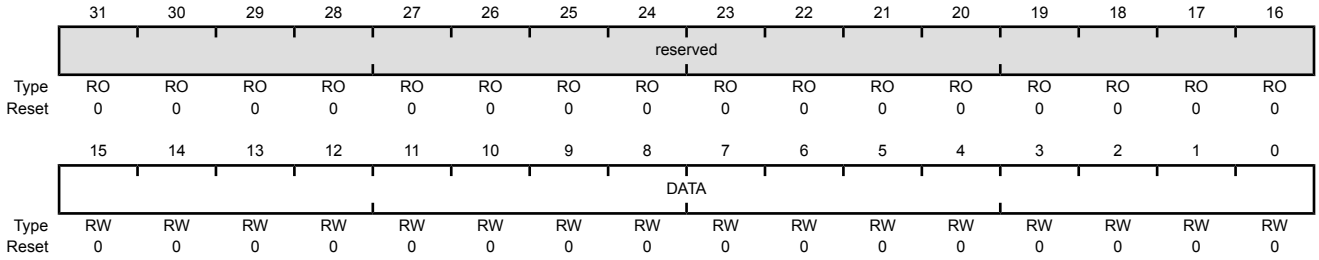
### Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4

### Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8

These registers contain the data to be sent or that has been received. In a CAN data frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

CAN IFn Data nn (CANIFnDnn)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x03C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | DATA | RW | 0x0000 | Data<br>The **CANIFnDA1** registers contain data bytes 1 and 0; **CANIFnDA2** data bytes 3 and 2; **CANIFnDB1** data bytes 5 and 4; and **CANIFnDB2** data bytes 7 and 6. |

## Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100

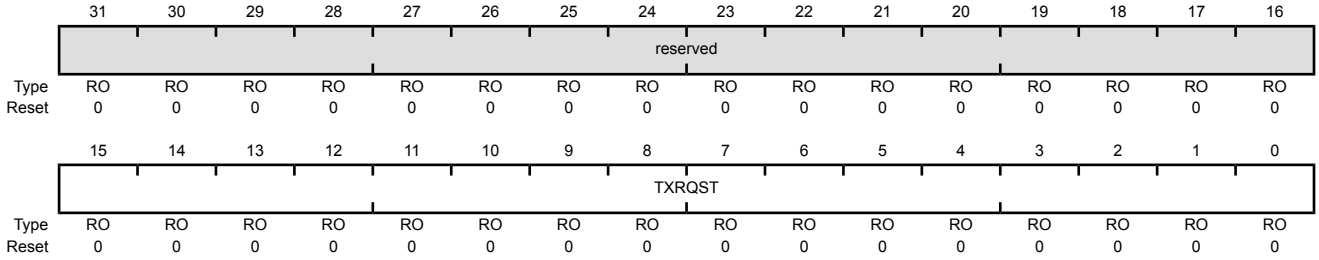
## Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104

The **CANTXRQ1** and **CANTXRQ2** registers hold the TXRQST bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TXRQST bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a remote frame, or (3) the message handler state machine after a successful transmission.

The **CANTXRQ1** register contains the TXRQST bits of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TXRQST bits of the second 16 message objects.

CAN Transmission Request n (CANTXRQn)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x100
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | TXRQST | RO | 0x0000 | Transmission Request Bits |

| Value | Description |
|---|---|
| 0 | The corresponding message object is not waiting for transmission. |
| 1 | The transmission of the corresponding message object is requested and is not yet done. |

## Register 32: CAN New Data 1 (CANNWDA1), offset 0x120

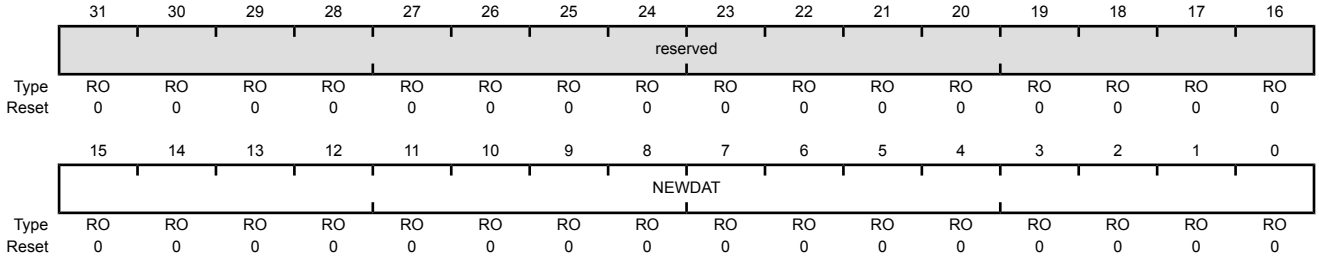
## Register 33: CAN New Data 2 (CANNWDA2), offset 0x124

The **CANNWDA1** and **CANNWDA2** registers hold the NEWDAT bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The NEWDAT bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a data frame, or (3) the message handler state machine after a successful transmission.

The **CANNWDA1** register contains the NEWDAT bits of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the NEWDAT bits of the second 16 message objects.

CAN New Data n (CANNWDAn)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x120
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | NEWDAT | RO | 0x0000 | New Data Bits |

| Value | Description |
|---|---|
| 0 | No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU. |
| 1 | The message handler or the CPU has written new data into the data portion of the corresponding message object. |

## Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

## Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

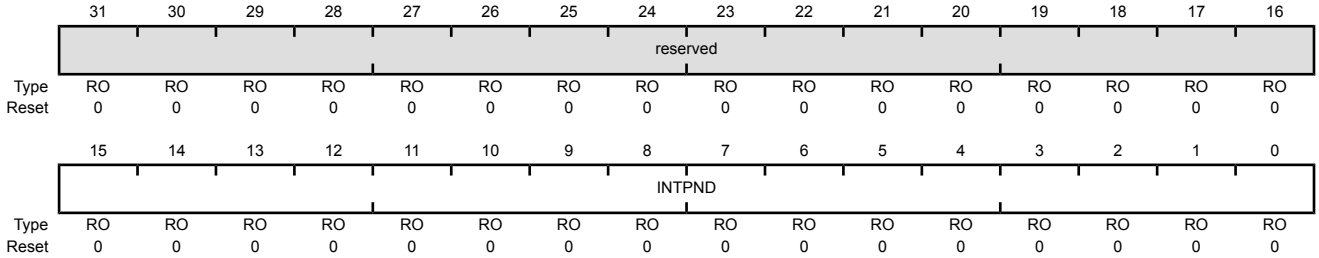
The **CANMSG1INT** and **CANMSG2INT** registers hold the INTPND bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The INTPND bit of a specific message object can be changed through two sources: (1) the CPU via the **CANIFnMCTL** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CANINT** register.

The **CANMSG1INT** register contains the INTPND bits of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the INTPND bits of the second 16 message objects.

CAN Message n Interrupt Pending (CANMSGnINT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x140
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | INTPND | RO | 0x0000 | Interrupt Pending Bits |

| Value | Description |
|---|---|
| 0 | The corresponding message object is not the source of an interrupt. |
| 1 | The corresponding message object is the source of an interrupt. |

## Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160

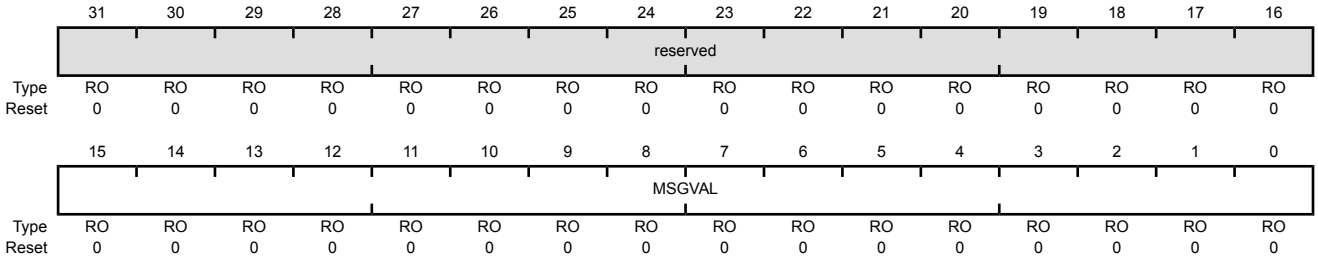
## Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the MSGVAL bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message valid bit of a specific message object can be changed with the **CANIFnARB2** register.

The **CANMSG1VAL** register contains the MSGVAL bits of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the MSGVAL bits of the second 16 message objects in the message RAM.

CAN Message n Valid (CANMSGnVAL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x160
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | MSGVAL | RO | 0x0000 | Message Valid Bits |

| Value | Description |
|---|---|
| 0 | The corresponding message object is not configured and is ignored by the message handler. |
| 1 | The corresponding message object is configured and should be considered by the message handler. |

# 24 Ethernet Controller

The Ethernet Controller has the following features:

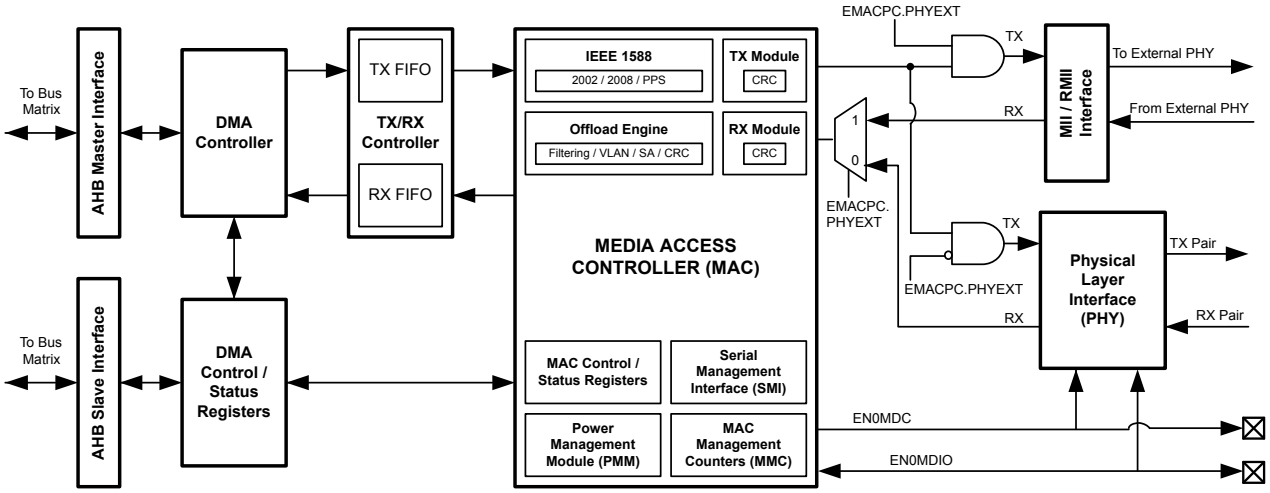
- Conforms to the IEEE 802.3 specification
  - 10BASE-T/100BASE-TX IEEE-802.3 compliant
  - Supports 10/100 Mbps data transmission rates
  - Supports full-duplex and half-duplex (CSMA/CD) operation
  - Supports flow control and back pressure
  - Full-featured and enhanced auto-negotiation
  - Supports IEEE 802.1Q VLAN tag detection
- Conforms to IEEE 1588-2002 Timestamp Precision Time Protocol (PTP) protocol and the IEEE 1588-2008 Advanced Timestamp specification
  - Transmit and Receive frame time stamping
  - Precision Time Protocol
  - Flexible pulse per second output
  - Supports coarse and fine correction methods
- Multiple addressing modes
  - Four MAC address filters
  - Programmable 64-bit Hash Filter for multicast address filtering
  - Promiscuous mode support
- Processor offloading
  - Programmable insertion (TX) or deletion (RX) of preamble and start-of-frame data
  - Programmable generation (TX) or deletion (RX) of CRC and pad data
  - IP header and hardware checksum checking (IPv4, IPv6, TCP/UDP/ICMP)
- Highly configurable
  - LED activity selection
  - Supports network statistics with RMON/MIB counters
  - Supports Magic Packet and wakeup frames
- Efficient transfers using integrated Direct Memory Access (DMA)
  - Dual-buffer (ring) or linked-list (chained) descriptors

  - Round-robin or fixed priority arbitration between TX/RX
  - Descriptors support up to 8 kB transfer blocks size
  - Programmable interrupts for flexible system implementation
- Physical media manipulation
  - MDI/MDI-X cross-over support
  - Register-programmable transmit amplitude
  - Automatic polarity correction and 10BASE-T signal reception
- MII and RMII interface support

## 24.1 Block Diagram

Figure 24-1 on page 1589 shows the block diagram of the Ethernet MAC with an integrated PHY interface:

**Figure 24-1. Ethernet MAC with Integrated PHY Interface**




## 24.2 Signal Description

The following table lists the external signals of the Ethernet Controller and describes the function of each. Some of the Ethernet signals have dedicated functions while the others listed in the table are alternate functions for GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these Ethernet module signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to assign the Ethernet signals. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the Ethernet signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

### Table 24-1. Ethernet Signals (212BGA)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| EN0COL | N18 | PM7 (14) | I | TTL | Ethernet 0 Collision Detect. |
| EN0CRS | N19 | PM6 (14) | I | TTL | Ethernet 0 Carrier Sense. |
| EN0INTRN | U19<br>D6 | PK4 (7)<br>PP0 (7) | I | TTL | Ethernet 0 Interrupt from the Ethernet PHY. |
| EN0LED0 | U6<br>U19 | PF0 (5)<br>PK4 (5) | O | TTL | Ethernet 0 LED 0. |
| EN0LED1 | V7<br>V16 | PF4 (5)<br>PK6 (5) | O | TTL | Ethernet 0 LED 1. |
| EN0LED2 | V6<br>V17 | PF1 (5)<br>PK5 (5) | O | TTL | Ethernet 0 LED 2. |
| EN0MDC | A17<br>W6 | PB2 (5)<br>PF2 (5) | O | TTL | Ethernet 0 Management Data Clock. |
| EN0MDIO | B17<br>T7 | PB3 (5)<br>PF3 (5) | I/O | TTL | Ethernet 0 Management Data Input/Output signal. |
| EN0PPS | N15<br>T2<br>C8 | PG0 (5)<br>PH5 (5)<br>PJ0 (5) | O | TTL | Ethernet 0 Pulse-Per-Second (PPS) Output. |
| EN0RREF_CLK | M18 | PM4 (14) | I/O | TTL | Ethernet 0 Reference Clock. |
| EN0RXCK | V5 | PA6 (14) | I | TTL | Ethernet 0 Receive Clock. |
| EN0RXD0 | W12<br>W10 | PQ5 (14)<br>PT0 (14) | I | TTL | Ethernet 0 Receive Data 0. |
| EN0RXD1 | U15<br>V10 | PQ6 (14)<br>PT1 (14) | I | TTL | Ethernet 0 Receive Data 1. |
| EN0RXD2 | V17 | PK5 (14) | I | TTL | Ethernet 0 Receive Data 2. |
| EN0RXD3 | U19 | PK4 (14) | I | TTL | Ethernet 0 Receive Data 3. |
| EN0RXDV | U14<br>R13 | PG7 (14)<br>PS7 (14) | I | TTL | Ethernet 0 Receive Data Valid. |
| EN0RXER | V12<br>U10 | PG6 (14)<br>PS6 (14) | I | TTL | Ethernet 0 Receive Error. |
| EN0RXIN | V13 | fixed | I/O | TTL | Ethernet PHY negative receive differential input. |
| EN0RXIP | W13 | fixed | I/O | TTL | Ethernet PHY positive receive differential input. |
| EN0TXCK | V11 | PG2 (14) | I | TTL | Ethernet 0 Transmit Clock. |
| EN0TXD0 | K17<br>V9 | PG4 (14)<br>PS4 (14) | O | TTL | Ethernet 0 Transmit Data 0. |
| EN0TXD1 | K15<br>T13 | PG5 (14)<br>PS5 (14) | O | TTL | Ethernet 0 Transmit Data 1. |
| EN0TXD2 | V16 | PK6 (14) | O | TTL | Ethernet 0 Transmit Data 2. |
| EN0TXD3 | W16 | PK7 (14) | O | TTL | Ethernet 0 Transmit Data 3. |
| EN0TXEN | M16<br>R10 | PG3 (14)<br>PR7 (14) | O | TTL | Ethernet 0 Transmit Enable. |
| EN0TXER | T12 | PN6 (14) | O | TTL | Ethernet 0 transmit error. |
| EN0TXON | V14 | fixed | I/O | TTL | Ethernet PHY negative transmit differential output. |
| EN0TXOP | V15 | fixed | I/O | TTL | Ethernet PHY positive transmit differential output. |
| RBIAS | W15 | fixed | O | Analog | 4.87-kΩ resistor (1% precision) for Ethernet PHY. |

## 24.3 Functional Description

The Ethernet Controller is made up of the following sub-modules:

- Clock Control
- MII/RMII Interface Module
- DMA Controller
- Transmit/Receive Controller (TX/RX Controller)
- Media Access Controller (MAC)
- AHB Bus Interface
- PHY Interface

The following sections describe the features and functions of each sub-module.

### 24.3.1 Ethernet Clock Control

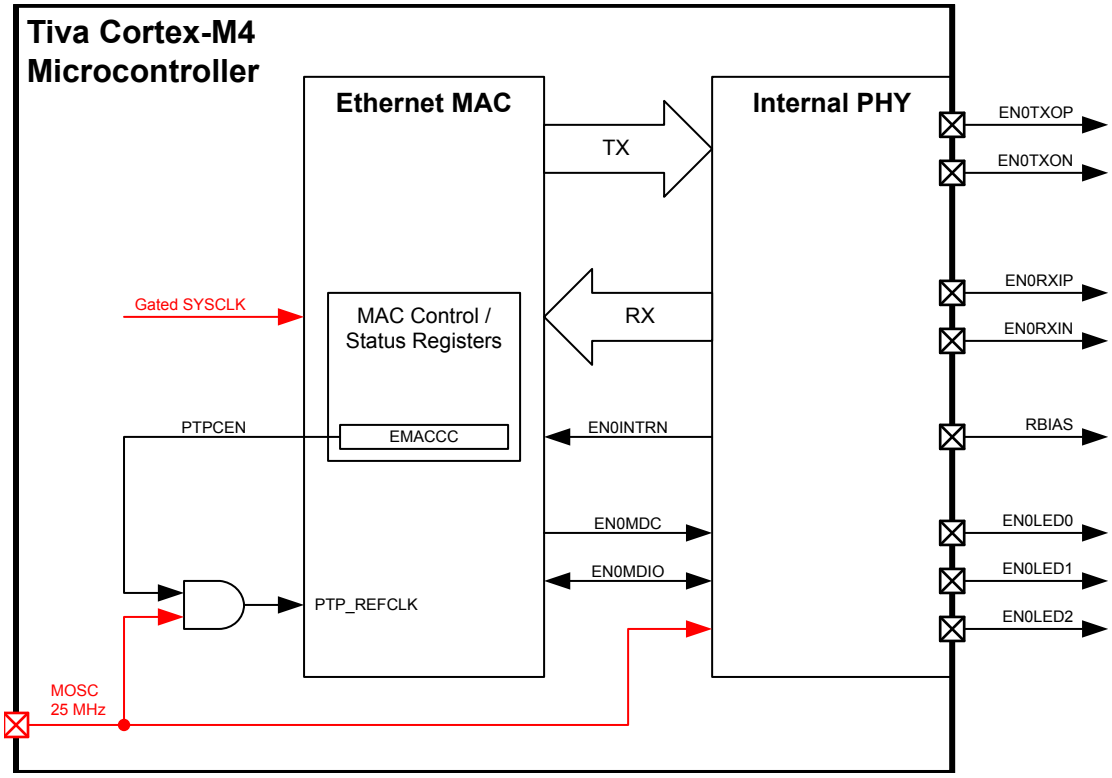
Available clock sources are dependent on the interface chosen. The following sections describe the clock control for the various interfaces.

#### 24.3.1.1 PHY Interface

The Ethernet Controller Module and Integrated PHY receive two clock inputs:

- A gated system clock acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SYSCLK frequency for Run, Sleep and Deep Sleep mode is programmed in the System Control module. Refer to “System Control” on page 231 for more information on programming SYSCLK and enabling the Ethernet MAC.
- The PHY receives the main oscillator (MOSC) which must be 25 MHz ± 50 ppm for proper operation. The MOSC source can be a single-ended source or a crystal. Figure 24-2 on page 1592 shows the clock inputs to the Ethernet Controller Module.

**Figure 24-2. Ethernet MAC and PHY Clock Structure**



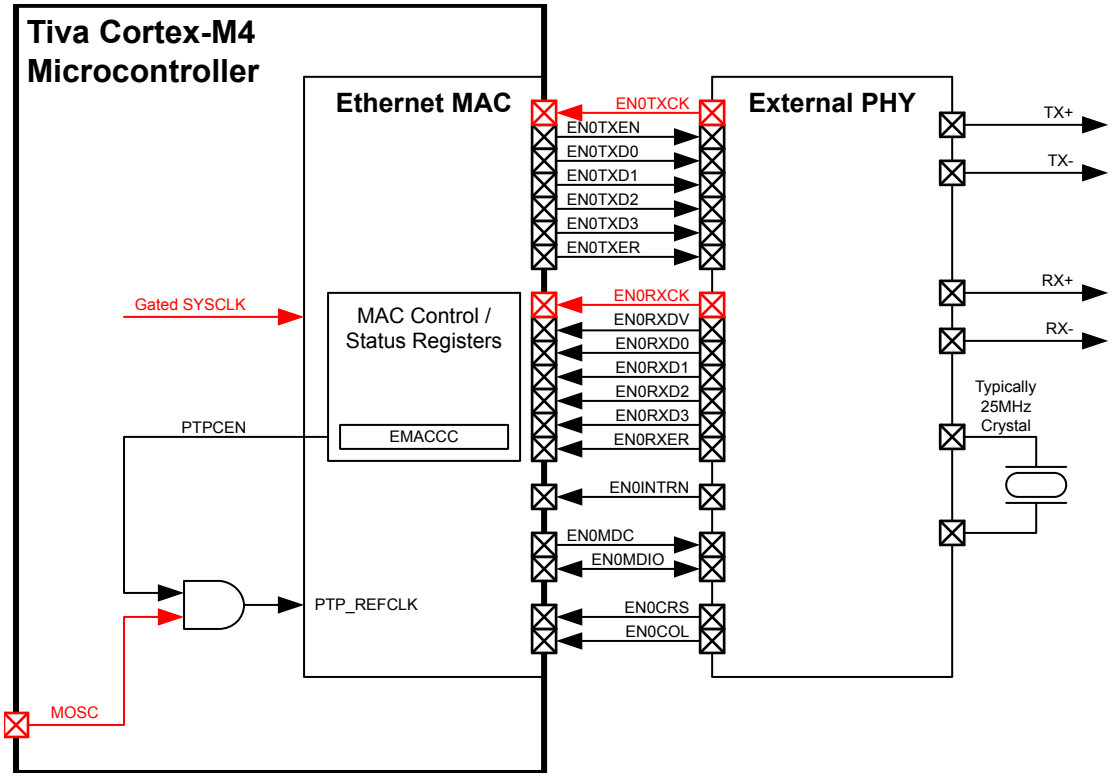

### 24.3.1.2 MII Interface

Four clock inputs are driven into the Ethernet MAC when the MII configuration is enabled. The clocks are described as follows:

- Gated system clock (SYSCLK): The SYSCLK signal acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SYSCLK frequency for Run, Sleep and Deep Sleep mode is programmed in the System Control module. Refer to “System Control” on page 231 for more information on programming SYSCLK and enabling the Ethernet MAC.
- MOSC: A gated version of the MOSC clock is provided as the Precision Time Protocol (PTP) reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the Precision Timer Protocol (PTP) module has been enabled by setting the PTPCEN bit in the **EMACCC** register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. Refer to “IEEE 1588 and Advanced Timestamp Function” on page 1625 for more information.
- EN0RXCK: This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz depending on whether the device is operating at 10 Mbps or 100 Mbps.
- EN0TXCK This clock signal is driven by the external PHY oscillator and is either 2.5 or 25 MHz depending on whether the device is operating at 10 Mbps or 100 Mbps.

Figure 24-3 on page 1593 depicts the clock inputs for an MII interface.

**Figure 24-3. MII Clock Structure**



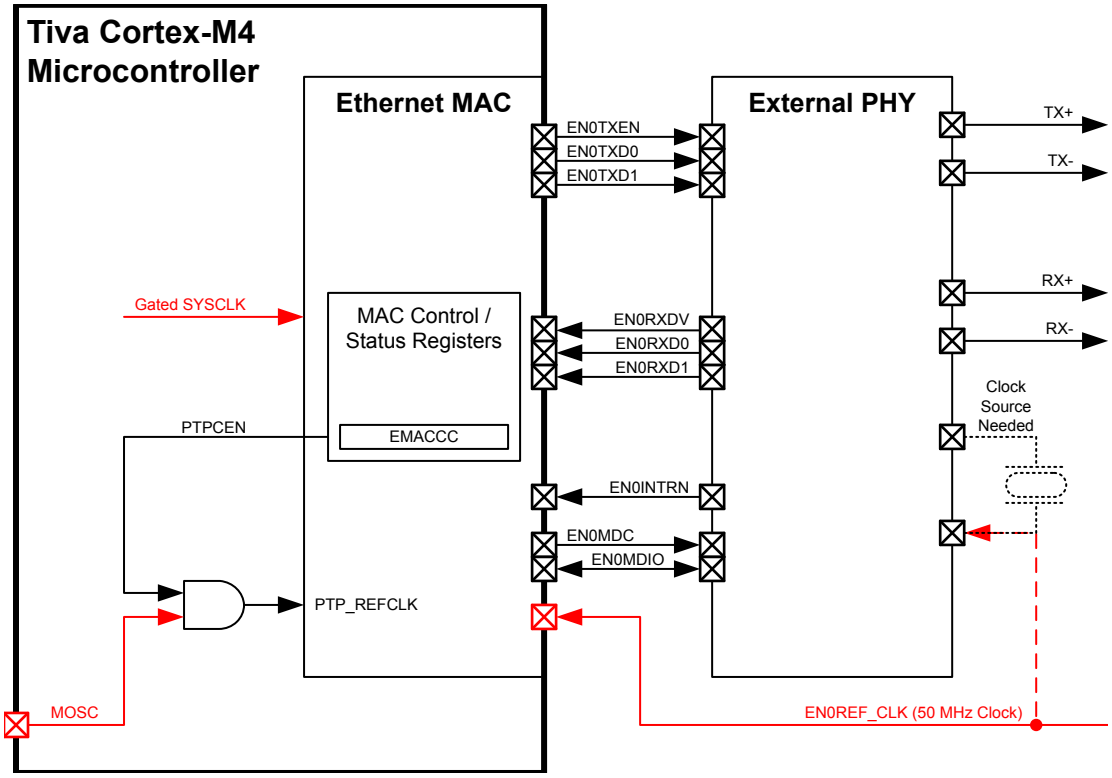

### 24.3.1.3 RMII Interface

There are three clock sources that interface to the Ethernet MAC in an RMII configuration:

- Gated system clock (SYSCLK): The SYSCLK signal acts as the clock source to the Control and Status registers (CSR) of the Ethernet MAC. The SYSCLK frequency for Run, Sleep and Deep Sleep mode is programmed in the System Control module. Refer to "System Control" on page 231 for more information on programming SYSCLK and enabling the Ethernet MAC.
- MOSC: A gated version of the MOSC clock is provided as the Precision Time Protocol (PTP) reference clock (PTPREF_CLK). The MOSC clock source can be a single-ended source on the OSC0 pin or a crystal on the OSC0 and OSC1 pins. When advanced timestamping is used and the PTP module has been enabled by setting the PTPCEN bit in the **EMACCC** register, the MOSC drives PTPREF_CLK. PTPREF_CLK has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. Refer to "IEEE 1588 and Advanced Timestamp Function" on page 1625 for more information.
- EN0REF_CLK: When using RMII, a 50 MHz external reference clock must drive the EN0REF_CLK input signal and the external PHY. Depending on the configuration of the FES bit in the **Ethernet MAC Configuration (EMACCFG)** register, the reference clock input (EN0REF_CLK) is divided by 20 for 10 Mbps or 2 for 100 Mbps operation and used as the clock for receive and transmit data.

Figure 24-4 on page 1594 depicts the clock inputs to the RMII clock interface.

**Figure 24-4. RMII Clock Structure**




## 24.3.2 MII/RMII Interface Signals

The MAC Module has the capability of providing an MII or RMII interface depending on the interface selected in the **Ethernet MAC Peripheral Configuration (EMACPC)** register, at offset 0xFC4. Except for EN0REF_CLK, the signals used for RMII mode are a subset of the MII interface signals. Thus, Table 24-2 on page 1594 details the MII signals that are used in RMII mode and the RMII function to which they correspond. In addition, the GPIO pin they are muxed with is listed.

**Table 24-2. MII and RMII Interface Signals**

| MII Signal | RMII | RMII Standard Name and Function | GPIO |
|---|---|---|---|
| N/A | EN0REF_CLK | REF_CLK: Synchronous clock reference for receive, transmit and control | PM4 |
| EN0TXCK | Not Used | N/A | PG2 |
| EN0TD3 | Not Used | N/A | PK7 |
| EN0TXD2 | Not Used | N/A | PK6 |
| EN0TXD1 | EN0TXD1 | TXD1: Transmit Data 1 | PG5/PS5 |
| EN0TXD0 | EN0TXD0 | TXD0: Transmit Data 0 | PG4/PS4 |
| EN0TXEN | EN0TXEN | TEX_EN: Transmit Enable | PG3/PR7 |
| EN0TXER | Not Used | N/A | PN6 |
| EN0RXCK | Not Used | N/A | PA6 |
| EN0RXD3 | Not Used | N/A | PK4 |
| EN0RXD2 | Not Used | N/A | PK5 |
| EN0RXD1 | EN0RXD1 | RXD1: Receive Data 1 | PQ6/PT1 |

**Table 24-2. MII and RMII Interface Signals *(continued)***

| MII Signal | RMII | RMII Standard Name and Function | GPIO |
|---|---|---|---|
| EN0RXD0 | EN0RXD0 | RXD0: Receive Data 0 | PQ5/PT0 |
| EN0RXDV | EN0RXDV | CRS_DV: Carrier Sense/Receive Data Valid | PG7/PS7 |
| EN0RXER | Not Used | RX_ER: Receive Error[a] | PG6/PS6 |
| EN0COL | Not Used | N/A | PM7 |
| EN0CRS | Not Used | N/A | PM6 |
| EN0MDC | EN0MDC | MDC: Management Data Clock | PB2/PF2 |
| EN0MDIO | EN0MDIO | MDIO: Management Data Input/Output | PB3/PF3 |
| EN0PPS | EN0PPS | Pulse-Per-Second (PPS) Output (optional-this is not a standard RMII signal) | PG0/PJ0/PH5 |
| EN0INTRN | EN0INTRN | Interrupt to Ethernet PHY (optional-this is not a standard RMII signal) | PK4/PP0 |

a. RX_ER is an optional standard RMII signal and is not used in this device

### 24.3.3 DMA Controller

The Ethernet Controller's integrated DMA is used to optimize data transfer between the MAC and system SRAM memory. The DMA has independent transmit and receive engines.

The DMA transmit engine transfers data from system memory to the Ethernet TX/RX Controller, while the receive engine transfers data from the RX FIFO to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. Fixed burst lengths of 1, 4, 8, or 16 words are supported along with re-initiation of bursts when retry or burst termination responses occur. For a burst retry, if the remaining address count is greater than 1 and the RIB bit in the **Ethernet MAC DMA Bus Mode (EMACDMABUSMOD)** register is clear, then the transfer resends data in one continuous burst. When one transfer is left, it is done as a single burst and the transaction is terminated immediately afterward. If the RIB bit in the **EMACDMABUSMOD** register is set, the DMA sends the remaining data in fixed burst sizes of 1, 4, 8, or 16 words.

The application may also choose between solely fixed bursts or mixed bursts by the DMA. If the MB bit is set and the FB bit is clear in the **EMACDMABUSMOD** register, then the DMA uses fixed bursts for burst sizes less than 16 and a full, non-divided burst for lengths greater than 16. Fixed burst lengths allow for more DMA bus arbitration with other masters. Maximum burst transfer lengths can be programmed for both the receive and transmit channels of the DMA through the PBL, RPBL and 8xPBL bit fields in the **EMACDMABUSMOD** register.

The DMA Controller requests a read transfer only when it can accept the received burst data completely. Data read from the bus is always pushed into the DMA without any delay or busy cycles. The DMA requests write transfers only when it has sufficient data to transfer the burst completely. When operating in fixed burst length mode, the DMA interface continues to burst with dummy data until the specified length is completed. The Ethernet controller can be programmed to interrupt the CPU in situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The integrated Ethernet DMA communicates through two data structures:

- Control and Status registers
- Descriptor lists and data buffers.

The DMA writes data frames received by the MAC to the receive buffer in system memory and transfers data frames for transmission from system memory to the MAC. Descriptors that reside in the system memory act as pointers to these buffers.

There are two descriptor lists: one for reception and one for transmission. The base address of each list is written into the **Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR)** register at offset 0xC0C and the **Ethernet Mac Transmit Descriptor List Address (EMACTXDLADDR)** register at offset 0xC10, respectively.

The descriptor structure can contain up to 8 words (32 bytes). These are described in more detail in “Enhanced and Alternate Descriptors” on page 1598. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by enabling second address chaining in both the Receive and Transmit descriptors (RDES0[14] and TDES0[20]). The descriptor lists reside in the SRAM memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used at different physical addresses rather than contiguous buffers in memory.

The data buffer also resides in the physical memory space and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data and buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when the end-of-frame is detected. Data chaining can be enabled or disabled through the descriptors.

**Note:** The EMAC DMA Controller only has access to internal system SRAM memory.

#### 24.3.3.1 Burst Access

The DMA attempts to execute fixed length Burst transfers if the FB bit is set in the **EMACDMABUSMOD** register. The maximum burst length is indicated and limited by the PBL field in the **EMACDMABUSMOD** register. The Receive and Transmit descriptors are always accessed in the maximum possible (limited by PBL) burst-size for the bytes to be read.

The TX DMA initiates a transfer only when there is sufficient space in the FIFO to accommodate the configured burst or remaining bytes of the end of a frame. When the DMA is configured for fixed-length burst, it transfers data using the best combination of fixed burst sizes of 4, 8, or 16 and single transactions. Otherwise when the FB bit is clear in the **EMACDMABUSMOD** register, the DMA transfers data as a continuous undefined burst and single transactions.

The RX DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in RX FIFO or when the end-of-frame (when it is less than the configured burst length) is detected in the RX FIFO. The DMA indicates the start address and the number of transfers required to the system. When the FB bit is set in the **EMACDMABUSMOD** register, then it transfers data using the best combination of fixed burst sizes of 4, 8, or 16 and single transactions. If the end-of frame is reached before the fixed-burst ends, then dummy transfers are performed in order to complete the fixed-burst. Otherwise, if the FB bit is clear, the DMA transfers data using INCR (undefined length) and SINGLE transactions. When the DMA is configured for address-aligned transfers, both DMA engines ensure that the first burst transfer on the system bus is less than or equal to the size of the configured PBL in the **EMACDMABUSMOD** register. Thus, all subsequent transfers start at an address that is aligned to the configured PBL. The DMA can only align the address for burst transfers up to size 16 because only bursts of 16 are supported.

#### 24.3.3.2 Data Buffer Alignment

The transmit and receive data buffers do not have any restrictions on the start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates write transfers, with address aligned to the bus

width and dummy data (old data) in the byte lanes that are not valid. This typically happens during the transfer of the beginning or end of an Ethernet frame. The software driver should discard the dummy bytes based on the start address of the buffer and size of the frame.

For example, if the transmit buffer address is 0x0000.0FF2, and 15 bytes need to be transferred, then the DMA reads five full words from address 0x0000.0FF0, but when transferring data to the TX FIFO, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures that it transfers a full 32-bit data to the TX FIFO, unless it is the end of frame.

If the receive buffer address is 0x0000.0FF2 and 15 bytes of a received frame need to be transferred, then the DMA writes five full words from address 0x0000.0FF0. However, the first two bytes of first transfer and the last three bytes of the fifth transfer have dummy data. The DMA considers the offset address only if it is the first Receive buffer of the frame. The DMA ignores the offset address and performs full word writes for the middle and the last Receive buffer of the frame.

#### 24.3.3.3 Buffer Size Calculations

The DMA does not update the size fields in the Transmit and Receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations. The TX DMA transfers the exact number of bytes (indicated by buffer size fields of TDES1) to the MAC. If a descriptor is marked as first (FS bit of TDES0 is set), then the DMA marks the first transfer from the buffer as the start-of-frame (SOF). If a descriptor is marked as last (LS bit of TDES0), then the DMA marks the last transfer from that data buffer as the end-of frame (EOF).

The RX DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the RX/TX Controller. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The Receive DMA always transfers the start of next frame with a new descriptor.

**Note:** Even when the start address of a receive buffer is not aligned to the data width of 32-bit system bus, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the Receive descriptor to have a 0x1002 offset. The Receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes, because of the start address offset.

#### 24.3.3.4 DMA Arbiter

The arbiter inside the DMA module performs the arbitration between the Transmit and Receive channel accesses. The DMA can be configured to arbitrate in a round-robin or fixed-priority configuration. When the DA bit of the **EMACDMABUSMOD** register is clear, the DMA arbiter allocates the data bus in the ratio set by the PR bit field of the **EMACDMABUSMOD** register when both the TX and RX DMA request access at the same time. When the DA bit is set, the RX DMA always has priority over the TX DMA for data access by default. However if the TXPR bit of the **EMACDMABUSMOD** register is also set, then the TX DMA always gets priority over the RX DMA.

### 24.3.3.5 Enhanced and Alternate Descriptors

Enhanced Descriptors can contain up to eight words (32 bytes) and buffers of up to 8 KB (useful for Jumbo frames). Enhanced Descriptors support IEEE 1588-2008 Advanced Timestamp and IPC Full Checksum (Type 2) Offload. These enhanced features are enabled with the TSEN bit of the **EMACTIMSTCTRL** register and the IPC bit of the **EMACCFG** register, respectively.

When using Enhanced Descriptors, set the descriptor size to eight words with the Alternate Descriptor Size (ATDS) bit in the **Ethernet MAC DMA Bus Mode (EMACDMABUSMOD)** register. If these enhanced features are not enabled, the extended descriptors (DES4 to DES7) are not required. Therefore, the software can use Alternate Descriptors with a default size of 16 bytes (4 words). For alternate descriptors, the software should clear the Alternate Descriptor Size (ATDS) bit in the **Ethernet MAC DMA Bus Mode (EMACDMABUSMOD)** register.

See the section called “Enhanced Transmit Descriptor” on page 1598 and the section called “Enhanced Receive Descriptor” on page 1603 for more details on the descriptor structure.

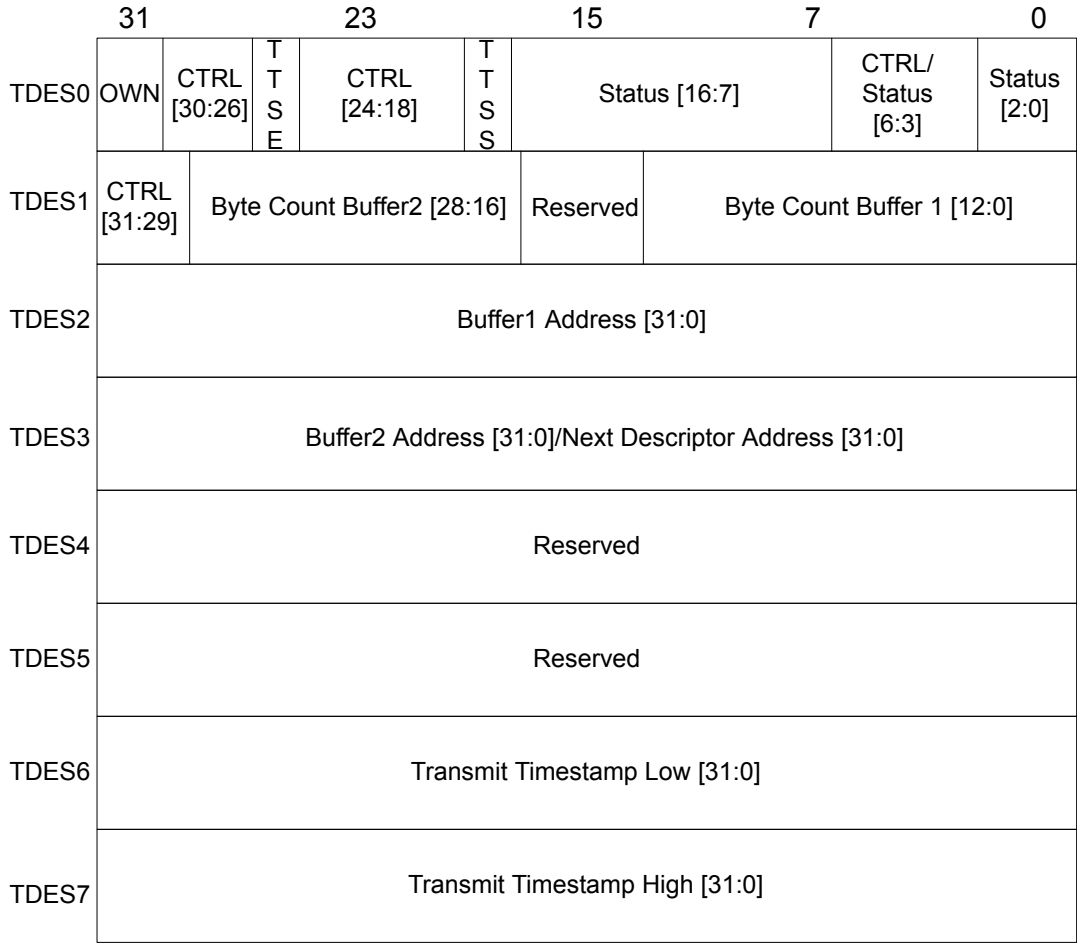
#### *Enhanced Transmit Descriptor*

The MAC requires at least one descriptor for a transmit frame. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit frame basis. Figure 24-5 on page 1599 shows the enhanced transmit descriptor. Software must program the control bits TDES0[31:18] during descriptor initialization. When the DMA updates the descriptor, it writes back all the control bits to their initialized value, clears the OWN bit and updates the status bits.

With advanced timestamp support, the snapshot of the timestamp to be taken can be enabled for a given frame by setting Bit 25 (TTSE) of TDES0. When the descriptor is closed (that is, when the OWN bit is cleared), the timestamp is written into TDES6 and TDES7.

**Note:** When the Advanced Timestamp feature is enabled, software should set the ATDS bit of the **Ethernet MAC DMA Bus Mode (EMACDMABUSMOD)** register, offset 0xC00, so that the DMA operates with extended descriptor size. When this control bit is reset to the default (0), the TDES4-TDES7 descriptor space is not valid and only Alternate Descriptors are available, with a default size of 16 bytes (4 words).

**Figure 24-5. Enhanced Transmit Descriptor Structure**




The following tables define the Enhanced Transmit Descriptors. Transmit Descriptor 0 (TDES0) contains the transmitted frame status and the descriptor ownership information. TDES1 contains the buffer sizes and other bits which control the descriptor chain or ring and the frame being transferred. TDES2 contains the address pointer to the first buffer of the descriptor. TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor. TDES6 and TDES7 contain the timestamp.

**Table 24-3. Enhanced Transmit Descriptor 0 (TDES0)**

| Bit | Description |
|---|---|
| 31 | OWN: Own Bit<br>When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit. |
| 30 | IC: Interrupt on Completion<br>When set this bit sets the Transmit Interrupt (TI) bit in the EMACDMARIS register when the frame contained in this descriptor has been transmitted. |

**Table 24-3. Enhanced Transmit Descriptor 0 (TDES0) *(continued)***

| Bit | Description |
|---|---|
| 29 | LS: Last Segment<br>When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, the TBS1 or TBS2 field in TDES1 should have a non-zero value. |
| 28 | FS: First Segment<br>When set, this bit indicates that the buffer contains the first segment of a frame. |
| 27 | DC: Disable CRC<br>When set, the MAC does not append a Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set. |
| 26 | DP: Disable Padding<br>When set, the MAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set. |
| 25 | TTSE: Transmit Timestamp Enable<br>When set, this bit enables IEEE1588 hardware timestamping for the transmit frame referenced by the descriptor. This bit is only valid when the First Segment Control bit (TDES0[28] is set. |
| 24 | CRCR: CRC Replacement Control<br>When set, the MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The CPU should ensure that the CRC bytes are present in the frame being transferred from the Transmit Buffer. CRC replacement is done only when Bit 27 (DC) is set to 1. |
| 23:22 | CIC: Checksum Insertion Control<br>These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6. This field is valid when the First Segment control bit (TDES0[28]) is set.<br>■ 0x0 = Do nothing. Checksum Engine bypassed.<br>■ 0x1 = Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram.<br>■ 0x2 = Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame's Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4.<br>■ 0x3 = Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. The TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4.<br>The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly. |
| 21 | TER: Transmit End of Ring<br>When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring. |
| 20 | TCH: Second Address Chained<br>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a "don’t care" value. TDES0[21] takes precedence over TDES0[20]. |

**Table 24-3. Enhanced Transmit Descriptor 0 (TDES0) *(continued)***

| Bit | Description |
|---|---|
| 19:18 | VLIC: VLAN Insertion Control<br>When set, these bits request the MAC to perform VLAN tagging or untagging before transmitting the frames. If the frame is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes.<br>The values of this field are as follows:<br>■ 0x0= Do not add a VLAN tag<br>■ 0x1= Remove the VLAN tag from the frames before transmission.<br>■ 0x2= Insert a VLAN tag with the tag value programmed in the **Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP)** register, offset 0x584.<br>■ 0x3= Replace the VLAN tag in frame with the tag value programmed in the **EMACVLNINCREP** register. This field is valid when the First Segment control bit (TDES0[28]) is set. |
| 17 | TTSS:TX Timestamp<br>This status bit indicates that a timestamp has been captured for the corresponding transmit frame. When this bit is set, TDES6 and TDES7 have timestamp values that were captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES0[29]) in a descriptor is set. |
| 16 | IHE: IP Header Error<br>When set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when TX Checksum Offload is enabled. Otherwise, it is reserved. If the Checksum Offload Engine detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload. |
| 15 | ES: Error Summary<br>Indicates the logical OR of the following bits:<br>■ TDES0[16]: IP Header Error<br>■ TDES0[14]: Jabber Timeout<br>■ TDES0[13]: Frame Flush<br>■ TDES0[12]: Payload Checksum Error<br>■ TDES0[11]: Loss of Carrier<br>■ TDES0[10]: No Carrier<br>■ TDES0[9]: Late Collision<br>■ TDES0[8]: Excessive Collision<br>■ TDES0[2]: Excessive Deferral<br>■ TDES0[1]: Underflow error |
| 14 | JT: Jabber Timeout<br>When set, this bit indicates that the MAC transmitter has experienced a jabber timeout. This bit can only be set when the Jabber Disabled (JD) bit of the **EMACCFG** register is clear. |
| 13 | FF: Frame Flushed<br>When set, this bit indicates that the DMA flushed the frame because of a software flush command given by the CPU. |
| 12 | IPE: IP Payload Error<br>When set, this bit indicates that MAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch. |

**Table 24-3. Enhanced Transmit Descriptor 0 (TDES0) *(continued)***

| Bit | Description |
|---|---|
| 11 | LC: Loss of Carrier<br>When set, this bit indicates that Loss of Carrier occurred during frame transmission. This is valid only for the frames transmitted without collision and when the MAC operates in half-duplex mode. |
| 10 | NC: No Carrier<br>When set, this bit indicates that the carrier sense signal form the PHY was not asserted during transmission. |
| 9 | LC: Late Collision<br>When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in MII Mode). Not valid if Underflow Error (bit 1) is set. |
| 8 | Excessive Collision<br>When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the Disable Retry (DR) bit in **EMACCFG** register is set, this bit is set after the first collision and the transmission of the frame is aborted. |
| 7 | VF: VLAN Frame<br>When set, this bit indicates that the transmitted frame was a VLAN-type frame. |
| 6:3 | CC: Collision Count<br>This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collision bit (TDES0[8]) is set. |
| 2 | ED: Excessive Deferral<br>When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times when Jumbo Frame is enabled). This bit is dependent on the Deferral Check (DC) bit being enabled in the **EMACCFG** register. |
| 1 | UF: Underflow Error<br>When set, this bit indicates that the MAC aborted the frame because the data arrived late from system memory. Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters the suspended state and sets both Transmit Underflow (UNF) and Transmit Interrupt (TI) bit in the **EMACDMARIS** register. |
| 0 | DB: Deferred Bit<br>This bit indicates the deferral mechanism is active and that the transmit state machine sends a JAM pattern to defer reception when it senses a carrier before a normal transmission is scheduled. This bit is only valid in half-duplex mode. |

**Table 24-4. Enhanced Transmit Descriptor 1 (TDES1)**

| Bit | Description |
|---|---|
| 31:29 | SAIC: SA Insertion Control<br>These bits request the MAC to add or replace the Source Address field in the Ethernet frame with the value given in the MAC Address 0 register. If the Source Address field is modified in a frame, the MAC automatically recalculates and replaces the CRC bytes.<br>The Bit 31 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[30:29]:<br>■ 0x0= Do not include the source address.<br>■ 0x1= Insert the source address. For reliable transmission, the application must provide frames without source addresses.<br>■ 0x2= Replace the source address. For reliable transmission, the application must provide frames with source addresses.<br>■ 0x3= Reserved<br>These bits are valid when the First Segment control bit (TDES0[28]) is set. |

**Table 24-4. Enhanced Transmit Descriptor 1 (TDES1) *(continued)***

| Bit | Description |
|---|---|
| 28:16 | TBS2: Transmit Buffer 2 Size<br>This field indicates the second data buffer size in bytes. This field is not valid if TDES0[20] is set. |
| 15:13 | Reserved |
| 12:0 | TBS1: Transmit Buffer 1 Size<br>These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]). |

**Table 24-5. Enhanced Transmit Descriptor 2 (TDES2)**

| Bit | Description |
|---|---|
| 31:0 | Buffer 1 Address Pointer<br>These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. Note that the buffers are stored in SRAM. |

**Table 24-6. Enhanced Transmit Descriptor 3 (TDES3)**

| Bit | Description |
|---|---|
| 31:0 | Buffer 2 Address Pointer (Next Descriptor Address)<br>Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES0[20]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES0[20] is set.<br>Note that the buffers are stored in SRAM. |

**Table 24-7. Enhanced Transmit Descriptor 6 (TDES6)**

| Bit | Description |
|---|---|
| 31:0 | TTSL: Transmit Frame Timestamp Low<br>This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment bit (LS), TDES0[29], in the descriptor is set and Timestamp status (TTSS) bit, TDES0[17], is set. |

**Table 24-8. Enhanced Transmit Descriptor 7 (TDES7)**

| Bit | Description |
|---|---|
| 31:0 | TTSH: Transmit Frame Timestamp High<br>This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field has the timestamp only if the Last Segment bit (LS), TDES0[29], in the descriptor is set and Timestamp status (TTSS) bit , TDES0[17], is set. |

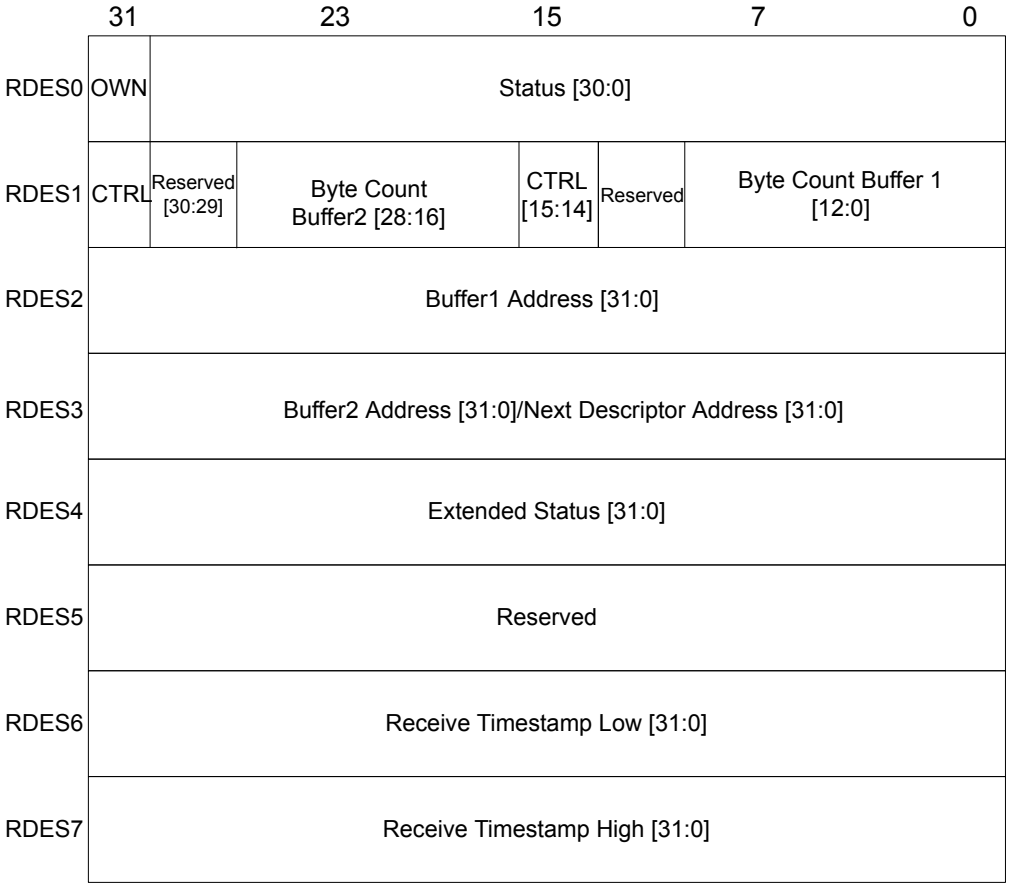
### *Enhanced Receive Descriptor*

The DMA requires at least two descriptors when receiving a frame. The DMA always attempts to acquire an extra descriptor in anticipation of an incoming frame. Before the DMA closes a descriptor, it attempts to acquire the next descriptor even if no frames are received. In a single descriptor (receive) system, the subsystem generates a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Figure 24-6 on page 1604 shows the enhanced receive descriptor. This descriptor is used when Advanced Timestamp or the Checksum Offload Engine is enabled.

**Note:** When the Advanced Timestamp or Checksum Offload Engine features are enabled, software should set the ATDS bit of the **Ethernet MAC DMA Bus Mode (EMACDMABUSMOD)**

register, offset 0xC00, so that the DMA operates with extended descriptor size. When this control bit is reset to the default (0), the TDES4-TDES7 descriptor space is not valid and only Alternate Descriptors are available, with a default size of 16 bytes (4 words).

**Figure 24-6. Enhanced Receive Descriptor Structure**




The following tables define the Enhanced Receive Descriptors. RDES0 contains the received frame status, the frame length, and the descriptor ownership information. RDES1 contains the buffer sizes and other bits that control the descriptor chain or ring. RDES2 and RDES3 contains the address pointers to the first and second data buffers in the descriptor. The availability of the extended status is indicated by Bit 0 of RDES0. RDES6 and RDES7 are available only when the Advanced Timestamp or IP Checksum Full Offload feature is enabled.

**Table 24-9. Enhanced Receive Descriptor 0 (RDES0)**

| Bit | Description |
|---|---|
| 31 | OWN: Own Bit<br>When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full. |
| 30 | AFM: Destination Address Filter Fail<br>When set, this bit indicates a frame failed in the DA filter in the MAC. |

**Table 24-9. Enhanced Receive Descriptor 0 (RDES0) *(continued)***

| Bit | Description |
|---|---|
| 29:16 | FL: Frame Length<br>These bits indicate the byte length of the received frame that was transferred to the system memory. This field is valid when the Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or the Overflow Error bit (RDES0[11]) is clear.<br>When the Last Descriptor bit is not set, this field indicates the accumulated number of bytes that have been transferred for the current frame. The inclusion of CRC length in the frame length depends on the settings of CRC configuration bits, ACS and CST in the **EMACCFG** register. |
| 15 | ES: Error Summary<br>Indicates the logical OR of the following bits:<br>■ RDES0[14]: Descriptor Error<br>■ RDES0[11]: Overflow Error<br>■ RDES0[7]: IPC Checksum (Type 2) or Giant Frame<br>■ RDES0[6]: Late Collision<br>■ RDES0[4]: Watchdog Timeout<br>■ RDES0[3]: Receive Error<br>■ RDES0[1]: CRC Error<br>■ RDES[4:3]: IP Header or Payload Error<br>This field is valid only when the Last Descriptor (RDES0[8]) is set. |
| 14 | DE: Descriptor Error<br>When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set. |
| 13 | SAF: Source Address Filter Fail<br>When set, this bit indicates that the SA field of frame failed the SA Filter in the MAC. |
| 12 | LE: Length Error<br>When set, this bit indicates that the actual length of the frame received and the Length/Type field do not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset. Length error status is not valid when CRC error is present. |
| 11 | OE: Overflow Error<br>When set, this bit indicates that the received frame is damaged because of buffer overflow in RX FIFO.<br>**Note:** This bit is set only when the DMA transfers a partial frame to the application. This happens only when the RX FIFO is operating in the threshold mode. In the store-and-forward mode, all partial frames are dropped completely in RX FIFO. |
| 10 | VLAN: VLAN Tag<br>When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the MAC. The VLAN tagging depends on checking VLAN fields of the received frame configured in the **Ethernet MAC VLAN Tag (EMACVLANTG)** register, offset 0x01C |
| 9 | FS: First Descriptor<br>When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame. |
| 8 | LS: Last Descriptor<br>When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame. |

**Table 24-9. Enhanced Receive Descriptor 0 (RDES0) *(continued)***

| Bit | Description |
|---|---|
| 7 | Timestamp Available or Giant Frame<br>When Advanced Timestamp feature is enabled, this bit indicates that a snapshot of the Timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set.<br>Otherwise, this bit, when set, indicates the Giant Frame Status. Giant frames are larger than 1,518-byte (or 1,522-byte for VLAN or 2,000-byte when Bit 27 of MAC Configuration register is set) normal frames and larger than 9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled. |
| 6 | LC: Late Collision<br>When set, this bit indicates that a late collision has occurred while receiving the frame in half-duplex mode. |
| 5 | FT: Frame Type<br>When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 1,536). When this bit is reset, it indicates that the received frame is an IEEE 802.3 frame. This bit is not valid for Runt frames less than 14 bytes. In addition when the IPC bit is set in the EMACCFG register, this bit conveys different information. See Table 24-10 on page 1606. |
| 4 | RWT: Receive Watchdog Timeout<br>When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout. |
| 3 | RE: Receive Error<br>When set, this bit indicates that an error occurred during frame reception. |
| 2 | DE: Dribble Bit Error<br>When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode. |
| 1 | CE: CRC Error<br>When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor bit (RDES0[8]) is set. |
| 0 | Extended Status Available/RX MAC Address<br>When set, this bit indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set. This bit is invalid when Bit 30 is set. |

Table 24-10 on page 1606 shows the frame information conveyed in bits 7, 5, and 0 of RDES0 when the Checksum Offload Engine is enabled and disabled through the `IPC` bit in the **EMACCFG** register.

**Table 24-10. RDES0 Checksum Offload bits**

| Bit 5: Frame Type | Bit 7: IPC Checksum Error | Bit 0: Payload Checksum Error | IPC bit value in EMACCFG register | Frame Status |
|---|---|---|---|---|
| 0 | 0 | 0 | X | IEEE 802.3 Type frame (Length field value is less than 1,536). This status definition is valid even when the Checksum Offload engine is disabled. |
| 1 | 0 | 0 | 0 | IPv4/IPv6 Type frame in which no checksum error is detected. |
| 1 | 0 | 0 | 1 | The frame is an IEEE 802.3 Type frame (Length field value is greater than or equal to 1,536). |
| 1 | 0 | 1 | 1 | IPv4/IPv6 Type frame with a payload checksum error detected |
| 1 | 1 | 1 | 1 | IPv4/IPv6 Type frame with both IP header and payload checksum errors detected |
| 0 | 0 | 1 | 1 | IPv4/IPv6 Type frame with no IP header checksum error and the payload check bypassed, due to an unsupported payload |
| 0 | 1 | 1 | 1 | A Type frame that is neither IPv4 or IPv6 (the Checksum Offload engine bypasses checksum completely.) |

**Table 24-10. RDES0 Checksum Offload bits *(continued)***

| Bit 5: Frame Type | Bit 7: IPC Checksum Error | Bit 0: Payload Checksum Error | IPC bit value in EMACCFG register | Frame Status |
|---|---|---|---|---|
| 0 | 1 | 0 | X | Reserved |

**Table 24-11. Enhanced Receive Descriptor 1 (RDES1)**

| Bit | Description |
|---|---|
| 31 | Disable Interrupt on Completion<br>When set, this bit prevents the setting of the Receive Interrupt (RI) bit in the **EMACDMARIS** register and prevents the receive interrupt from being asserted. |
| 30:29 | Reserved |
| 28:16 | RBS2: Receive Buffer 2 Size<br>These bits indicate the second data buffer size. The buffer size must be a multiple of 4, even if the value of RDES3 (buffer 2 address pointer) is not aligned to the bus width. When the buffer size is not a multiple of 4, the resulting behavior is undefined. This field is not valid if RCH bit (RDES1[14]) is set. |
| 15 | RER: Receive End of Ring<br>When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring. |
| 14 | RCH: Second Address Chained<br>When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a "don't care" value. RDES1[15] takes precedence over RDES1[14]. |
| 13 | Reserved |
| 12:0 | RBS1: Receive Buffer 1 Size<br>These bits indicate the first data buffer size in bytes. The buffer size must be a multiple of 4 even if the value of RDES2 (buffer 1 address pointer) is not aligned to the bus width. When the buffer size is not a multiple of 4, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor depending on the value of RCH (Bit 14). |

**Table 24-12. Enhanced Receive Descriptor 2 (RDES2)**

| Bit | Description |
|---|---|
| 31:0 | Buffer 1 Address Pointer<br>These bits indicate the physical address of Buffer 1. The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. The DMA performs a write operation with the RDES2[1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored.<br>Note that buffers should be word-aligned. |

**Table 24-13. Enhanced Receive Descriptor 3 (RDES3)**

| Bit | Description |
|---|---|
| 31:0 | Buffer 2 Address Pointer (Next Descriptor Address)<br>These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[14]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[14] is set, the buffer (Next Descriptor) address pointer must be bus word-aligned (RDES3[1:0] = 0) However, when RDES1[14] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3 [1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored. |

**Table 24-14. Enhanced Received Descriptor 4 (RDES4)**

| Bit | Description |
|---|---|
| 31:15 | Reserved |
| 14 | Timestamp Dropped<br>When set, this bit indicates that the timestamp was captured for this frame but got dropped in the RX FIFO because of overflow. |
| 13 | PTP Version<br>When set, this bit indicates that the received PTP message uses the IEEE 1588 version 2 format. When reset, it uses the version 1 format. |
| 12 | PTP Frame Type<br>When set, this bit indicates that the PTP message is sent directly over Ethernet. When this bit is clear and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information about IPv4 or IPv6 can be obtained from Bits 6 and 7. |
| 11:8 | MessageType<br>These bits are encoded to give the type of the message received:<br>■ 0x0= No PTP message received<br>■ 0x1= SYNC (all clock types)<br>■ 0x2= Follow_Up (all clock types)<br>■ 0x3= Delay_Req (all clock types)<br>■ 0x4= Delay_Resp (all clock types)<br>■ 0x5= Pdelay_Req (in peer-to-peer transparent clock)<br>■ 0x6= Pdelay_Resp (in peer-to-peer transparent clock)<br>■ 0x7= Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock)<br>■ 0x8= Announce<br>■ 0x9= Management<br>■ 0xA= Signaling<br>■ 0xB to 0xE= Reserved<br>■ 0xF= PTP packet with Reserved message type |
| 7 | IPv6 Packet Received<br>When set, this bit indicates that the received packet is an IPv6 packet. This bit is updated only when the IPC bit of the **EMACCFG** register is set. |
| 6 | IPv4 Packet Received<br>When set, this bit indicates that the received packet is an IPv4 packet. This bit is updated only when the IPC bit of the **EMACCFG** register is set. |
| 5 | IP Checksum Bypassed<br>When set, this bit indicates that the checksum offload engine is bypassed. |
| 4 | IP Payload Error<br>When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. This bit is valid when either Bit 7 or Bit 6 is set. |

**Table 24-14. Enhanced Received Descriptor 4 (RDES4) *(continued)***

| Bit | Description |
|---|---|
| 3 | IP Header Error<br>When set, this bit indicates that either the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or the IP datagram version is not consistent with the Ethernet Type value. This bit is valid when either Bit 7 or Bit 6 is set. |
| 2:0 | IP Payload Type<br>These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE). The COE sets this field to 0x0 if it does not process the IP datagram's payload due to an IP header error or fragmented IP packet.<br>■ 0x0= Unknown or did not process IP payload<br>■ 0x1= UDP<br>■ 0x2= TCP<br>■ 0x3= ICMP<br>■ 0x4 to 0x7= Reserved<br>This bit is valid when either Bit 7 or Bit 6 is set. |

**Table 24-15. Enhanced Receive Descriptor 6 (RDES6)**

| Bit | Description |
|---|---|
| 31:0 | RTSL: Receive Frame Timestamp Low<br>This field is updated by the DMA with the least significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by the DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]). |

**Table 24-16. Enhanced Receive Descriptor 7 (RDES7)**

| Bit | Description |
|---|---|
| 31:0 | RTSH: Receive Frame Timestamp High<br>This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]). |

### 24.3.3.6 DMA Transmission Operation

The following sections describe the modes of the transmit operation.

#### *TX DMA Default Operation*

The TX DMA engine in default mode, operates as follows:

1. The CPU configures the transmit descriptor (TDES0-TDES3) and sets the OWN bit (TDES0[31]) after setting up the corresponding data buffers with the Ethernet frame.

2. When the ST bit of the **Ethernet MAC DMA Operation Mode (EMACDMAOPMODE)** register, offset 0xC18, is set, the DMA enters the RUN state.

3. While in RUN state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the CPU (TDES0[31]=0), or if an error condition occurs, transmission is suspended and both the Transmit Buffer Unavailable (TU) bit and the

Normal Interrupt Summary (NIS) bit are set in the **Ethernet MAC DMA Interrupt Status (EMACDMARIS)** register, offset 0xC14. The transmit engine the proceeds to Step 9.

4. If the acquired descriptor is flagged as owned by the DMA (TDES0[31]=1), the DMA decodes the transmit data buffer address from the acquired descriptor.

5. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.

6. The DMA fetches the transmit data from system memory and transfers the data to the TX/RX Controller for transmission.

7. If the Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3, 4, and 5 are repeated until the end of the Ethernet frame data is transferred to the TX/RX Controller.

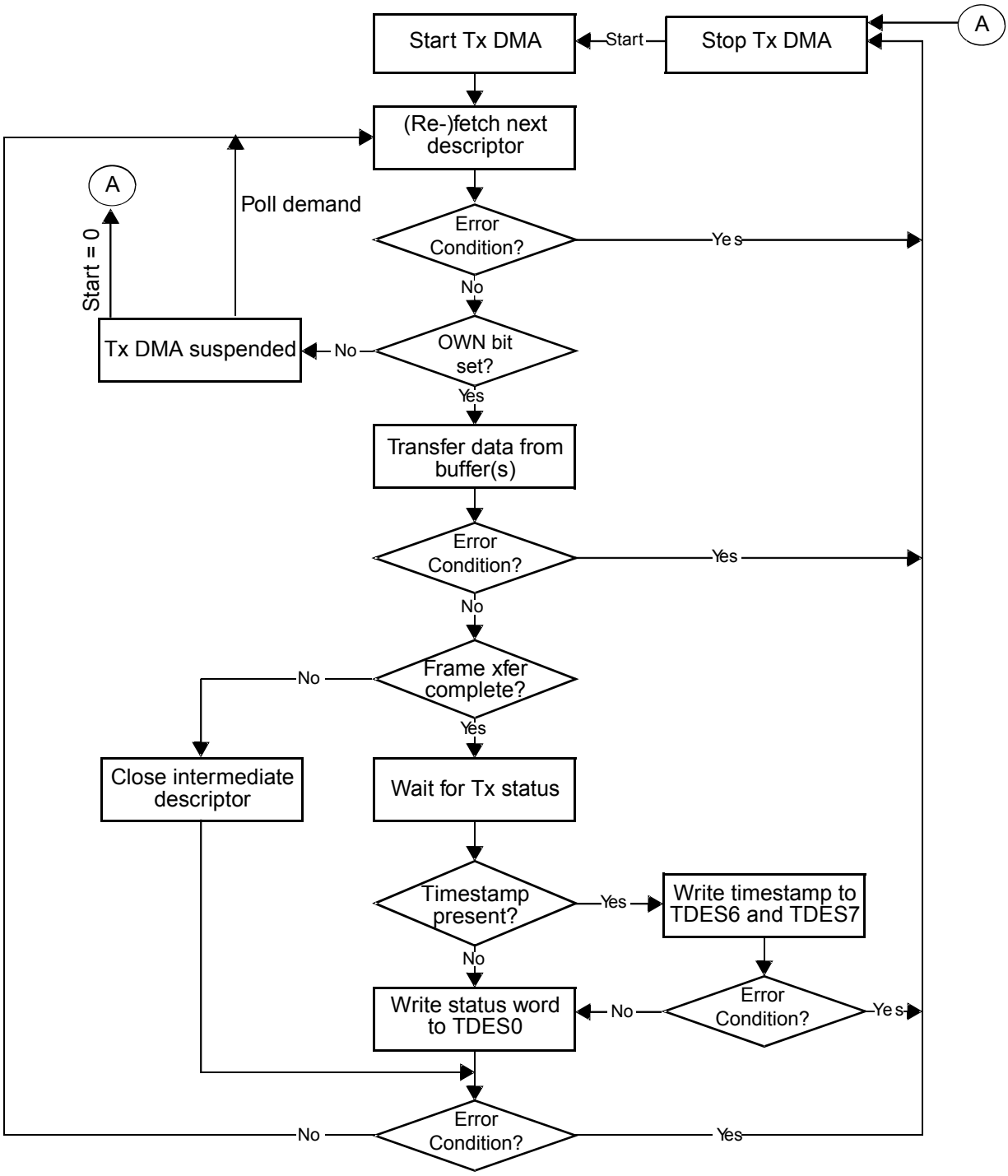
8. When frame transmission is complete, if IEEE 1588 timestamping was enabled for the frame (as indicated in the transmit status) the timestamp value is written to the transmit descriptor (TDES6 and TDES7) that contains the end-of-frame buffer. The status information is then written to transmit descriptor TDES0. Because the OWN bit is cleared during this step, the CPU now owns this descriptor. If timestamping was not enabled for this frame, the DMA does not alter the contents of TDES6 and TDES7.

9. The Transmit Interrupt (TI) bit is set in the **EMACDMARIS** register after transmission completion of a frame that has Interrupt on Completion set in its last descriptor. The Interrupt on Completion bit resides in TDES0[30]. The DMA engine then returns to Step 3.

10. In the suspend state, the DMA tries to reacquire the descriptor (and thereby return to Step 3) when it receives a transmit poll demand in the **Ethernet MAC Transmit Poll Demand (EMACTXPOLLD)** register, offset 0xC04, and the Underflow Interrupt Status (UNF) bit is cleared in the **EMACDMARIS** register. If the CPU stopped the DMA by clearing the ST bit of the **EMACDMAOPMODE** register, the DMA enters the STOP state.

Figure 24-7 on page 1611 shows the flow for the TX DMA default operation.

**Figure 24-7. TX DMA Default Operation Using Descriptors**




### *TX DMA OSF Mode Operation*

While in the RUN state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first if the OSF bit of the **EMACDMAOPMODE** register is set. As the transmit process finishes transferring the first frame, it immediately polls the transmit descriptor list

for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

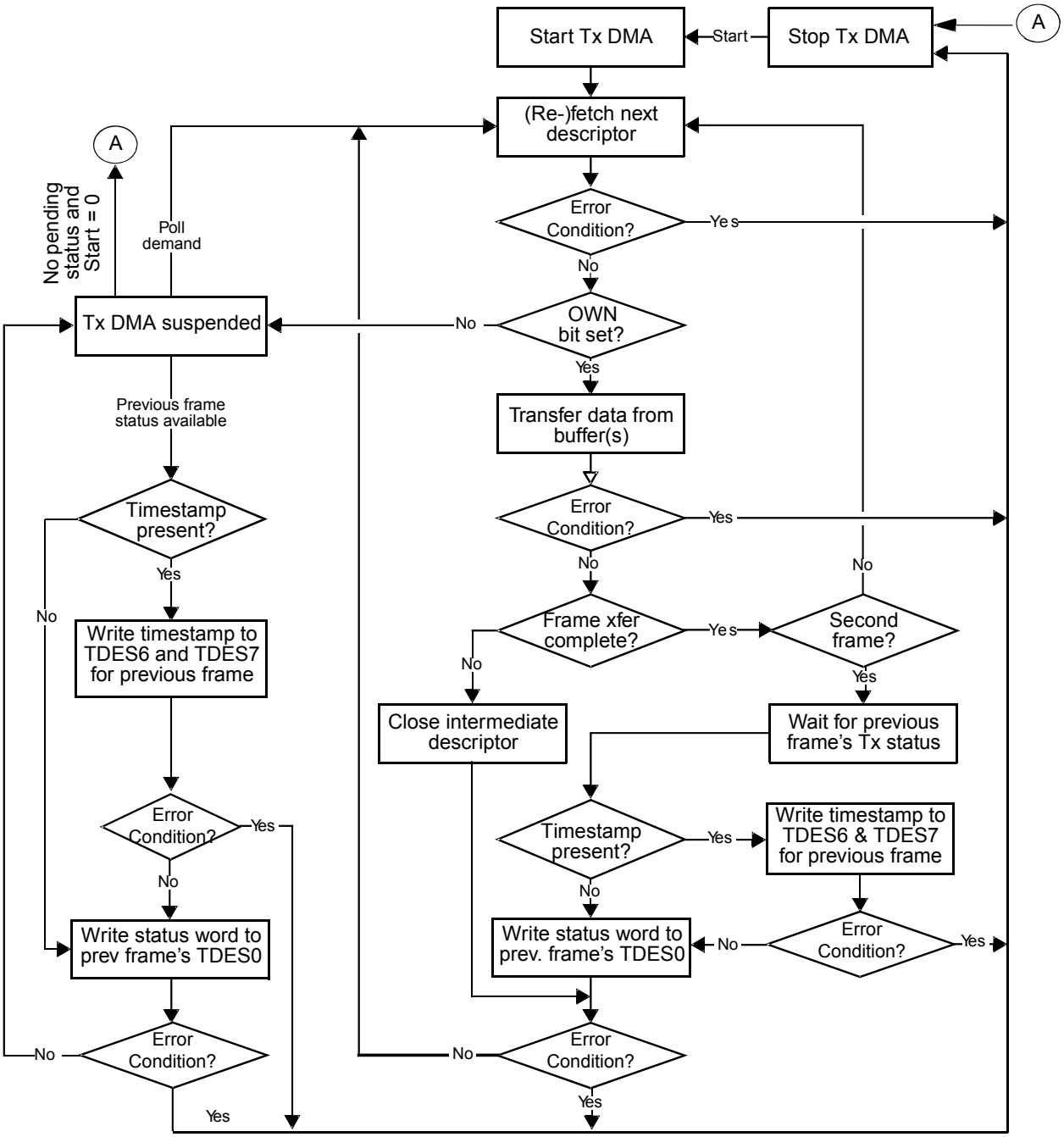
In Operate on Second Frame (OSF) mode, the RUN state TX DMA operates in the following sequence:

1. The DMA operates as described in Steps 1 to 6 (see page 1609).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into SUSPEND mode and skips to 7.
4. The DMA fetches the Transmit frame from system memory and transfers the frame until the end-of-frame data is reached. It closes the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's transmission status and timestamp. When the status is available, the DMA writes the timestamp to TDES6 and TDES7, if such timestamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared OWN bit, to the corresponding TDES0, thus closing the descriptor. If timestamping was not enabled for the previous frame, the DMA does not alter the contents of TDES6 and TDES7.
6. If enabled, the Transmit interrupt (`TI`) bit is set in the **EMACDMARIS** register, the DMA fetches the next descriptor and then proceeds to 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into SUSPEND mode
7. In SUSPEND mode, if a pending status and timestamp are received, the DMA writes the timestamp (if enabled for the current frame) to TDES6 and TDES7, then writes the status to the corresponding TDES0. It then sets relevant interrupts and returns to SUSPEND mode.
8. The DMA can exit SUSPEND mode and enter the RUN state only after receiving the Transmit Poll demand in the **EMACTXPOLLD** register.

**Note:** The DMA fetches the next descriptor in advance before closing the current descriptor. Therefore, the descriptor chain should have more than two different descriptors for correct and proper operation.

Figure 24-8 on page 1613 shows the flow for the TX DMA Operate-On-Second-Frame (OSF) operation.

**Figure 24-8. TX DMA OSF Mode Operation Using Descriptors**




### *Transmit Frame Processing*

The TX DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and Frame Check Sequence (FCS) fields. The Destination Address (DA), Source Address (SA), and Type/Length fields must contain valid data. If the Transmit Descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes. Frames can be data-chained and can span several buffers. Frames must be delimited by the First Segment Descriptor and the Last Segment Descriptor,

respectively. The First Descriptor bit is located at TDES0[28] and the Last Descriptor is located at TDES0[29].

As transmission starts, the First Descriptor must have TDES0[28] set. When this occurs, frame data transfers from the host memory buffer to the TX FIFO. Concurrently, if the current frame has the Last Segment Descriptor (TDES0[29]) clear, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES0[28] clear. If TDES0[29] is clear, it indicates an intermediary buffer. If TDES0[29] is set, it indicates the last buffer of the frame. After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the Transmit Descriptor word of the descriptor that has the last segment bit set in Transmit Descriptor. At this time, if Interrupt on Completion (IC) bit is set, the TI bit in the **EMACDMARIS** register is set, the next descriptor is fetched and the process repeats.

The actual frame transmission begins after the TX FIFO has reached either the transmit threshold as configured by the TTC bit field of the **EMACDMAOPMODE** register, or a full frame is contained in the TX FIFO. To wait until there is a full frame in the TX FIFO the TSF bit in the **EMACDMAOPMODE** register must be set. Descriptors are released (OWN bit in the TDES0[31] cleared) when the DMA finishes transferring the frame.

**Note:** To ensure proper transmission of a frame and the next frame, the transmit descriptor that has the Last Descriptor bit (TDES0[29]) set, must specify a non-zero buffer size.

***Transmit Polling Suspended***

Transmit polling can be suspended by either of the following conditions:

- The DMA detects a descriptor owned by the CPU (TDES0[31]=0). To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command.
- A frame transmission is aborted when a transmit error because of underflow is detected. The appropriate Transmit Descriptor 0 (TDES0) bit is set.

If the DMA goes into SUSPEND state because of the first condition, then both the Normal Interrupt Summary (NIS) bit and the Transmit Buffer Unavailable (TU) bit are set in the **EMACDMARIS** register. If the second condition occurs, the Abnormal Interrupt Summary (AIS) bit and the Transmit Underflow (UNF) bit of the **EMACDMARIS** register are set and the information is written to Transmit Descriptor 0, causing the suspension.

In both cases, the position in the Transmit list is retained . The retained position is that of the descriptor following the last descriptor closed by the DMA. The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension case.

### 24.3.3.7 DMA Receive Operation

The following section describes the receive operation process.

***Default Receive Operation***

The RX DMA engine's reception sequence is as follows:

1. The host sets up receive descriptors (RDES0-RDES3) and sets the OWN bit (RDES0[31]).
2. When the SR bit of the **EMACDMAOPMODE** register is set, the DMA enters the RUN state. While in the RUN state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the CPU), the DMA enters the Suspend state and jumps to Step 9.

3. The DMA decodes the receive data buffer address from the acquired descriptors.

4. Incoming frames are processed and placed in the acquired descriptor's data buffers.

5. When the buffer is full or the frame transfer is complete, the RX DMA engine fetches the next descriptor.

6. If the current frame transfer is complete, the DMA proceeds to Step 7. If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error (DE) bit in RDES0 (unless flushing is disabled through the DFF bit in the **EMACDMAOPMODE** register). The DMA closes the current descriptor (clears the OWN bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value. If flushing is not disabled, then the DMA would mark it as the last descriptor. In either case, the DMA proceeds to Step 8. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to Step 4.

7. If IEEE 1588 timestamping is enabled, the DMA writes the timestamp to the current descriptor's RDES6 and RDES7. It then takes the receive frame's status and writes the status word to the current descriptor's RDES0, with the OWN bit cleared and the Last Segment (LS) bit set. If the host stopped the RX DMA by clearing the SR bit of the **EMACDMAOPMODE** register, DMA goes to the STOP state, otherwise the RX DMA proceeds to Step 8.

8. The RX DMA engine checks the last descriptor's OWN bit. If the CPU owns the descriptor (OWN bit is 0), the RU bit of the **EMACDMARIS** register is set and the DMA RX engine enters the SUSPEND state. If the DMA owns the descriptor, the engine returns to Step 4 and awaits the next frame.

9. Before the RX DMA engine enters the SUSPEND state, partial frames are flushed from the RX FIFO. Flushing can be controlled through the DFF bit of the **EMACDMAOPMODE** register.

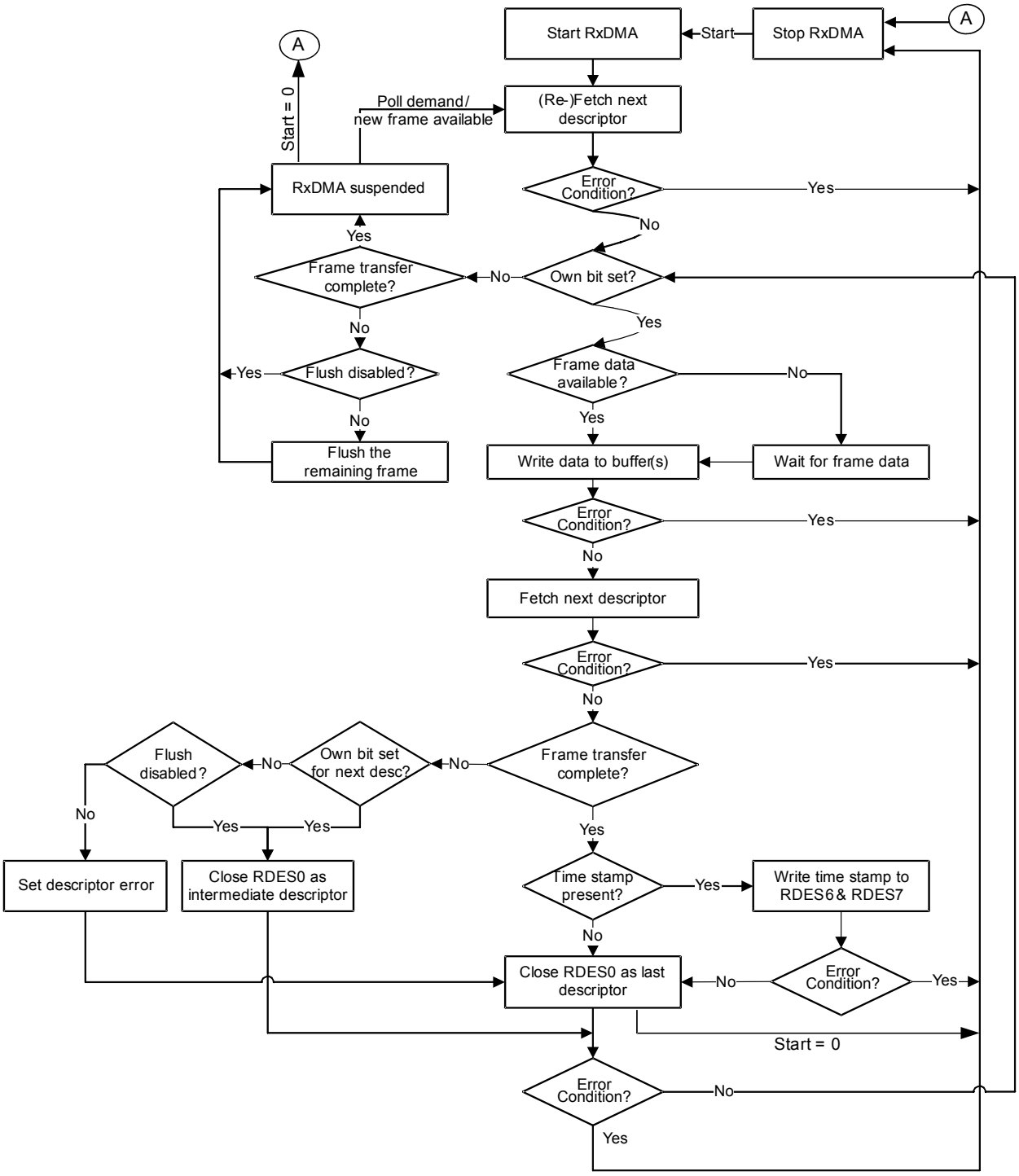
10. The RX DMA enters the STOP state if the CPU has cleared the SR bit of the **EMACDMAOPMODE** register. Otherwise, it exits the SUSPEND state when a Receive Poll Demand is given or the start of the next frame is available from the RX FIFO. The DMA engine proceeds to Step 2 and re-fetches the next descriptor.

The DMA does not acknowledge accepting status from the TX/RX Controller until it has completed the timestamp write-back and is ready to perform status write-back to the descriptor.

If software has enabled timestamping through the **Ethernet MAC Timestamp Control (EMACTIMSTCTRL)** register, offset 0x700, when a valid timestamp is not available for the frame (for example, because the receive FIFO was full before the timestamp could be written to it), the DMA writes all-ones to RDES6 and RDES7. Otherwise if timestamping is not enabled, RDES6 and RDES7 remain unchanged.

Figure 24-9 on page 1616 shows the flow of a RX DMA Operation.

**Figure 24-9. RX DMA Operation Flow**




### *Receive Descriptor Acquisition*

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- The SR bit of the **EMACDMAOPMODE** register has been set immediately after being placed in the RUN state.
- The data buffer or current descriptor is full before the frame ends for the current transfer.
- The controller has completed frame reception, but the current receive descriptor is not yet closed.
- The receive process has been suspended because of a host-owned buffer (RDES0[31]=0) and a new frame is received.
- A receive poll demand has been issued.

***Receive Frame Processing***

The MAC transfers the received frames to the system memory only when the frame passes the address filter and the frame size is greater than or equal to configurable threshold bytes set for the RX FIFO, or when the complete frame is written to the RX FIFO in store-and-forward mode.

If the frame fails the address filtering, it is dropped in the MAC block unless the Receive All (RA) bit is set in the **Ethernet MAC Frame Filter (EMACFRAMEFLTR)**, offset 0x004. If the RA bit is set, then the MAC passes all received frames. Frames that are shorter than 64 bytes, because of collision or premature termination, can be removed from the RX FIFO if the DFF bit is clear in the **EMACDMAOPMODE** register.

After 64 bytes have been received, the TX/RX Controller requests the DMA block to begin transferring the frame data to the receive buffer pointed by the current descriptor. The DMA sets the First Descriptor (RDES0[9]) bit to delimit the frame after the DMA Interface becomes ready to receive a data transfer (if DMA is not fetching transmit data from the system memory). The descriptors are released when the OWN (RDES0[31]) bit is reset to 0, either as the data buffer fills up or as the last segment of the frame is transferred to the receive buffer. If the frame is contained in a single descriptor, both Last Descriptor (RDES0[8]) and First Descriptor (RDES0[9]) are set.

The DMA fetches the next descriptor, sets the Last Descriptor (RDES0[8]) bit, and releases the RDES0 status bits in the previous frame descriptor. Then the DMA sets the RI bit of the **EMACDMARIS** register. The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host. If this occurs, the receive process sets the RU bit of the **EMACDMARIS** and enters the SUSPEND state. The position in the receive list is retained.

***Receive Process Suspend***

If a new receive frame arrives while the receive process is in SUSPEND state, the DMA refetches the current descriptor in the system memory. If the descriptor is now owned by the DMA, the receive process re-enters the RUN state and starts frame reception. If the descriptor is still owned by the CPU, by default, the DMA discards the current frame at the top of the RX FIFO and increments the missed frame counter. If more than one frame is stored in the RX FIFO, the process repeats. The discarding or flushing of the frame at the top of the RX FIFO can be prevented by disabling flushing through the DFF bit of the **EMACDMAOPMODE** register. In such conditions, the receive process sets the Receive Buffer Unavailable (RU) status and returns to the SUSPEND state.

### 24.3.3.8 DMA Interrupts

Interrupts can be generated as a result of various transfer events. The current status of interrupts can be read from the **EMACDMARIS** register and are enabled to trigger an interrupt through the programming of the **Ethernet MAC DMA Interrupt Mask (EMACDMAIM)** register. There are two groups of transfer event interrupts: Normal and Abnormal. The following lists the two groups:

- Normal Interrupts:
  - Transmit Interrupt (TI, bit 0): Indicates that frame transmission is complete.
  - Transmit Buffer Unavailable (TU, bit 2): Indicates the CPU owns the next descriptor in the transmit list and the DMA cannot acquire it.
  - Receive Interrupt (RI, bit 6): Indicates the frame reception is complete.
  - Early Receive Interrupt (ERI, bit 14): Indicates the DMA has filled the first half of the data buffer of the packet
- Abnormal Interrupts:
  - Transmit Process Stopped (TPS, bit 1): Indicates transmission is stopped.,
  - Transmit Jabber Timeout (TJT, bit 3): Indicates the Transmit Jabber Timer expired.
  - Receive FIFO Overflow (0VF, bit 4): Indicates the receive buffer had an overflow during frame reception.
  - Transmit Underflow (UNF, bit 5): Indicates the transmit buffer had an underflow during frame transmission.
  - Receive Buffer Unavailable (RU, bit): Indicates the CPU owns the next descriptor in the receive list and the DMA cannot acquire it.
  - Receive Process Stopped (RPS, bit 8): Indicates the receive process entered the STOP state.
  - Receive Watchdog Timeout (RWT, bit 9): Indicates a frame length greater than 2 KB is received (10,240 when Jumbo Frame is enabled)
  - Early Transmit Interrupt (ETI, bit 10): Indicates a frame to be transmitted is fully transferred to the TX FIFO.
  - Fatal Bus Error (FBI, bit 13): Indicates a bus error occurred.

Any of the interrupts in the Normal Interrupt group that are enabled in the **EMACDMAIM** register are ORed together to create the Normal Interrupt Summary (NIS) bit in the **EMACDMARIS** register. Any of the interrupts in the Abnormal Interrupt group that are enabled in the **EMACDMAIM** register are ORed together to create the Abnormal Interrupt Summary (AIS) bit in the **EMACDMARIS** register. Interrupts are cleared by writing a 1 to the corresponding bit position in the **EMACDMARIS** register. When all enabled interrupts within a group are cleared, the corresponding summary bit is cleared.

Interrupts are not queued and if the interrupt event occurs again before the driver has responded to it, no additional interrupts are generated. An interrupt is only generated once for simultaneous, multiple events. The driver must read the **EMACDMARIS** register for the cause of the interrupt.

The **Ethernet MAC Receive Interrupt Watchdog Timer (EMACRXINTWDT)** register, offset 0xC24 can be used to control the Receive Interrupt (RI) assertion. If the RDES1[31] bit (Receive Interrupt) bit has not been set in the receive descriptor and the **EMACRXINTWDT** register is programmed with a non-zero value, it gets activated as soon as the RX DMA completes a transfer of a received frame to system memory without asserting the receive interrupt. When this counter runs out as per the programmed value, the RI bit is set in the **EMACDMARIS** register and the interrupt is asserted

if the corresponding RI bit is enabled in the **EMACDMAIM** register. This counter gets disabled before it runs out if a frame is transferred to memory and the RI bit is set because it is enabled for that descriptor.

#### 24.3.3.9 DMA Bus Error

If an internal bus error occurs during a DMA transfer, the fatal bus error (FBI) interrupt is set in the **EMACDMARIS** register and the Access Error status (AE) bit field in the **EMACDMARIS** register indicates the type of error that caused the bus error. The DMA controller can resume operation only after soft resetting the Ethernet MAC and the re-initializing the DMA.

### 24.3.4 TX/RX Controller

The TX/RX Controller consists of a FIFO memory which buffers and regulates the frames between the system memory and the MAC. It also controls the data transferred between clock domains. Both the transmit and receive data paths are 32-bits wide. The TX FIFO and RX FIFO are each 2 KB in depth.

At reset, the TX/RX Controller is configured and ready to manage data flow to and from the DMA to the MAC. Note that the DMA and MAC must be initialized by the application out of reset.

#### 24.3.4.1 Transmit (TX) Control Path

The DMA controller is used for all Ethernet transmissions. The Ethernet frames are read from memory and transferred to the TX FIFO by the DMA. When the MAC is available, the frame is transferred from the FIFO. When the end-of-frame (EOF) is transferred, the MAC notifies the DMA the status of the transmission.

The TX FIFO has a depth of 2 KB. The FIFO fill level has the capability of triggering the DMA to initiate a burst transfer. The DMA also transfers start-of-frame (SOF), end-of-frame (EOF), CRC and pad-insertion information to the TX/RX Controller so that this information can be passed to the MAC when it is ready for transmission from the TX FIFO.

Data can be transmitted to the MAC in threshold mode or store-and-forward mode. If the TTC field is configured in the **Ethernet MAC DMA Operation Mode (EMACDMAOPMODE)** register at offset 0xC18 and the TSF bit in the same register is 0x0, then the TX Controller is operating in threshold mode. In this mode, the data is transferred to the MAC when the number of bytes in the FIFO crossed the value configured in the TTC bit field or when the end-of-frame is written before the threshold is crossed. In store-and forward mode, the TTC bit field is configured and the TSF bit is set. Data is transferred to the MAC only when one or more of the following conditions are true:

- A complete frame is stored in the FIFO
- The TX FIFO becomes almost full
- The TX FIFO does not have space to accommodate the requested burst length

With these conditions, the TX Controller continues store-and-forward mode even if the Ethernet frame length is bigger than the TX FIFO size.

The TX FIFO can be flushed of all contents by setting the FTF bit in the **EMACDMAOPMODE** register. This bit is self-clearing and initializes the FIFO pointers to the default state. If the FTF bit is set during a frame transfer from the TX Controller to the MAC, then the TX Controller stops further transfer. Early termination of the transfer causes a underflow event and this status is communicated to the DMA.

### Transmit Operation

During a transmit, single-packet or double-packets can reside in the buffer. The following describes the details of each:

- Single-packet transmit: During single packet transmission, the DMA controller fetches data from the CPU memory and forwards it to the TX FIFO and continues to receive data until the end-of-frame is transferred. The data is transmitted from the TX FIFO to the MAC by the TX/RX Controller when the threshold level is crossed or a full packet of data is received into the TX FIFO. When the TX/RX Controller receives acknowledgment from the MAC that it has received the EOF, it notifies the DMA so another transmit can begin.

- Two-packet transmit: Because the DMA must update the descriptor status before releasing it to the CPU, there can be at most two frames inside a transmit FIFO. The second frame is fetched by the DMA and put into the TX FIFO only if the OSF bit is set in the **EMACDMAOPMODE** register at offset 0xC18. If this bit is not set, the next frame is fetched from memory only after the MAC has completely processed the frame and the DMA has released the descriptors.

  If the OSF bit is set, the DMA starts fetching the second frame immediately after completing the transfer of the first frame to the FIFO. It does not wait for the status to be updated. The TX/RX Controller receives the second frame into the FIFO while transmitting the first frame. As soon as the first frame has been transferred and the status is received from the MAC, the TX/RX Controller sends the acknowledgement to the DMA. If the DMA has already completed sending the second packet to the TX/RX Controller, it must wait for the status of the first packet before proceeding to the next frame.

### Collision and Retransmission

If a collision occurs at the MAC application interface while the TX/RX Controller is transferring data to the MAC, the transmission is aborted and the MAC indicates a retry attempt by giving a collision status before the EOF is transferred to the TX/RX Controller from the DMA. This enables the TX/RX Controller to retry transmission of the frame data from the FIFO.

After more than 96 bytes are transferred to the MAC, the FIFO controller clears space in the FIFO and makes it available to the DMA to transfer more data. Retransmission is not possible after this threshold is crossed or when the MAC indicates a late collision event.

When a frame transmission is aborted because of underflow and a collision event follows, which initiates a retry, then the retry has higher priority than the abort.

### TX FIFO Flush Operation

The TX FIFO can be flushed by setting the FTF bit in the **EMACDMAOPMODE** register. The flush operation is immediate and the TX/RX Controller clears the TX FIFO and the corresponding pointers to the initial state even if it is in the middle of transferring a frame to the MAC. The data which is already accepted by the MAC transmitter is not flushed. This data is scheduled for transmission and results in an underflow event because the TX FIFO did not complete the transfer or the rest of the frame. As in all underflow conditions, a runt frame is transmitted and observed on the line. The status of such a frame is marked with both underflow and frame flush events in the Transmit Descriptor 0 (TDES0) word.

The TX/RX Controller also stops accepting any data from the DMA during the flush operation. It generates and transfers Transmit Status Words to the application for the frames that are flushed inside the FIFO, including partial frames. Frames that are completely flushed in the TX/RX Controller are identified by setting the Flush Status (FF) bit in the Transmit Descriptor 0 (TDES0) word. The TX/RX Controller completes the flush operation when the DMA accepts all of the status words for

the frames that were flushed and then clears the TX FIFO Flush control (FTF) bit in the **EMACDMAOPMODE** register. At this point, the TX/RX Controller starts accepting new frames from the DMA.

***Transmit Status Word***

At the end of the transfer of the Ethernet frame to the MAC and after the MAC completes the transmission of the frame, the TX/RX delivers a transmit status word (TDES0) to the application. If IEEE timestamping is enabled, the TX/RX Controller returns the specific frame's 64-bit timestamp, along with the transmit status word. The fields for the Transmit Descriptors are described in “Enhanced and Alternate Descriptors” on page 1598.

### 24.3.4.2 Receive (RX) Control Path

TX/RX Controller receives frames from the MAC and pushes them into the RX FIFO. When the fill level of the RX FIFO crosses the programmed RX Threshold, the DMA is notified.

***Receive Operation***

During a receive operation the TX/RX Controller is a slave to the MAC. The steps of the receive operation are as follows:

1. The MAC receives a frame. This data, along with SOF, EOF and byte enable information is sent to the TX/RX Controller. The TX/RX Controller accepts the data and pushes it into the RX FIFO. After the EOF is transferred, the MAC drives the status word, which is also pushed in to the RX FIFO.

2. When timestamp is enabled by setting the TSEN bit in the **Ethernet MAC Timestamp Control (EMACTIMSTCTRL)** register, at offset 0x700, and the 64-bit timestamp is present with the receive status, it is appended to the frame and received by the MAC and pushed into the TX FIFO before the corresponding receive status word is written. Thus, two additional locations per frame are taken for storing timestamp in the RX FIFO.

3. Data can be sent to the TX/RX Controller in cut-through mode or store-and-forward mode. When the RTC bit field of the **EMACDMAOPMODE** register is set to 0x0 and cut-through mode is enabled (RSF=0), the TX/RX Controller indicates availability to transfer to the DMA when 64 bytes are in the RX FIFO or a full packet of data has been received into the RX FIFO. When the DMA initiates transfers to system memory, the TX/RX Controller continues to transfer data from the RX FIFO until a complete packet has been transferred. When EOF has occurred, the TX/RX Controller sends the status word to the DMA.

   **Note:** The timestamp transfer takes two clock cycles and the lower 32-bits of the timestamp are sent first when timestamping is enabled.

   When the RSF bit is set in the **EMACDMAOPMODE** register, RX FIFO store-and-forward mode is enabled and a frame is read by the DMA only after it is completely written into the RX FIFO. In this mode, only valid frames are read and forwarded to the application. In cut-through mode, some error frames are not dropped because the error status is received at the end of the frame and by that time the start of that frame has already been read out of the RX FIFO.

The TX/RX Controller is capable of storing any number of frames in the RX FIFO as long as it is not full.

### *Error Handling*

If the RX FIFO is full before it receives the EOF data from the MAC, an overflow is declared, the entire frame (including the status word) is dropped and the **Ethernet MAC Missed Frame and Buffer Overflow Counter (EMACMFBOC)** register is incremented. These error actions occur even if the FEF bit is set in the **EMACDMAPOPMODE** register. If the start address of such a frame has already been transferred to the TX/RX Controller, the rest of the frame is dropped and a dummy EOF is written to the FIFO along with its status word. The descriptor status indicates a partial frame because of overflow. In such frames, the Frame Length (FL) field in the receive descriptor is invalid. If the RX FIFO is configured to operate in the store-and-forward mode and if the length of the received frame is more than the FIFO size, overflow occurs and all such frames are dropped. During error handling, the DMA flushes the error frame currently being read.

The Receive control logic can filter error and undersized frames if enabled through configuring the FEF or FUF bit of the **EMACDMAOPMODE** register. Filtering must be set before the start address of the frame has been transferred to the TX/RX controller for it to take effect.

### *Receive Word Status*

At the end of an Ethernet frame transfer to the system memory, the TX/RX Controller sends a receive status word, RDES0, to the application. Until the end of the frame transfer, the TX/RX Controller stores the status and frame length in an asynchronous status FIFO whose depth is determined by the size of the RX FIFO (2K) and the minimum size of the frame. If the frame size if 64, then the asynchronous FIFO depth is 2048/64 = 32 bytes in length. Note that when the status of a partial frame (because of overflow) is sent to the application, the Frame Length field of RDES0 is not valid and is set to zero.

**Note:** When the timestamp feature is enabled, the receive status field is greater than 32-bits. An extended status bit-field [63:32] provides information about the received Ethernet payload when it is carrying PTP packets or TCP/UDP/ICMP over IP packets. Since the data bus is 32 bits, the status is transferred over two clock cycles.

## 24.3.4.3 MAC Flow Control

Flow control mechanisms can be enabled for both the TX and RX FIFO datapath, depending on the configurations in the **Ethernet MAC Flow Control (EMACFLOWCTL)** register at offset 0x018 and the DUPM bit configuration in the **Ethernet MAC Configuration (EMACCFG)** register at offset 0x000.

**Table 24-17. TX MAC Flow Control**

| TFE bit in EMACFLOWCTL | DUPM bit in EMACCFG | Description |
|---|---|---|
| 0 | X | The MAC transmitter does not perform the flow control or backpressure operation. |
| 1 | 0 | The MAC transmitter performs backpressure when the FCBBPA bit in the **Ethernet MAC Flow Control (EMACFLOWCTL)** register is set. |
| 1 | 1 | The MAC transmitter sends a pause frame when the FCBBPA bit in the **Ethernet MAC Flow Control (EMACFLOWCTL)** register is set. |

**Table 24-18. RX MAC Flow Control**

| TFE bit in EMACFLOWCTL | DUPM bit in EMACCFG | Description |
|---|---|---|
| 0 | X | The MAC receiver does not detect the received Pause frames. |
| 1 | 0 | The MAC receiver does not detect the received Pause frames but recognizes such frames as Control frames. |

**Table 24-18. RX MAC Flow Control *(continued)***

| TFE bit in EMACFLOWCTL | DUPM bit in EMACCFG | Description |
|---|---|---|
| 1 | 1 | The MAC receiver detects or processes the Pause frames and responds to such frames by stopping the MAC transmitter. |

## 24.3.5 MAC Operation

The MAC module enables the CPU to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard. The MAC supports the interface to the PHY and is comprised of a receive and transmit module whose features are described in the following sections.

### 24.3.5.1 MAC Transmit Module

MAC transmission is initiated when the TX/RX Controller transmits data with the start of frame (SOF) signal asserted. When the SOF signal is detected, the MAC accepts the data and begins transmitting to the PHY. The time required to transmit the frame data after the application initiates transmission varies, depending on delay factors like inter-frame gap (IFG) delay, time to transmit preamble or start of frame data (SFD), and any back-off delays for half-duplex mode. Until then, the MAC does not accept data received from the TX/RX Controller.

After the end-of-frame (EOF) is transferred to the MAC, the MAC completes the normal transmission and gives the status of transmission to the TX/RX Controller. If a normal collision (in half-duplex mode) occurs during transmission, the MAC conveys the transmit status to the TX/RX Controller. It then accepts and drops all further data until the next SOF is received. The TX/RX Controller should retransmit the same frame from SOF on observing a retry request (in the transmit status word) from the MAC. The MAC issues an underflow status if the TX/RX Controller is not able to provide the data continuously during the transmission. During the normal transfer of a frame from the TX/RX Controller, if the MAC receives an SOF without getting an EOF for the previous frame, it ignores the SOF and considers the new frame as a continuation of the previous frame.

If the number of bytes received from memory is less than 60 bytes, zeros are automatically appended to the transmitting frame to make the data length exactly 46 bytes to meet the minimum data field requirement of IEEE802.3.

The transmit engine controls the operation of Ethernet frame transmission. Some of the functions of the transmit module include:

- Output of (32-bit) Transmit Status (TDES0) to the application at the end of normal transmission or collision
- Generating preamble and Start of Frame Data (SFD)
- Generating jam pattern in half-duplex mode
- Supporting Jabber time-out
- Supporting flow control
- Generating timestamp information for transmission
- Scheduling frame transmission to satisfy inter-frame gap (IFG) and back-off delays
- Generating CRC and FCS field for Ethernet Frame
- Generating pause frames as necessary in full duplex mode

When a new frame transmission is requested, the MAC transmit module sends out a preamble and SFD, followed by the data in the TX FIFO. The preamble is defined as 7 bytes of 0xAA and the SFD is defined as 1 byte of 0xAB pattern.

The collision window is defined as 1 slot time (512-bit times). If a collision occurs any time from the beginning of the frame to the end of the CRC field, the MAC transmit module sends a 32-bit jam pattern of 0x5555.5555 to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the MAC transmit module completes the transmission of the preamble and SFD, and then sends the jam pattern. If the collision occurs after the collision window and before the end of the FCS field, the MAC transmit module sends a 32-bit jam pattern and sets the late collision bit in the transmit frame status. The jam pattern generation is applicable only to half-duplex mode.

A jabber timer is enabled by default at reset in the MAC module. If the JD bit in the **EMACCFG** register at offset 0x000 is clear and the MAC module has transferred more than 2K bytes, the jabber timer causes the MAC module to stop transmission of Ethernet frames. The timeout is changed to 10 KB when the Jumbo Frame Enable (JFEN) bit is enabled in the **EMACCFG** register.

The MAC transmit module uses a deferral mechanism for flow control (back pressure) in half-duplex mode. When the application requests to stop receiving frames, the transmit module sends a jam pattern of 32 bytes whenever it senses the reception of a frame, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a frame to be transmitted, the frame is scheduled and transmitted even when backpressure is activated. If the backpressure is activated for a long time (more than 16 consecutive collision events occur) then the remote stations abort their transmissions because of excessive collisions.

If IEEE 1588 timestamping is enabled for the transmit frame, the MAC transmit module takes a snapshot of the system time when the start-of-frame data is output on the bus.

#### 24.3.5.2 MAC Transmit Module CRC Generator

The Transmit CRC Generator is used to generate the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following generating polynomial:

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1}$.

The calculated CRC is valid on the next clock after the data is received.

#### 24.3.5.3 MAC Receive Module

A receive operation is initiated when the MAC detects start-of-frame data (SFD). The MAC strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field is used to verify the CRC for the frame. The received frame is stored in a buffer until the address filtering is performed. The frame is dropped in the MAC if it fails the address filter.

Some of the functions of the MAC receive module are as follows:

- Stripping the preamble, SFD and carrier extension of the Ethernet frame
- Providing IEEE 1588 timestamping whenever an SFD is detected
- Converting receive nibble data into bytes in MII mode
- Decoding Length/Type field of the Ethernet frame
- Auto-CRC/pad stripping if enabled in the **EMACCFG** register

- Frame data and status buffering for received frames
- Data path conversion of 8-bit data to 32-bit data
- Frame filtering
- Attaching calculated IP checksum input to frame
- Detecting receiving pause frames and pausing the frame transmission for the delay specified withing the received pause frame
- Receive IP Checksum Verification
- Performing destination and source address checking functions on all received frames and reporting the address filtering status to the receive frame controller module.

#### 24.3.5.4 MAC Receive Module CRC Generator

The receive CRC Generator is used to generate the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following generating polynomial:

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The calculated CRC is valid on the next clock after the data is received.

#### 24.3.5.5 MAC Receive Frame Controller

If the RA bit is set in the **Ethernet MAC Frame Filter (EMACFRAMEFLTR)** register, offset 0x004, the MAC Receive Frame Controller initiates the data transfer to the RX FIFO as soon as four bytes of Ethernet data are received. At the end of the data transfer, the received frame status that includes the frame filter bits (SA or DA filter fail) and status are also sent. These bits indicate to the application whether the received frame has passed the filter controls. This module does not drop any frame on its own in this mode.

If the RA bit is clear, the MAC Receive Frame Controller performs frame filtering based on the destination/source address (the application still needs to perform another level of filtering if it decides not to receive any bad frames like runt, CRC error frames, etc.) After receiving the destination or source address bytes, the MAC Receive Frame Controller checks the filter-fail sign for an address match. On detecting a filter-fail, the frame is dropped and not transferred to the application.

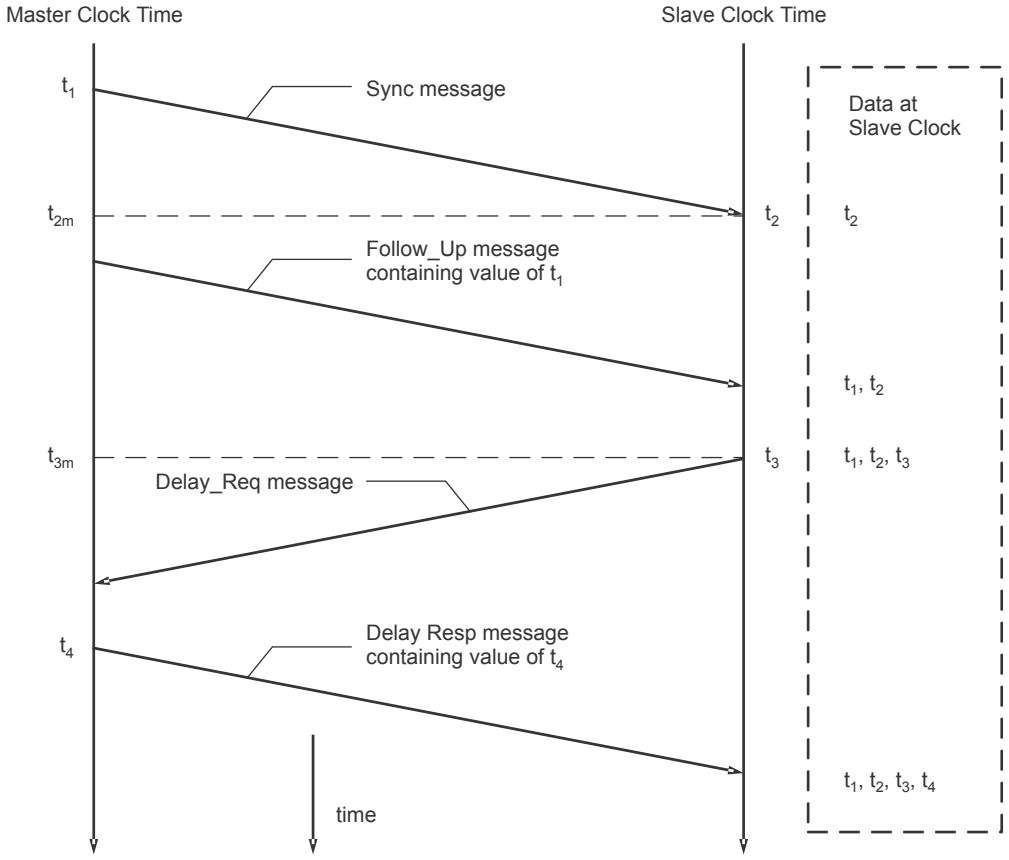
**Note:** When the PMT module is configured for power-down mode, all received frames are dropped and not forwarded to the application.

### 24.3.6 IEEE 1588 and Advanced Timestamp Function

The MAC module supports the IEEE 1588-2002 Timestamp Precision Time Protocol (PTP) and the IEEE 1588-2008 Advanced Timestamp features. PTP enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by a local area network supporting multicast messaging. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The PTP is transported over UDP/IP. The system or network is classified into master and slave nodes for distributing the timing and clock information. Figure 24-10 on page 1626 shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

**Figure 24-10. Networked Time Synchronization**




As shown in Figure 24-10 on page 1626, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is t1. This time is captured at the MII interface.
2. The slave receives the Sync message and also captures the exact time, t2, using its timing reference.
3. The master sends a Follow_Up message to the slave, which contains t1 information for later use.
4. The slave sends a Delay_Req message to the master, noting the exact time, t3, at which this frame leaves the MAC.
5. The master receives the message, capturing the exact time, t4, at which it enters its system.
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet

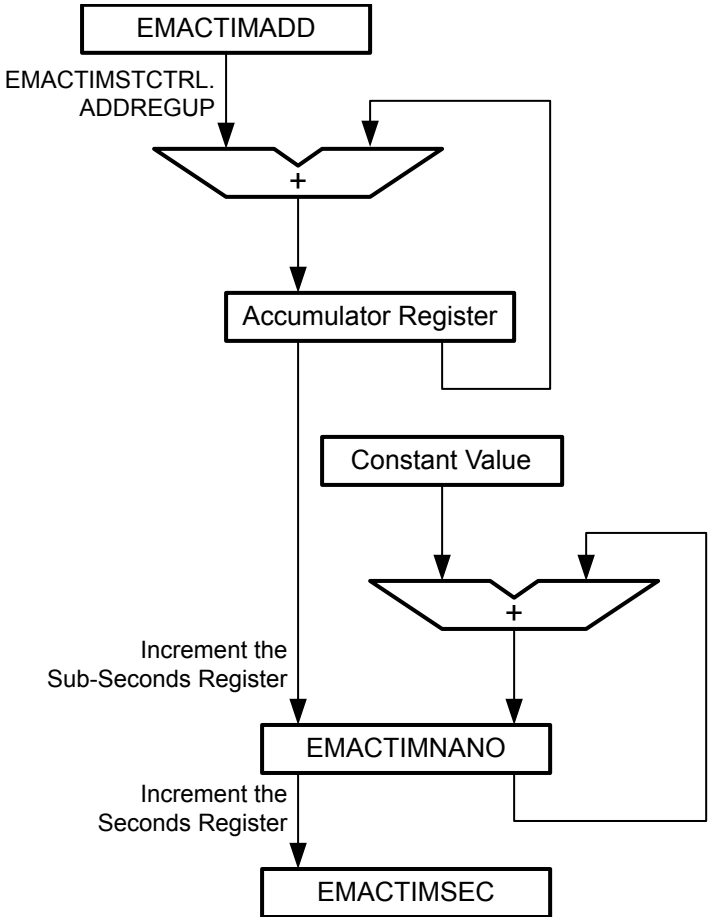
MAC. This timing information is captured and returned to the software for the proper implementation of PTP with high accuracy.

#### 24.3.6.1 System Time Module

The System Time module maintains a 64-bit time and is updated using the MOSC clock source as the PTP clock reference. This time is the source for taking snapshots (timestamps) of the Ethernet frames being transmitted or received. Two methods of updating the system time counter are implemented. The counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the **MAC System Time - Seconds Update (EMACTIMSECU)** register along with the **MAC System Time - Nanoseconds Update (EMACTIMNANOU)** register. For initialization the system time counter is written with the value in these registers, while for system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, the slave clock's frequency drift with respect to the master clock is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the **EMACTIMADD** register, as shown in Figure 24-11 on page 1628. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high precision frequency multiplier or divider.

**Figure 24-11. System Time Update Using Fine Correction Method**




**Note:** If the Ethernet Controller is configured to use the MII/RMII interface to an external PHY, then the MOSC clock that feeds the PTP reference clock to the System Time Module has a minimum frequency requirement of 5 MHz and a maximum frequency of 25 MHz. For course correction methods, the value of MOSC can be anywhere within this range, but for the fine correction method, a 25 MHz value MOSC crystal should be used for the best accuracy. If the Ethernet Controller is configured to use the MII interface connected to the integrated PHY, then the MOSC clock that feeds the PTP reference clock to the System Time Module must be 25 MHz because it also clocks the integrated PHY module.

Initially, the Ethernet's slave clock (from the MOSC) is adjusted with a compensation value (as described in the previous paragraph) which is written to the Timestamp Addend Register (TSAR) field in the **EMACTIMADD** register. This value is calculated as: $FreqCompensationValue_0$= TSAR = $2^{32}$/ FreqDivisionRatio.

The System Time Module requires a 20-MHz PTP reference clock frequency to achieve 50-ns accuracy in the fine correction method. An addend must be written to the **Ethernet MAC Time Stamp Addend (EMACTIMADD)** register, offset 0x718 to achieve timing synchronization. If the MOSC clock source is 25 MHz, the frequency division ratio (FreqDivisionRatio) of the two is calculated as 25 MHz / 20 MHz = 1.25. Hence, the default addend value to be set in the register is $2^{32}$ / 1.25 or 0xCCCC.CCD0. If the reference clock drifts lower, to 24 MHz for example, the ratio is 24 / 20, or 1.2 and the value to set in the addend register is $2^{32}$ / 1.20, or 0xDFF1.65D2. The software must

calculate the drift in frequency based on the Sync messages and update the **EMACTIMADD** register, at offset 0x718, accordingly.

If the master to slave delay is initially assumed to be the same for consecutive Sync messages, then the following steps can be used to calculate a new TSAR value. The following algorithm calculates the precise mater to slave delay value to allow for re-synchronization with the master using the new value:

1. At time $MasterSyncTime_n$ the master sends the slave clock a Sync message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as:

   $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$

2. The master clock count, $MasterClockCount_n$, for the current Sync cycle is given by:

   $MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$

   This assumes that the MasterToSlaveDelay is the same for Sync cycles n and n-1.

3. The slave clock count for current Sync cycle, $SlaveClockCount_n$ is given by:

   $SlaveClockCount_n = SlaveClockTime_n – SlaveClockTime_{n-1}$

4. The difference between the master and slave clock counts for the current Sync cycle, $ClockDiffCount_n$, is given by:

   $ClockDiffCount_n = MasterClockCount_n – SlaveClockCount_n$

5. The frequency-scaling factor for the slave clock, $FreqScaleFactor_n$ is given by:

   $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$

6. The frequency compensation value, FreqCompensationValue, to be written in the TSAR field of the EMACTIMADD register is:

   $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one Sync cycle; however, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm corrects it at the cost of more Sync cycles.

#### 24.3.6.2 Transmit Timestamping

The MAC captures a timestamp when the Start Frame Delimiter (SFD) of a frame is sent. The transmit frames are marked to indicate whether a timestamp should be captured for that frame and written to the extended transmit descriptors that support timestamping. The MAC returns the timestamp automatically to the corresponding transmit descriptor, thus connecting the timestamp with the specific PTP frame. The 64-bit timestamp information is written to the TDES6 and TDES7 fields.

#### 24.3.6.3 Receive Timestamping

The MAC captures the timestamp of all received frames. The MAC does not process the received frames to identify the PTP frames in default timestamping mode (when Advanced Timestamp is disabled). The MAC gives the timestamp and the corresponding status to the TX/RX Controller along with the EOF data. The TX/RX Controller validates and indicates the availability of the timestamp so that the DMA can return the timestamp to the corresponding receive descriptor. The

64-bit timestamp information is written to the RDES6 and RDES7 fields. The timestamp is written only to the receive descriptor for which the Last Descriptor status field has been set to 1 (the EOF marker). When the timestamp is not available (for example, because of an RX FIFO overflow), an all '1s' pattern is written to the descriptors (RDES6 and RDES7), indicating that the timestamp is not correct. If timestamping is disabled, the DMA does not alter RDES6 or RDES7. RDES0[7] indicates whether the timestamp is updated in RDES6 and RDES7.

#### 24.3.6.4 Timestamp Error Margin

According to the IEEE 1588 specifications, a timestamp must be captured at the SFD of the transmitted and received frames at the MAC interface. Because the reference timing source, MOSC, is taken as different from MAC reference clocks, a small error margin is introduced, because of the transfer of information across asynchronous clock domains. In the transmit path, the captured and reported timestamp has a maximum error margin of 2 PTP (MOSC) clocks. It means that the captured timestamp has the reference timing source (MOSC) value that is given within 2 clocks after the SFD has been transmitted to the PHY. Similarly, in the receive path, the error margin is 3 MAC reference clocks, plus up to 2 PTP clocks. The error margin of the three MAC reference clocks can be ignored by assuming that this constant delay is present in the system (or link) before the SFD data reaches the interface of MAC.

**Note:** When the Ethernet Controller is configured to use the MII interface to an external PHY, the MII clock is provided by the external PHY through EN0RXCK and EN0TXCK. When the Ethernet Controller is connected to the integrated PHY, the reference clock must be 25 MHz because it is also the source to the PHY.

**Note:** When IEEE 1588 timestamping is enabled with internal timestamp, use a PTP clock frequency that is greater than 5 MHz. This is because the SSINC field in the **EMACSUBSECINC** register limits the PTP frequency that can be used to ~4 MHz.

#### 24.3.6.5 IEEE 1588-2008 Advanced Timestamps

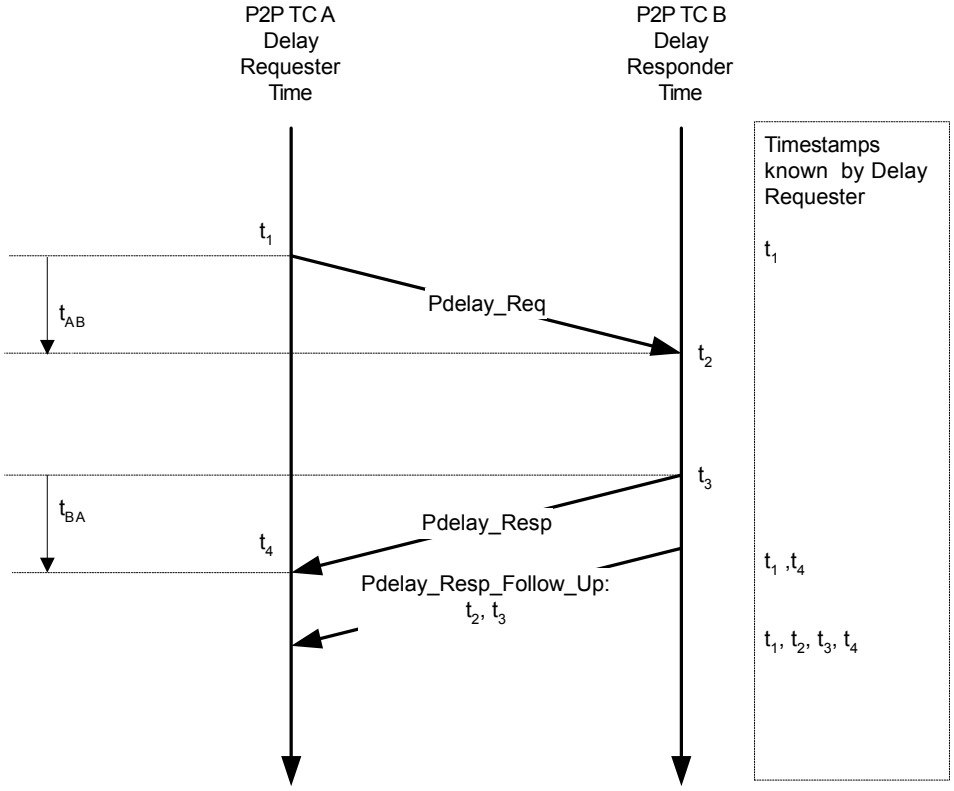
In addition to the basic timestamp features mentioned in IEEE 1588-2002 Timestamps, the Ethernet Controller supports the following advanced timestamp features defined in the IEEE 1588-2008 standard:

- Supports the IEEE 1588-2008 (version 2) timestamp format.
- Provides an option to take snapshot of all frames or only PTP type frames.
- Provides an option to take snapshot of only event messages.
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end, and peer-to-peer.
- Provides an option to select the node to be a master or slave clock.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to measure sub-second time in digital or binary format.

##### *Peer-to-Peer Transparent Clock Message Support*

The IEEE 1588-2008 version supports Peer-to-Peer PTP (Pdelay) message in addition to SYNC, Delay Request, Follow-up, and Delay Response messages. Figure 24-12 on page 1631 shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

**Figure 24-12. Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction**




As shown in Figure 24-12 on page 1631, the propagation delay is calculated in the following way:

1. Port-1 issues a Pdelay_Req message and generates a timestamp, t1, for the Pdelay_Req message.
2. Port-2 receives the Pdelay_Req message and generates a timestamp, t2, for this message.
3. Port-2 returns a Pdelay_Resp message and generates a timestamp, t3, for this message.

   To minimize errors because of any frequency offset between the two ports, Port-2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. The Port-2 returns any one of the following:

   - The difference between the timestamps t2 and t3 in the Pdelay_Resp message.
   - The difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message.
   - The timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.
4. Port-1 generates a timestamp, t4, on receiving the Pdelay_Resp message.
5. Port-1 uses all four timestamps to compute the mean link delay.

### *Advanced Timestamp Supported Clock Types*

The Advance Timestamp Module supports an ordinary clock as defined by the IEEE 1588-2008 standard. The characteristics of this clock is as follows:

- Sending and receiving of PTP messages. The timestamp snapshot can be controlled through the **Ethernet MAC Timestamp Control (EMACTIMSTCTRL)** register, offset 0x700. Timestamp snapshots can be for Sync messages.
- Maintaining the data sets such as timestamp values.

  For an ordinary clock, snapshots can be taken of either version 1 or version 2 PTP types but not both. Selecting between the two is controlled by the PTPVER2 bit of the **EMACTIMSTCTRL** register.

### *PTP Processing and Control*

There are fields in an Ethernet Payload that the Ethernet Controller can use to detect the PTP packet type and control the snapshot to be taken. These fields are different depending on whether the PTP frames are:

- PTP frames over IPv4
- PTP frames over IPv6
- PTP frames over Ethernet

The PTPIPV4, PTPIPV6 or PTPETH bits can be set depending on which PTP processing is required.

### *Reference Timing Source*

The MAC supports the following reference timing source features defined in the IEEE 1588-2008 standard:

- 80-bit timestamp
- Fixed Pulse-Per-Second PPS0 Output
- Flexible Pulse-Per-Second PPS0 Output

### *80-Bit Timestamp*

The MAC supports an 80-bit timestamp with a lengthened seconds integer portion which is 48-bits wide. The **Ethernet MAC System Time - Seconds (EMACTIMSEC)** register at offset 0x708 and the **Ethernet MAC System Time-Higher Word Seconds (EMACHWORDSEC)** register at offset 0x724 comprise the seconds count. The 32-bit **Ethernet MAC System Time - Nanoseconds (EMACTIMNANO)** register contains the fractional portion of the timestamp units of nanoseconds. The nanoseconds register supports the following two modes:

- Digital rollover mode: In digital rollover mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.
- Binary rollover mode: In binary rollover mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit.

Digital or binary rollover mode can be selected by programming the DGTLBIN bit of the **EMACTIMSTCTRL** register. Note that the timestamp maintained in the MAC is still 64-bits wide, but that the overflow to the upper 16-bits seconds register happens once in 130 years.

### *Fixed Pulse-Per-Second Output*

The Ethernet MAC module supports a pulse-per-second output feature such that the signal, EN0PPS, can indicate one second intervals. The frequency of the EN0PPS output can be changed by setting the PPSCTRL bit field of the **Ethernet MAC PPS Control (EMACPPSCTRL)** register, offset 0x72C.

### *Flexible Pulse-Per-Second Output*

The Ethernet MAC also provides the ability to control the following features of the EN0PPS output:

- Start or stop time
- The start point of a single pulse and the start and stop points of the pulse train in terms of a 64-bit system time. The **EMACTIMESEC** and **EMACTIMNANO** registers are used to program the start and stop time.
- The stop time can be programmed in advance of starting.
- The signal width. The rising edge and the corresponding falling edge of the EN0PPS output can be programmed in terms of number of units of sub-second increment value programmed in the **Ethernet MAC Sub-Second Increment (EMACSUBSECINC)** register, offset 0x704.
- The signal interval. The time between the rising edges of EN0PPS signal, in terms of number of units of sub-second increment value can be programmed in the **Ethernet MAC PPS0 Interval (EMACPPS0INTVL)** register, offset 0x760 . You can program the interval between pulses from 1 to $2^{32}$-1 units of sub-second increment value.
- Cancellation of the programmed EN0PPS start or stop request.
- Error indication if the start or stop time being programmed has already passed.

The start time can be programmed in the **EMACTARGSEC** and **EMACTARGNANO** registers. The TRGTBUSY bit in the **EMACTARGNANO** register indicates when the value is synchronized to the PTP clock domain. When this bit is clear, a new start time can be programmed, even before the earlier start time has elapsed. The start or stop time should be programmed with advanced system time to ensure proper EN0PPS signal output. If the application programs a start or stop time that has already elapsed, then the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

For a flexible EN0PPS output, the **EMACPPS0INTVL** and **EMACPPS0WIDTH** registers can be configured. The PPS0WIDTH and PPS0INT fields are programmed in terms of granularity of system time, that is, number of the units of sub-second increment value. For example, to have a EN0PPS pulse width of 80 ns and interval of 120ns, with the PTP reference clock of 25MHz, you should program the width and interval to values 1 and 2, respectively. Note that the PPS0WIDTH and PPS0INT value must be programmed as one less than the required interval. Before giving the command to trigger a pulse or pulse train on the EN0PPS output, the interval and width of the PPS signal output should be programmed or updated.

### *Advanced Timestamp Transmit Path Functions*

The only aspect of the transmit path that changes with advanced timestamp is the descriptor, which extends to 32-bytes long.

When you enable the advanced timestamp feature, the structure of the descriptor changes. The advanced timestamp feature is supported only through the enhanced descriptors format. The descriptor is 32-bytes long (eight words) and the snapshot of the timestamp is written in descriptor TDES6 and TDES7.

***Advanced Timestamp Receive Path Functions***

When the advanced timestamp feature is enabled, the MAC processes the received frames to identify valid PTP frames. The snapshot of the time sent to the application can be configured to:

- Enable snapshot for all frames
- Enable snapshot for IEEE 1588 version 2 or version 1 timestamp
- Enable snapshot for PTP frames transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received frame for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type. This controls the type of messages for which snapshots are taken

The MAC provides the timestamp, along with EOF to the TX/RX Controller. The DMA returns the timestamp inside the corresponding receive descriptor.

## 24.3.7 Frame Filtering

The following types of filtering are available for receive frames:

- Source Address (SA) or Destination Address (DA) filtering
- VLAN Filtering

The frame filtering supports a sequence where the packet is not forwarded to VLAN filtering if it does not pass the SA or DA filtering first.

### 24.3.7.1 VLAN Filtering

The Ethernet MAC provides VLAN Tag Perfect Filtering and VLAN Tag Hash Filtering. In VLAN tag perfect filtering, the MAC compares the VLAN tag of the received frame and provides the VLAN frame status to the application. Based on the programmed mode of the ETV bit in the **EMACVLANTG** register, the MAC compares the lower 12 bits or all 16 bits of the received VLAN tag to determine the perfect match. If VLAN tag perfect filtering is enabled, the MAC forwards the VLAN-tagged frames along with VLAN tag match status and drops the VLAN frames that do not match. Inverse matching for VLAN frames can also be enabled by setting the VTIM bit of the **Ethernet MAC VLAN Tag (EMACVLANTG)** register, offset 0x01C. In addition, matching of S-VLAN tagged frames along with the default C-VLAN tagged frames can be enabled by setting the ESVL bit of the **EMACVLANTG** register. The VLAN frame status bit (Bit 10 of RDES0) indicates the VLAN tag match status for the matched frames.

**Note:** The Source or Destination Address filter has precedence over the VLAN tag filters. A frame which fails the Source or Destination Address filter is dropped irrespective of the VLAN tag filter results.

The MAC provides VLAN tag hash filtering with a 16-bit Hash table. The MAC performs the VLAN hash matching based on the VTHM bit of the **EMACVLANTG** register. If the VTHM bit is set, the most significant four bits of VLAN tag's CRC-32 are used to index the content of the **Ethernet MAC VLAN Hash Table (EMACVLANHASH)** register, offset 0x588. A value of 1 in the **EMACVLANHASH** register, corresponding to the index, indicates that the VLAN tag of the frame matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged frame should be dropped. The MAC also supports the inverse matching of the VLAN frames. In the inverse matching mode, when the VLAN tag of a frame matches the perfect or hash filter, the packet should be dropped. If the VLAN perfect and VLAN hash match are enabled, a frame is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and hash filters indicate mismatch. Table 24-19 on page 1635 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of the **EMACFRAMEFLTR** register is set, all frames are received and the VLAN match status is indicated in Bit 10 of Receive Descriptor word 0 (RDES0). When the RA bit is not set and the VTFE bit in the **EMACFRAMEFLTR** register is set, the frame is dropped if the final VLAN match status is fail. In Table 24-19 on page 1635, value X means that this column can have any value. When the VL field is programmed to 0x0 in **EMACVLANTG** register, all VLAN-tagged frames are considered as perfect matched but the status of the VLAN hash match depends on the VLAN hash enable (VTHM) bit and VLAN inverse filter (VTIM) bit.

**Table 24-19. VLAN Match Status**

| VLAN ID (VL field) | VLAN Perfect Filter Match Status (VPF) | VLAN HASH Enable bit (HPF bit in EMACFRAMEFLTR) | VLAN Hash Filter Match Status (VTHM) | VLAN Inverse Filter Bit (VTIM) | Final VLAN Match Status |
|---|---|---|---|---|---|
| VL = 0 | Pass | 0 | X | X | Pass |
| | Pass | 1 | X | 0 | Pass |
| | Pass | 1 | Fail | 1 | Pass |
| | Pass | 1 | Pass | 1 | Fail |
| VL !=0 | Pass | X | X | 0 | Pass |
| | Fail | 0 | X | 0 | Fail |
| | Fail | 1 | Fail | 0 | Fail |
| | Fail | 1 | Pass | 0 | Pass |
| | Fail | 0 | X | 1 | Pass |
| | Pass | X | X | 1 | Fail |
| | Fail | 1 | Pass | 1 | Fail |
| | Fail | 1 | Fail | 1 | Pass |

## 24.3.8 Source Address, VLAN, and CRC Insertion, Replacement or Deletion

The MAC supports the following functions for transmit frames:

- Source Address Insertion or replacement
- VLAN Insertion, Replacement or Deletion
- CRC Replacement

### 24.3.8.1 Source Address Insertion or Replacement

Software can use the SA insertion or replacement feature to instruct the MAC to do the following for transmit frames:

- Insert the content of the MAC Address Registers in the SA field.
- Replace the content of the SA field with the content of the MAC Address Registers.

The software can enable the SA insertion or replacement for all transmitted frames or selective frames:

- To enable SA insertion or replacement feature for all frames, program the SADDR field of the **Ethernet MAC Configuration (EMACCFG)** register.
- To enable SA insertion or replacement for selective frames, program the SA Insertion Control field (TDES1 Bits [31:29]) in the first transmit descriptor of the frame. When Bit 31 of TDES1 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When Bit 31 of TDES1 is reset, it indicates insertion or replacement by MAC Address 0 registers.

When SA insertion is enabled, the application should ensure that the frames that are sent to the MAC do not have the SA field. The MAC does not check the presence of SA field in the transmit frame and just inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application should ensure that the frames that are sent to the MAC have the SA field. The MAC just replaces the six bytes, following the Destination Address field in the transmit frame, with the content of the MAC Address Registers.

#### 24.3.8.2 VLAN Insertion, Replacement or Deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for transmit frames:

- Insert or replace the VLAN Type field (C-VLAN or S-VLAN indicated by the CSVL bit of the **Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP)**, MAC offset 0x584) and the VLAN Tag field in the transmit frame with the VLT field of the **EMACVLNINCREP** register.
- Delete the VLAN Type and VLAN Tag fields in the transmit frame.

The software can enable the VLAN insertion, replacement, or deletion feature for all transmitted frames or selective frames. To enable this function for all transmit frames, configure the VLT field in the **EMACVLNINCREP** register.

When VLAN replacement or deletion is enabled, the MAC checks the presence of the VLAN Type field (0x8100 or 0x88a8), after the Destination address (DA) and SA fields, in the transmit frame. The replace or delete operation does not occur if the VLAN Type field is not detected in the two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the transmit frame and just inserts the VLAN Type and VLAN Tag fields.

#### 24.3.8.3 CRC Replacement

The software can use the CRC replacement feature to instruct the MAC to replace the FCS field in the transmit frame with the CRC computed by the MAC. This feature works on a per-frame basis. The CRC replacement control field in the Transmit Descriptor Word 0 (TDES0) can be programmed to enable this for a frame. This feature is valid only when the Disable CRC control (Bit 27 of TDES0) is enabled. If SA or VLAN insertion control is enabled, the MAC appends or replaces the FCS field with the computed CRC when Disable CRC Control is enabled or disabled, respectively.

**Table 24-20. CRC Replacement Based on Bit 27 and Bit 24 of TDES0**

| Bit 27 (DC) | Bit 24 (CRCR) | Description |
|---|---|---|
| 0 | X | Append CRC<br>When DC= 0, the MAC appends the computed CRC irrespective of the CRCR setting. |
| 1 | 1 | Replace CRC |
| 1 | 0 | No operation (user has appended CRC) |

## 24.3.9 Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the MAC Checksum Offload Engine (COE) supports checksum calculation and insertion in the transmit path, and error detection in the receive path.

### 24.3.9.1 Transmit Checksum Offload Engine

The checksum for TCP, UDP, or ICMP is calculated over a complete frame, and then inserted into its corresponding header field. Because of this requirement this function is enabled only when the TX FIFO is configured for the store-and-forward mode (the TSF bit is set in the **EMACDMAOPMODE** register).

**Note:** The TX FIFO must be deep enough to store a complete frame before the frame is transferred to the MAC when the checksum offload is being used. If space is not available to accept the programmed burst length of data, the TX/RX Controller starts reading to avoid deadlock. When reading starts, the checksum offload fails and the consequently all succeeding frames may be corrupted because of improper recovery. Therefore, checksum insertion must only be enabled in frames that are less than [2048-((PBL+3)*4)] bytes in size, where PBL is the Programmable Burst Length field in the **EMACDMABUSMOD** register.

### 24.3.9.2 IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The checksum offload engine detects an IPv4 datagram when the Ethernet frame's Type field has the value 0x0800 and the IP datagram's Version field has the value 0x4. The input frame's checksum field is ignored during calculation and replaced with the calculated value. IPv6 headers do not have a checksum field. Therefore, the checksum offload does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 16 in TDES0). This status bit is set whenever the values of the Ethernet Type field and the IP header's Version field are not consistent, or when the Ethernet frame does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the IP header's Version field is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total frame length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:

– The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6.

– The frame ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

If the checksum offload engine detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

#### 24.3.9.3 Receive Checksum Offload Engine

Both IPv4 and IPv6 frames in the received Ethernet frames are detected and processed for data integrity. The receive checksum feature can be enabled by setting the `IPC` bit of the **Ethernet MAC Configuration (EMACCFG)** register. The EMAC receiver identifies IPv4 or IPv6 frames by checking for value 0x0800 or 0x86DD, respectively, in the received Ethernet frames' Type field. This identification also applies to single VLAN-tagged frames. The offline receive checksum engine calculates IPv4 header checksums and checks that they match the received IPv4 header checksums. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received frame does not have enough bytes, as indicated by the Length field of the IPv4 header or when fewer than 20 bytes are available in an IPv4 or IPv6 header. This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

### 24.3.10 MAC Management Counters

The MAC Management Counters (MMC) module maintains a set of registers for gathering statistics on the received and transmitted frames. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (one for receive and one for transmit), and two 32-bit registers containing masks for the Interrupt register (one for receive and one for transmit).The MMC counters are free running and start counting when a corresponding frame is received or transmitted. The MMC counter registers provided are as follows:

- **Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB)**
- **Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL)**
- **Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL)**
- **Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG)**
- **Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB)**
- **Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR)**
- **Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR)**
- **Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI)**

### 24.3.11 Power Management Module

The power management (PMT) module supports the reception of network remote wake-up frames and AMD Magic Packet™ frames. The PMT module does not perform the clock gate function, but generates interrupts for remote wake-up frames and magic packets that the MAC receives.

When the application enables the power-down mode in the PMT module by setting the PWRDWN bit in the **Ethernet MAC PMT Control and Status Register (EMACPMTCTLSTAT)** register, MAC offset 0x02C, the MAC drops all received frames and does not forward any frame to the TX/RX Controller RxFIFO or the application. The MAC comes out of the power-down mode only when a magic packet or a remote wake-up frame is received and the corresponding detection is enabled.

#### 24.3.11.1 Remote Wake-Up

The Remote Wake-Up register bank is made up of eight 32-bit registers. It is loaded by writing the **Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF)** register eight times. To load values in the **EMACRWUFF** register, the entire register must be written. The first write is assigned to register 0 of the bank, then register 1 and so on. The **Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF)** register is read the same way. The current pointer value of the bank is updated in the Remote Wake-Up FIFO Pointer (RWKPTR) field of the **Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT)** register.

**Figure 24-13. Wake-Up Frame Filter Register Bank**

<table>
<tr><td colspan="8">Filter 0 Byte Mask</td></tr>
<tr><td colspan="8">Filter 1 Byte Mask</td></tr>
<tr><td colspan="8">Filter 2 Byte Mask</td></tr>
<tr><td colspan="8">Filter 3 Byte Mask</td></tr>
<tr><td>RSVD</td><td>Filter 3 Command</td><td>RSVD</td><td>Filter 2 Command</td><td>RSVD</td><td>Filter 1 Command</td><td>RSVD</td><td>Filter 0 Command</td></tr>
<tr><td colspan="2">Filter 3 Offset</td><td colspan="2">Filter 2 Offset</td><td colspan="2">Filter 1 Offset</td><td colspan="2">Filter 0 Offset</td></tr>
<tr><td colspan="4">Filter 1 CRC - 16</td><td colspan="4">Filter 0 CRC - 16</td></tr>
<tr><td colspan="4">Filter 3 CRC - 16</td><td colspan="4">Filter 2 CRC - 16</td></tr>
</table>

***Filter n Byte Mask***

The Filter n Byte Mask registers of the Remote Wake-Up register define the bytes of the frame that are examined by filter n (0, 1, 2, and 3) in order to determine whether or not a frame is a remote wake-up frame. The most significant bit (bit 31) of each mask must be zero. Bits [30:0] are the Byte Mask. If bit, j, of the Byte Mask is set, then the CRC block processes the Filter n Offset + j of the incoming frame; otherwise Filter n Offset + j is ignored.

***Filter n Command***

The 4-bit Filter n Command field controls the Filter n operation in the following way:

- Filter n Command Bit 3 specifies the address type of the pattern. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame.
- Filter n Command Bit 2 and Bit 1 are reserved.
- Filter n Command Bit 0 is the enable for Filter n. If bit 0 is not set, filter n is disabled.

***Filter n Offset***

The Filter n Offset register defines the offset (within the frame) from which the filter n examines the frames. This 8-bit pattern offset is the offset for the filter n first byte to be examined. The minimum allowed offset is 12, which refers to the 13th byte of the frame. The offset value 0 refers to the first byte of the frame.

***Filter n CRC-16***

The Filter n CRC-16 register contains the CRC_16 value calculated from the pattern and the byte mask programmed to the wake-up filter register block.

### 24.3.11.2 Remote Wake-Up Frame Detection

When the MAC is in sleep mode and the remote wake-up frame enable bit, WUPFREN, is set in the **Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT)** register, the normal operation is resumed after a remote wake-up frame is received. The application writes all eight wake-up filter registers, by performing eight sequential writes to the **Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF)** register. The Power Management (PMT) block supports four programmable filters that allow support of different receive frame patterns. If the incoming frame passes the address filtering of Filter Command, and if Filter CRC-16 matches the CRC of the incoming pattern, then the MAC identifies the frame as wake-up frame. The Filter Offset determines the offset from which the frame is to be examined. The Filter Byte Mask determines which bytes of the frame must be examined. The 31st bit of Byte Mask must be set to zero. The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The remote wake-up frame is checked only for length error, FCS error, dribble bit error, MII error, and collision. In addition, the remote wake-up frame is checked to ensure that it is not a runt frame. Even if the remote wakeup frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. The remote wake-up frame detection is updated in the **Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT)** register for every remote wake-up frame received. If the PMT interrupt is enabled in the **Ethernet MAC Interrupt Mask (EMACIM)** register, a PMT interrupt is asserted and the **EMACPMTCTLSTAT** register can be read to determine reception of a remote wake-up frame.

### 24.3.11.3 Magic Packet Detection

The magic packet frame is based on a method that uses Advanced Micro Device's magic packet technology to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network. The MAC checks only those magic packets that are addressed to the MAC or a broadcast address to determine whether these packets meet the wake-up requirements. The magic packets that pass the address filtering (unicast or broadcast) are checked to determine whether they meet the remote wake-up frame data format of 6 bytes of all ones followed by a MAC Address appearing 16 times. The application enables the magic packet wake-up by setting the magic packet enable bit, MGKPKTEN, of the **Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT)** register. The power management block constantly monitors each frame addressed to the node for a specific magic packet pattern. Each frame received is checked for a 0xFFFF.FFFF.FFFF pattern following the destination and source address field. The power management block then checks the frame for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again

scans the 0xFFFF.FFFF.FFFF pattern in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (0xFFFF.FFFF.FFFF). The device can also accept a multicast frame, as long as the 16 duplications of the MAC address are detected. If the MAC address of a node is 0x0011.2233.4455, then the MAC scans for the following data sequence:

Destination Address Source Address …….. FF FF FF FF FF FF

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 …CRC

The magic packet detection is updated in the **EMACPMTCTLSTAT** register for the received magic packet. If the PMT interrupt is enabled in the **Ethernet MAC Interrupt Mask (EMACIM)** register, a PMT interrupt is asserted and the **EMACPMTCTLSTAT** register can be read to determine whether a magic packet frame has been received.

### 24.3.11.4 Power Management Interrupts

The PMT interrupt signal can be asserted when a valid remote wake-up frame or magic packet is received. The PMT interrupt signal restores the application clock and TX clock to the MAC. When the **Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT)** register is read, the PMT interrupt is cleared in the **EMACRIS** register at least after four clock cycles of RX clock. When software resets the PWRDWN bit in the **Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT)** register, the MAC comes out of the power-down mode, but this event does not generate at PMT interrupt.

### 24.3.11.5 Power-Down/Wake-Up Sequence

The recommended power-down and wake-up sequence is as follows:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. These transmissions can be detected when TI is set in the **Ethernet MAC DMA Interrupt Status (EMACDMARIS)** register.
2. Disable the MAC transmit and receive state machine by clearing the TE and RE bits in the **Ethernet MAC Configuration (EMACCFG)** register.
3. Wait until the RX DMA empties all the frames from the Rx FIFO to system memory by polling the RXF field of the **Ethernet MAC Status (EMACSTATUS)** register.
4. Enable a power management mode by setting the magic packet, global unicast or remote wake-up enable bit in the **EMACPMTCTLSTAT** register.
5. Enable the MAC receive state machine in the **EMACCFG** register and enter the Power-Down mode by setting the PWRDWN bit in the **EMACPMTCTLSTAT** register.
6. On receiving a valid remote wake-up frame, the PMT interrupt is set in the **EMACRIS** register and the Ethernet MAC exits the Power-Down mode.
7. Read the **EMACPMTCTLSTAT** register to clear the PMT interrupt, then enable the other modules in the system and resume normal operation.

### 24.3.12 Serial Management Interface

The Ethernet MAC has the ability to read or write to the PHY registers through the EN0MDIO and EN0MDC signals of the Serial Management Interface defined by the IEEE 802.3 standard. The internal EN0MDIO and EN0MDC signals connect to the integrated PHY as well as to the external EN0MDIO and EN0MDC pins. The EN0MDC signal is a 2.5 MHz clock that is sourced from System Clock (SYSCLK) and then divided down to the required frequency by programming the CR field in the **Ethernet MAC MII Address (EMACMIIADDR)** register. To access the integrated PHY, the PLA field in the **EMACMIIADDR** register must be 0x0. The available addresses for external PHYs are 0x01 to 1F.

### 24.3.13 Reduced Media Independent Interface (RMII)

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Ethernet MACs. According to the IEEE 802.3u standard, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces such as switches, the number of pins adds significant cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port — a 62.5% decrease in pin count.

The RMII module has the following features:

- Supports both 10-Mbps and 100-Mbps operating rates
- Provides independent, two-bit wide transmit and receive paths

Each nibble is transmitted on the RMII two bits at a time. For a nibble {D3, D2, D1, D0}, the data is transferred as {D1, D0} followed by {D3, D2}.

### 24.3.14 Interrupt Configuration

Interrupts can be generated from the MAC as a result of various events in the MAC and sub-modules. MAC interrupts are enabled or disabled in the **Ethernet MAC Interrupt Mask (EMACIM)** register, MAC offset 0x03C. Each interrupt event can be masked by setting the corresponding mask bit in the **EMACIM** register.

The interrupt register bits in the **Ethernet MAC Raw Interrupt Status (EMACRIS)** register only indicate the sub-module from which the event is reported. The application must read the corresponding status registers to clear the interrupt.

## 24.4 Ethernet PHY

The integrated PHY supports 10Base-T and 100Base-TX signaling. It integrates all the physical-layer functions needed to transmit and receive data on standard twisted-pair cables. The PHY directly interfaces to the integrated Media Access Controller (MAC).

The Ethernet PHY uses mixed-signal processing to perform equalization, data recovery, and error correction to achieve robust operation over CAT5 twisted-pair wiring. It not only meets the requirements of IEEE 802.3, but maintains high margins in terms of alien cross-talk. The following highlights the features of the PHY module:

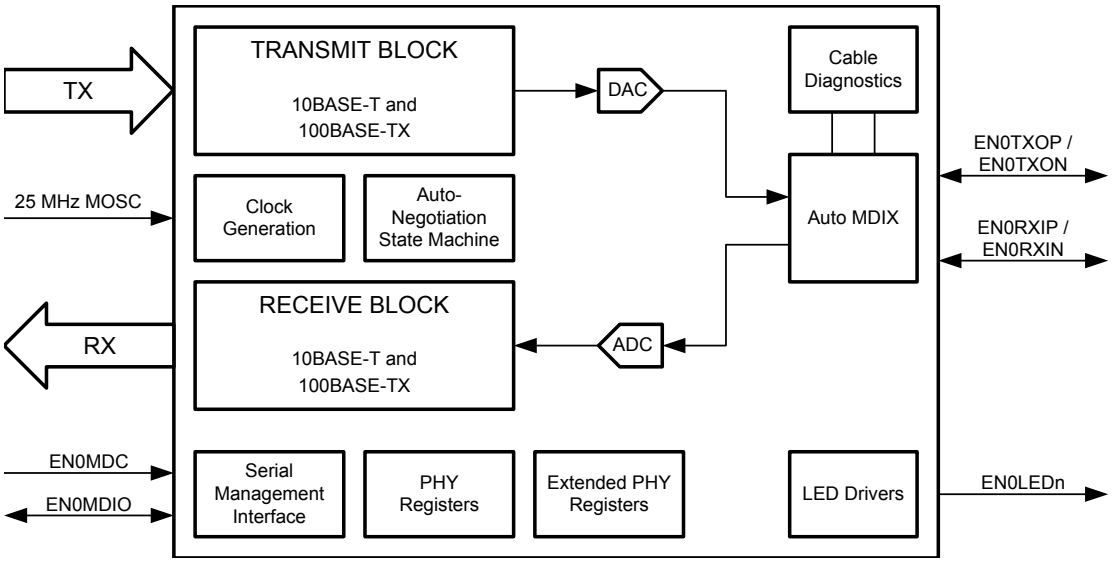
- Cable Diagnostics
- Programmable Fast Link Down Modes
- Auto-MDIX for 10/100Mbs

- Energy Detection Mode
- Serial Management Interface
- IEEE 802.3u Auto-Negotiation and Parallel Detection
- IEEE 802.3u ENDEC, 10Base-T Transceivers and Filters
- IEEE 802.3u PCS, 100Base-TX Transceivers
- Integrated ANSI X3.263 Compliant TP-PMD Physical Sublayer with Adaptive Equalization and Baseline Wander Compensation
- Three programmable LEDs that support detection of Link OK, 10/100Mbs activity, TX/RX transfers, collisions and full duplex mode

### 24.4.1 Integrated PHY Block Diagram

The following figure shows the internal PHY integration. Note that the reference clock input comes from an external 25 MHz ± 50 ppm crystal or oscillator connected to the MOSC signals.

**Figure 24-14. Integrated PHY Diagram**




### 24.4.2 Functional Description

The following sections describe the functional characteristics of the integrated PHY.

#### 24.4.2.1 Auto-Negotiation

The Ethernet PHY can auto-negotiate to operate in 10Base-T or 100Base-TX. When the Ethernet PHY is enabled or on the deassertion of software reset, the **Ethernet MAC Peripheral Configuration Register (EMACPC)** register, offset 0xFC4, is configured such that auto-negotiation is enabled and the auto negotiation mode (ANMODE) bit field is programmed to 10Base-T,Half/Full-Duplex 100Base-TX, Half/Full-Duplex. With auto-negotiation enabled, the PHY negotiates with the link partner to determine the speed and duplex mode with which to operate. If the link partner is unable to auto-negotiate, the PHY goes into parallel-detect mode to determine the speed of the link partner. Under parallel-detect mode, the duplex mode is fixed at half-duplex. The PHY supports four different

Ethernet protocols (10Mbs Half-Duplex, 10Mbs Full-Duplex, 100Mbs Half-Duplex, and 100Mbs Full-Duplex). Auto-Negotiation selects the highest performance protocol based on the advertised ability of the Link Partner. If a different auto-negotiation configuration is required other than the reset initialization values, the application can customize the configuration of the ANEN bit and ANMODE bit field as described in “Custom Configuration” on page 1653. The values of ANEN and ANMODE determine whether the PHY is forced into a specific mode, or if Auto-negotiation advertises a specific ability (or abilities) as listed in Table 24-21 on page 1644 and Table 24-22 on page 1644:

**Table 24-21. Forced Mode Configurations**

| ANEN value | ANMODE | Forced Mode |
|---|---|---|
| 0 | 0x0 | 10Base-T, Half-Duplex |
| 0 | 0x1 | 10Base-T, Full-Duplex |
| 0 | 0x2 | 100Base-TX, Half-Duplex |
| 0 | 0x3 | 100Base-TX, Full-Duplex |

**Table 24-22. Advertised Mode Configurations**

| ANEN value | ANMODE | Advertised Mode |
|---|---|---|
| 1 | 0x0 | 10Base-T, Half/Full-Duplex |
| 1 | 0x1 | 100Base-TX, Half/Full-Duplex |
| 1 | 0x2 | 10Base-T, Half Duplex 100Base-TX, Half Duplex |
| 1 | 0x3 | 10Base-T, Half/Full-Duplex 100Base-TX, Half/Full-Duplex |

#### 24.4.2.2 Auto-MDIX

The PHY automatically determines whether or not it needs to cross over between pairs so that an external crossover cable is not required. If the PHY communicates with a device that implements MDI/MDIX crossover, a random algorithm as described in IEEE 802.3 determines which device performs the crossover. Auto-MDIX is enabled by default at reset. If a different auto-MDIX configuration is required other than the reset initialization, the application can customize the configuration as described in “Custom Configuration” on page 1653. Neither Auto- Negotiation nor Auto-MDIX is required to be enabled in forcing crossover of the MDI pairs. Auto-MDIX can be used in the forced 100Base-TX mode. Because in modern networks all the nodes are 100Base-TX, having the Auto-MDIX working in the forced 100Base-TX mode resolves the link faster without the need for the long Auto-Negotiation period.

#### 24.4.2.3 Isolate Mode

The PHY can be put into Isolate mode by writing ISOLATE bit of the **Ethernet PHY Basic Mode Control - MR0 (EPHYBMCR)** register, address 0x000. When in the Isolate mode, the PHY does not respond to packet data present at the internal interface to the Ethernet MAC. When in isolate mode, the PHY continues to respond to all management transactions and the PMD output pair does not transmit packet data, but continues to source 100Base-TX scrambled idles or 10Base-T normal link pulses. The PHY can auto-negotiate or parallel detect on the receive signal at the PMD input pair. A valid link can be established for the receiver even when the PHY is in Isolate mode.

#### 24.4.2.4 LED Interface

The PHY supports three configurable Light Emitting Diode (LED) pins to indicate link status of a port. The **Ethernet PHY LED Configuration - MR37 (EPHYLEDCFG)** register, address 0x025 can be used to assign different functions to each LED. Each LED can be configured to be active during one of these events:

- Link OK (0x0)
- RX/TX Activity (0x1)
- TX Activity (0x2)
- RX Activity (0x3)
- Collision (0x4)
- 100-BASE TX speed (0x5)
- 10-Base TX speed (0x6)
- Full Duplex (0x7)
- Link OK/Blink on TX/RX Activity (0x8)

At reset, LED0 is initialized to display Link OK and LED1 and LED 2 are initialized to the RX/TX activity encoding. The blink rate of the LEDs can be set by programming the BLINKRATE bit field of the **Ethernet PHY LED Control - MR24 (EPHYLEDCR)** register, address 0x018

#### 24.4.2.5 PHY Address

The integrated PHY's address is 0x00. To access the integrated PHY registers through the internal MAC's MDIO interface, the PLA bit field of the **Ethernet MAC MII Address (EMACMIIADDR**) register must be 0x00 and the MII field should be programmed to the desired address value.

#### 24.4.2.6 Serial Management Interface

The Ethernet MAC module has the capability of programming the integrated PHY registers and extended registers through an internal Serial Management Interface (SMI). The SMI is compatible with IEEE802.3-2002 clause 22.

The SMI includes an MDC management clock input and management MDIO data pin to the Ethernet PHY. The internal MDC clock is sourced by the system clock and then divided down to the required 2.5 MHz maximum clock frequency by setting the CR bit in the **Ethernet MAC MII Address (EMACMIIADDR)** register. The MDC is not expected to be continuous, and can be turned off by the external management entity when the bus is idle. The MDIO is sourced by the integrated MAC and by the PHY. The data on the MDIO pin is latched on the rising edge of the MDC clock. The PHY's physical address is 0x00.

#### 24.4.2.7 Extended Address Space Access

The PHY SMI function supports read/write accesses to the extended register set using **Ethernet PHY Register Control - MR13 (EPHYREGCTL)** register and the **Ethernet PHY Address or Data - MR14 (EPHYADDAR)** registers. Accessing the standard register set, MDIO registers 0 to 31, can be performed using the normal direct MDIO access or the indirect method, except for the **EPHYREGCTL (0x00D)** and the **EPHYADDAR (0x00E)** which can be accessed only using the normal MDIO transaction. The SMI function ignores indirect accesses to these registers.

The **EPHYREGCTL** register is the MDIO Manageable MMD access control. In general, register **EPHYREGCTL[4:0]** is the device address DEVAD that directs any accesses of **EPHYADDAR** register to the appropriate MMD. Specifically, the PHY uses the vendor specific DEVAD[4:0] = 0xF for accesses. All accesses through registers **EPHYREGCTL** and **EPHYADDAR** registers should use

this DEVAD. Transactions with other DEVAD are ignored. The **EPHYREGCTL[15:14]** register holds the access function:

- **EPHYREGCTL[15:14] =** 0x0: A write to **EPHYADDAR** modifies the extended register set address register. This address register must be initialized in order to access any of the registers within the extended register set.
- **EPHYREGCTL[15:14]** = 0x1: A read/write to **EPHYADDAR** operates on the register within the extended register set selected (pointed to) by the value in the address register. The address register contents (pointer) remain unchanged.
- **EPHYREGCTL[15:14]** = 0x2: A read/write to **EPHYADDAR** operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for both reads and writes, the value in the address register is incremented.
- **EPHYREGCTL[15:14] =** 0x3: A read/write to **EPHYADDAR** operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for write accesses only, the value in the address register is incremented. For read accesses, the value of the address register remains unchanged.

The following sections describe how to perform operations on the extended register set using the **EPHYREGCTL** and **EPHYADDAR** registers.

***Write to PHY Registers***

The following describes the steps to write to a PHY register.

1. Check the MIIB bit in the **EMACMIIADDR** register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
2. Write the data to be written to the PHY register in the **EMACMIIDATA** register.
3. Initiate write by programming the **EMACMIIADDR** register fields as follows:
   - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
   - MII: Address of the PHY register to be written.
   - CR: Clock Reference for the MDIO interface.
   - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
   - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.

***Write to Extended PHY Registers***

The following describes the steps to write to an extended PHY register.

1. Check the MIIB bit in the **EMACMIIADDR** register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
2. The **EMACMIIDATA** register should be written with the value to be passed into the **EPHYREGCTL** register. The **EPHYREGCTL** register is used for extended PHY register accesses. The DEVAD field of the **EPHYREGCTL** register identifies the device address, which is 0x1F, for

the integrated PHY. The FUNC field of the **EPHYREGCTL** register should be set to 0x0 to indicate a write to an extended register address.

3. Initiate write by programming the **EMACMIIADDR** register fields as follows:
   - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
   - MII: Address of the PHY register to be written. In this case, it should be the address of the **EPHYREGCTL** register, 0xD.
   - CR: Clock Reference for the MDIO interface.
   - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
   - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.
4. Check the MIIB bit in the **EMACMIIADDR** register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
5. The **EMACMIIDATA** register should be written with the address of the extended register to be accessed. This value is written to the **EPHYADDAR** register.
6. Initiate write by writing the **EMACMIIADDR** register fields:
   - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
   - MII: Address of the PHY register to be written. In this case, it should be the address of the **EPHYADDAR** register, 0xE.
   - CR: Clock Reference for the MDIO interface.
   - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
   - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.
7. Check the MIIB bit in the **EMACMIIADDR** register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
8. The **EMACMIIDATA** register should be written with the value to be passed into the **EPHYREGCTL** register. The DEVAD field of the **EPHYREGCTL** register identifies the device address, which is 0x1F, for the integrated PHY. The FUNC field of the **EPHYREGCTL** register should be set to 0x1 to indicate a write to an extended register address with no increment.
9. Initiate write by writing the **EMACMIIADDR** register fields:
   - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
   - MII: Address of the PHY register to be written. In this case, it should be the address of the **EPHYADDAR** register, 0xD.

- CR: Clock Reference for the MDIO interface.
- MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
- MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.

10. Check the MIIB bit in the **EMACMIIADDR** register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to write to the PHY registers.
11. The **EMACMIIDATA** register should be programmed with the data to be written to the **EPHYADDAR** register which is transferred to the previously selected extended PHY register.
12. Initiate write by writing the **EMACMIIADDR** register fields:
    - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
    - MII: Address of the PHY register to be written. In this case, it should be the address of the **EPHYADDAR** register, 0xD.
    - CR: Clock Reference for the MDIO interface.
    - MIIW: Write Initiation. This bit is set to 1 to indicate that a write operation is to be executed.
    - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a write operation. The EMAC clears this bit when the write has been transmitted.

***Read from PHY Registers***

The following describes the steps to read from an extended PHY register.

1. Check the MIIB bit in the **EMACMIIADDR** register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to read to the PHY registers.
2. Initiate the read by writing the **EMACMIIADDR** register fields:
    - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
    - MII: Register address of PHY register to be written.
    - CR: Clock Reference for the MDIO interface.
    - MIIW: Write/Read Initiation. This bit is programmed to a 0 to indicate that a read operation is to be executed.
    - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a read operation. The EMAC clears this bit when the write has been transmitted.
3. Wait for the MII interface to complete the read by polling the MIIB bit.
4. When the MIIB is clear, read the contents of the **EMACMIIDATA** register.

***Read from Extended PHY Registers***

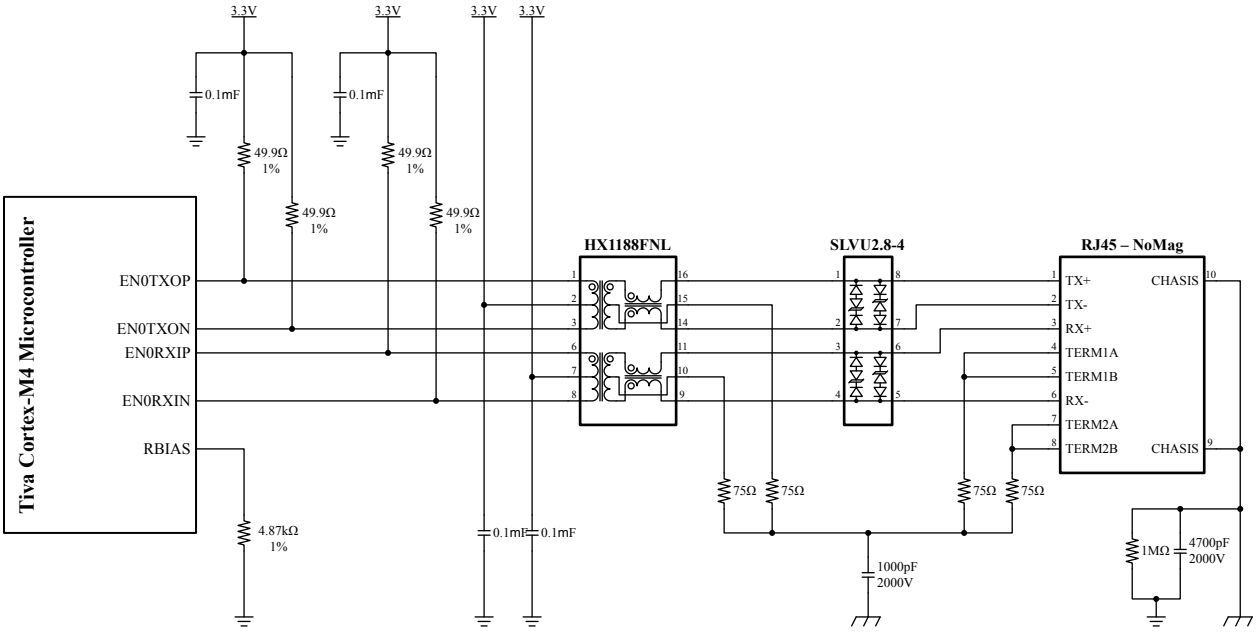
The following describes the steps to read from an extended PHY register.

1. Check the MIIB bit in the **EMACMIIADDR** register to identify if the MII interface is busy. When the MIIB bit is 0, the MII interface is available to read to the PHY registers.
2. Initiate the read by writing the **EMACMIIADDR** register fields:
   - PLA: Physical Layer Address of the PHY. The integrated PHY's address is 0x0. The values 0x1 to 0x1F are available for external PHYs.
   - MII: Register address of PHY register to be written.
   - CR: Clock Reference for the MDIO interface.
   - MIIW: Write/Read Initiation. This bit is programmed to a 0 to indicate that a read operation is to be executed.
   - MIIB: MII Busy. This bit is set to 1 to indicate that the MII is now busy with a read operation. The EMAC clears this bit when the write has been transmitted.
3. Wait for the write to complete by polling the MIIB bit.
4. When the MIIB is clear, read the contents of the **EMACMIIDATA** register.

### 24.4.3 Interface Configuration

Figure 24-15 on page 1650 shows the proper method for interfacing the Ethernet Controller to a 10/100BASE-T Ethernet jack.

**Figure 24-15. Interface to Ethernet Jack**




**NOTE:** The 0.1mF capacitors on pins 2 and 7 of the HX1188FNL should be located near the physical pins of the transformer.

The Ethernet PHY is designed to work with transformers that meet the IEEE 802.3 standard. To utilize the Auto-MDIX (AMDIX) capability of the Ethernet PHY, a symmetrical transformer is recommended. The Pulse HX1188 transformer has been tested and is known to successfully interface to the Ethernet PHY.

**Note:** If the PHY is not to be used, then the EN0RXIN/EN0TX0N EN0RXIP/EN0TX0P pair should be left unconnected and the RBIAS pin should be unconnected as well.

**Note:** When entering hibernation in VDD3ON mode, the supply rails to the Ethernet resistors R1, R2, R3, R4 found in Figure 24-15 on page 1650 must be switched off.

## 24.5 Initialization and Configuration

The MAC module and registers are enabled and powered at reset. When reset has completed, the application should enable the clock to the Ethernet MAC by setting the R0 bit in the **Ethernet Controller Run Mode Clock Gating Control (RCGCEMAC)** register at System Control Module offset 0x69C. When the **PREMAC** register, at System Control offset 0xA9C reads as 0x0000.0001, the EMAC registers are ready to be accessed.

The EMAC interface defaults to MII mode. If RMII mode is required, follow these steps:

1. Enable the external clock source input to the RMII interface signal EN0RREF_CLK by setting both the ECEXT and CLKEN bit in the in the **Ethernet Clock Configuration (EMACCC)** register at offset 0xFC8. The external clock source must be 50 MHz with a frequency tolerance of 50 PPM.

2. Select the RMII interface by programming the PINTFS bit field to 0x4 in the **Ethernet Peripheral Configuration (EMACPC)** register at offset 0xFC4.

3. Reset the Ethernet MAC to latch the new RMII configuration by setting the SWR bit in the **EMACDMABUSMOD** register. This bit resets the Ethernet MAC registers in addition to configuring the RMII interface. Software must poll the SWR bit to determine when the new configuration has been registered.

   **Note:** After this configuration is active, if the Ethernet MAC is reset by setting the R0 bit in the **Ethernet MAC Software Reset (SREMAC)** register in the System Control Module, then the interface is set back to its default MII configuration. In this case, the steps listed above must be repeated to return to an RMII interface.

The Initialization for the DMA for the Ethernet MAC is as follows:

1. Write to the Ethernet **MAC DMA Bus Mode (EMACDMABUSMOD)** register to set Host bus parameters.
2. Write to the Ethernet **MAC DMA Interrupt Mask Register (EMACDMAIM)** register to mask unnecessary interrupt causes.
3. Create the transmit and receive descriptor lists and then write to the **Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR**) register and the **Ethernet MAC Transmit Descriptor List Address (EMACTXDLADDR)** register providing the DMA with the starting address of each list.
4. Write to the **Ethernet MAC Frame Filter (EMACFRAMEFLTR)** register, the **Ethernet MAC Hash Table High (EMACHASHTBLH**) and the **Ethernet MAC Hash Table Low (EMACHASHTBLL)** for desired filtering options.
5. Write to the **Ethernet MAC Configuration Register (EMACCFG)** to configure the operating mode and enable the transmit operation.
6. Program Bit 15 (PS) and Bit 11 (DM) of the **EMACCFG** register based on the line status received or read from the PHY status register after auto-negotiation.
7. Write to the **Ethernet MAC DMA Operation Mode (EMACDMAOPMODE)** register to set Bits 13 and 1 to start transmission and reception.
8. Write to the **EMACCFG** register to enable the receive operation.

   The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

### 24.5.1 Ethernet PHY Initialization

After reset, when the EMAC is powered and enabled, the **EMACPC** register default reset value may be sampled and used to configure the PHY. The results of this configuration can also be read in the following EPHY registers:

- **EPHYBMCR** register (MR0)
- **EPHYCFG1** register (MR9)
- **EPHYCFG2** register (MR10)
- **EPHYCFG3** register (MR11)

- **EPHYCTL** register (MR25)

The mappings of the **EMACPC** register bits to the PHY register and bits are as follows:

**Table 24-23. EMACPC to PHY Register Mapping**

| EMACPC Register Bit | Corresponding PHY Register | Corresponding PHY Bit (Bit No.) |
|---|---|---|
| PHYEXT | N/A | N/A |
| DIGRESTART | N/A | N/A |
| NIBDETDIS | EPHYCFG2 | ODDNDETDIS (1) |
| RXERIDLE | EPHYCFG2 | RXERRIDLE (2) |
| ISOMILL | EPHYCFG2 | ISOMILL (3) |
| LRR | EPHYCFG1 | LLR (7) |
| TDRRUN | EPHYCFG1 | TDRAR (8) |
| FASTLDMODE | EPHYCFG3 | FLDWNM (4:0) |
| POLSWAP | EPHYCFG3 | POLSWAP (7) |
| MDISWAP | EPHYCFG3 | MDIMDIXS (6) |
| RBSTMDIX | EPHYCFG1 | RAMDIX (5) |
| FASTMDIX | EPHYCFG1 | FAMDIX (6) |
| MDIXEN | EPHYCTL | AUTOMDI (15) |
| FASTRXDV | EPHYCFG1 | FRXDVDET (1) |
| FASTLUPD | EPHYCFG2 | FLUPPD (6) |
| EXTFD | EPHYCFG2 | EXTFD (5) |
| FASTANEN | EPHYCFG1 | FASTANEN (4) |
| FASTANSEL | EPHYCFG1 | FANSEL (3:2) |
| ANEN | EPHYBMCR | ANEN |
| ANMODE | N/A | N/A |
| PHYHOLD | N/A | N/A |

The MAC module and registers are enabled and powered at reset. When reset has completed and the clock to the Ethernet MAC is enabled by setting the R0 bit in the **Ethernet Controller Run Mode Clock Gating Control (RCGCEMAC)** register at System Control Module offset 0x69C, the application has the option to enable the PHY with its default interface configuration (as defined by the **Ethernet MAC Peripheral Configuration Register (EMACPC)** register) or with a custom configuration.

#### 24.5.1.1 Default Configuration

To enable the Ethernet PHY with its default configuration, the steps are as follows:

1. To hold the Ethernet PHY from transmitting energy on the line during configuration, set the PHYHOLD bit to 1 in the **EMACPC** register.

2. Enable the clock to the PHY module by writing 0x0000.0001 to the **Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY)** register at offset 0x630. When the R0 bit reads as 1 in the **PREPHY** register at System Control offset 0xA30, continue initialization.

3. Enable power to the Ethernet PHY by setting the P0 bit in the **PCEPHY** register at System Control offset 0x930. When the R0 bit reads as 1 in the **PREPHY** register at System Control offset 0xA30, the PHY registers are ready for programming.

#### 24.5.1.2 Custom Configuration

If a custom configuration of the Ethernet PHY is required, the application can program the configuration registers after reset. The steps for custom configuration are as follows:

1. To hold the PHY from transmitting energy on the line during configuration, set the PHYHOLD bit to 1 in the **EMACPC** register.
2. Enable the clock to the PHY module by writing 0x0000.0001 to the **Ethernet PHY Run Mode Clock Gating Control (RCGCEPHY)** register at offset 0x630. When the R0 bit reads as 1 in the **PREPHY** register at System Control offset 0xA30, continue initialization.
3. Enable power to the Ethernet PHY by setting the P0 bit in the **PCEPHY** register at System Control offset 0x930. When the R0 bit reads as 1 in the **PREPHY** register at System Control offset 0xA30, the PHY registers are ready for programming.
4. Once the **Ethernet PHY Peripheral Ready (PREPHY)** register reads 0x0000.0001, software can write the **EMACPC** register with the required value.
5. After software configuration is complete, the application must set the DONE bit in the **Ethernet PHY Configuration 1 (EPHYCFG1)** register at offset 0x009.

**Note:** If a software reset is asserted to the PHY afterwards through the **SREPHY** register, the custom configuration is lost and the steps described above must be repeated.

## 24.6 Register Map

Table 24-24 on page 1653 lists the Ethernet Controller MAC and PHY registers. For the MAC registers, the offset listed is a hexadecimal increment to the MAC base address of 0x400E.C000. PHY registers are accessed through the **EMACMIIADDR** register thus the base address is n/a (not applicable) and noted as such above the register descriptions.

The *IEEE 802.3* standard specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers. Table 24-24 on page 1653 also lists these MII Management registers for interfacing to the internal PHY. All addresses given are absolute and are written directly to the MII field of the **Ethernet MAC MII Address (EMACMIIADDR)** register, offset 0x010. The PLA value of the **EMACMIIADDR** register for the internal PHY is 0x00.

The **Ethernet MAC MII Address (EMACMIIADDR)** register, offset 0x010, is used to access MII Management registers on the external PHY device. The PLA field in the **EMACMIIADDR** register supports PHY addresses 1 to 31.

**Table 24-24. Ethernet Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **Ethernet MAC (Ethernet Offset)** | | | | | |
| 0x000 | EMACCFG | RW | 0x0000.8000 | Ethernet MAC Configuration | 1658 |
| 0x004 | EMACFRAMEFLTR | RW | 0x0000.0000 | Ethernet MAC Frame Filter | 1665 |
| 0x008 | EMACHASHTBLH | RW | 0x0000.0000 | Ethernet MAC Hash Table High | 1669 |
| 0x00C | EMACHASHTBLL | RW | 0x0000.0000 | Ethernet MAC Hash Table Low | 1670 |

**Table 24-24. Ethernet Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x010 | EMACMIIADDR | RW | 0x0000.0000 | Ethernet MAC MII Address | 1671 |
| 0x014 | EMACMIIDATA | RW | 0x0000.0000 | Ethernet MAC MII Data Register | 1673 |
| 0x018 | EMACFLOWCTL | RW | 0x0000.0000 | Ethernet MAC Flow Control | 1674 |
| 0x01C | EMACVLANTG | RW | 0x0000.0000 | Ethernet MAC VLAN Tag | 1676 |
| 0x024 | EMACSTATUS | RO | 0x0000.0000 | Ethernet MAC Status | 1678 |
| 0x028 | EMACRWUFF | RW | 0x0000.0000 | Ethernet MAC Remote Wake-Up Frame Filter | 1681 |
| 0x02C | EMACPMTCTLSTAT | RW | 0x0000.0000 | Ethernet MAC PMT Control and Status Register | 1682 |
| 0x038 | EMACRIS | RO | 0x0000.0000 | Ethernet MAC Raw Interrupt Status | 1684 |
| 0x03C | EMACIM | RW | 0x0000.0000 | Ethernet MAC Interrupt Mask | 1686 |
| 0x040 | EMACADDR0H | RW | 0x8000.FFFF | Ethernet MAC Address 0 High | 1687 |
| 0x044 | EMACADDR0L | RW | 0xFFFF.FFFF | Ethernet MAC Address 0 Low Register | 1688 |
| 0x048 | EMACADDR1H | RW | 0x0000.FFFF | Ethernet MAC Address 1 High | 1689 |
| 0x04C | EMACADDR1L | RW | 0xFFFF.FFFF | Ethernet MAC Address 1 Low | 1691 |
| 0x050 | EMACADDR2H | RW | 0x0000.FFFF | Ethernet MAC Address 2 High | 1692 |
| 0x054 | EMACADDR2L | RW | 0xFFFF.FFFF | Ethernet MAC Address 2 Low | 1694 |
| 0x058 | EMACADDR3H | RW | 0x0000.FFFF | Ethernet MAC Address 3 High | 1695 |
| 0x05C | EMACADDR3L | RW | 0xFFFF.FFFF | Ethernet MAC Address 3 Low | 1697 |
| 0x0DC | EMACWDOGTO | RW | 0x0000.0000 | Ethernet MAC Watchdog Timeout | 1698 |
| 0x100 | EMACMMCCTRL | RW | 0x0000.0000 | Ethernet MAC MMC Control | 1699 |
| 0x104 | EMACMMCRXRIS | RO | 0x0000.0000 | Ethernet MAC MMC Receive Raw Interrupt Status | 1702 |
| 0x108 | EMACMMCTXRIS | R | 0x0000.0000 | Ethernet MAC MMC Transmit Raw Interrupt Status | 1704 |
| 0x10C | EMACMMCRXIM | RW | 0x0000.0000 | Ethernet MAC MMC Receive Interrupt Mask | 1706 |
| 0x110 | EMACMMCTXIM | RW | 0x0000.0000 | Ethernet MAC MMC Transmit Interrupt Mask | 1708 |
| 0x118 | EMACTXCNTGB | RO | 0x0000.0000 | Ethernet MAC Transmit Frame Count for Good and Bad Frames | 1710 |
| 0x14C | EMACTXCNTSCOL | RO | 0x0000.0000 | Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision | 1711 |
| 0x150 | EMACTXCNTMCOL | RO | 0x0000.0000 | Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions | 1712 |
| 0x164 | EMACTXOCTCNTG | RO | 0x0000.0000 | Ethernet MAC Transmit Octet Count Good | 1713 |
| 0x180 | EMACRXCNTGB | RO | 0x0000.0000 | Ethernet MAC Receive Frame Count for Good and Bad Frames | 1714 |
| 0x194 | EMACRXCNTCRCERR | RO | 0x0000.0000 | Ethernet MAC Receive Frame Count for CRC Error Frames | 1715 |

**Table 24-24. Ethernet Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x198 | EMACRXCNTALGNERR | RO | 0x0000.0000 | Ethernet MAC Receive Frame Count for Alignment Error Frames | 1716 |
| 0x1C4 | EMACRXCNTGUNI | RO | 0x0000.0000 | Ethernet MAC Receive Frame Count for Good Unicast Frames | 1717 |
| 0x584 | EMACVLNINCREP | RW | 0x0000.0000 | Ethernet MAC VLAN Tag Inclusion or Replacement | 1718 |
| 0x588 | EMACVLANHASH | RW | 0x0000.0000 | Ethernet MAC VLAN Hash Table | 1720 |
| 0x700 | EMACTIMSTCTRL | RW | 0x0000.2000 | Ethernet MAC Timestamp Control | 1721 |
| 0x704 | EMACSUBSECINC | RW | 0x0000.0000 | Ethernet MAC Sub-Second Increment | 1725 |
| 0x708 | EMACTIMSEC | RO | 0x0000.0000 | Ethernet MAC System Time - Seconds | 1726 |
| 0x70C | EMACTIMNANO | RO | 0x0000.0000 | Ethernet MAC System Time - Nanoseconds | 1727 |
| 0x710 | EMACTIMSECU | RW | 0x0000.0000 | Ethernet MAC System Time - Seconds Update | 1728 |
| 0x714 | EMACTIMNANOU | RW | 0x0000.0000 | Ethernet MAC System Time - Nanoseconds Update | 1729 |
| 0x718 | EMACTIMADD | RW | 0x0000.0000 | Ethernet MAC Timestamp Addend | 1730 |
| 0x71C | EMACTARGSEC | RW | 0x0000.0000 | Ethernet MAC Target Time Seconds | 1731 |
| 0x720 | EMACTARGNANO | RW | 0x0000.0000 | Ethernet MAC Target Time Nanoseconds | 1732 |
| 0x724 | EMACHWORDSEC | RW | 0x0000.0000 | Ethernet MAC System Time-Higher Word Seconds | 1733 |
| 0x728 | EMACTIMSTAT | RO | 0x0000.0000 | Ethernet MAC Timestamp Status | 1734 |
| 0x72C | EMACPPSCTRL | RW | 0x0000.0000 | Ethernet MAC PPS Control | 1735 |
| 0x760 | EMACPPS0INTVL | RW | 0x0000.0000 | Ethernet MAC PPS0 Interval | 1738 |
| 0x764 | EMACPPS0WIDTH | RW | 0x0000.0000 | Ethernet MAC PPS0 Width | 1739 |
| 0xC00 | EMACDMABUSMOD | RW | 0x0002.0101 | Ethernet MAC DMA Bus Mode | 1740 |
| 0xC04 | EMACTXPOLLD | WO | 0x0000.0000 | Ethernet MAC Transmit Poll Demand | 1744 |
| 0xC08 | EMACRXPOLLD | WO | 0x0000.0000 | Ethernet MAC Receive Poll Demand | 1745 |
| 0xC0C | EMACRXDLADDR | RW | 0x0000.0000 | Ethernet MAC Receive Descriptor List Address | 1746 |
| 0xC10 | EMACTXDLADDR | RW | 0x0000.0000 | Ethernet MAC Transmit Descriptor List Address | 1747 |
| 0xC14 | EMACDMARIS | RW | 0x0000.0000 | Ethernet MAC DMA Interrupt Status | 1748 |
| 0xC18 | EMACDMAOPMODE | RW | 0x0000.0000 | Ethernet MAC DMA Operation Mode | 1754 |
| 0xC1C | EMACDMAIM | RW | 0x0000.0000 | Ethernet MAC DMA Interrupt Mask Register | 1759 |
| 0xC20 | EMACMFBOC | RO | 0x0000.0000 | Ethernet MAC Missed Frame and Buffer Overflow Counter | 1762 |
| 0xC24 | EMACRXINTWDT | RW | 0x0000.0000 | Ethernet MAC Receive Interrupt Watchdog Timer | 1763 |
| 0xC48 | EMACHOSTXDESC | R | 0x0000.0000 | Ethernet MAC Current Host Transmit Descriptor | 1764 |
| 0xC4C | EMACHOSRXDESC | RO | 0x0000.0000 | Ethernet MAC Current Host Receive Descriptor | 1765 |

**Table 24-24. Ethernet Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xC50 | EMACHOSTXBA | R | 0x0000.0000 | Ethernet MAC Current Host Transmit Buffer Address | 1766 |
| 0xC54 | EMACHOSRXBA | R | 0x0000.0000 | Ethernet MAC Current Host Receive Buffer Address | 1767 |
| 0xFC0 | EMACPP | RO | 0x0000.0103 | Ethernet MAC Peripheral Property Register | 1768 |
| 0xFC4 | EMACPC | RW | 0x0080.040E | Ethernet MAC Peripheral Configuration Register | 1769 |
| 0xFC8 | EMACCC | RO | 0x0000.0000 | Ethernet MAC Clock Configuration Register | 1773 |
| 0xFD0 | EPHYRIS | RO | 0x0000.0000 | Ethernet PHY Raw Interrupt Status | 1774 |
| 0xFD4 | EPHYIM | RW | 0x0000.0000 | Ethernet PHY Interrupt Mask | 1775 |
| 0xFD8 | EPHYMISC | RW1C | 0x0000.0000 | Ethernet PHY Masked Interrupt Status and Clear | 1776 |
| **MII Management (Accessed through the EMACMIIADDR Register)** | | | | | |
| - | EPHYBMCR | RW | 0x3100 | Ethernet PHY Basic Mode Control - MR0 | 1777 |
| - | EPHYBMSR | RO | 0x7849 | Ethernet PHY Basic Mode Status - MR1 | 1779 |
| - | EPHYID1 | R | 0x2000 | Ethernet PHY Identifier Register 1 - MR2 | 1782 |
| - | EPHYID2 | R | 0xA221 | Ethernet PHY Identifier Register 2 - MR3 | 1783 |
| - | EPHYANA | RW | 0x01E1 | Ethernet PHY Auto-Negotiation Advertisement - MR4 | 1784 |
| - | EPHYANLPA | RO | 0x0000 | Ethernet PHY Auto-Negotiation Link Partner Ability - MR5 | 1786 |
| - | EPHYANER | RO | 0x0004 | Ethernet PHY Auto-Negotiation Expansion - MR6 | 1788 |
| - | EPHYANNPTR | RW | 0x2001 | Ethernet PHY Auto-Negotiation Next Page TX - MR7 | 1789 |
| - | EPHYANLNPTR | RO | 0x0000 | Ethernet PHY Auto-Negotiation Link Partner Ability Next Page - MR8 | 1791 |
| - | EPHYCFG1 | RW | 0x0000 | Ethernet PHY Configuration 1 - MR9 | 1793 |
| - | EPHYCFG2 | RW | 0x0004 | Ethernet PHY Configuration 2 - MR10 | 1796 |
| - | EPHYCFG3 | RW | 0x0000 | Ethernet PHY Configuration 3 - MR11 | 1798 |
| - | EPHYREGCTL | WO | 0x0000 | Ethernet PHY Register Control - MR13 | 1800 |
| - | EPHYADDAR | WO | 0x0000 | Ethernet PHY Address or Data - MR14 | 1802 |
| - | EPHYSTS | RO | 0x0002 | Ethernet PHY Status - MR16 | 1803 |
| - | EPHYSCR | RW | 0x0103 | Ethernet PHY Specific Control- MR17 | 1806 |
| - | EPHYMISR1 | RW | 0x0000 | Ethernet PHY MII Interrupt Status 1 - MR18 | 1809 |
| - | EPHYMISR2 | RW | 0x0000 | Ethernet PHY MII Interrupt Status 2 - MR19 | 1812 |
| - | EPHYFCSCR | RO | 0x0000 | Ethernet PHY False Carrier Sense Counter - MR20 | 1815 |
| - | EPHYRXERCNT | RO | 0x0000.0000 | Ethernet PHY Receive Error Count - MR21 | 1816 |
| - | EPHYBISTCR | RW | 0x0100 | Ethernet PHY BIST Control - MR22 | 1817 |
| - | EPHYLEDCR | RW | 0x0400 | Ethernet PHY LED Control - MR24 | 1820 |

**Table 24-24. Ethernet Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| - | EPHYCTL | RW | 0x8000 | Ethernet PHY Control - MR25 | 1821 |
| - | EPHY10BTSC | RW | 0x0000 | Ethernet PHY 10Base-T Status/Control - MR26 | 1823 |
| - | EPHYBICSR1 | RW | 0x007D | Ethernet PHY BIST Control and Status 1 - MR27 | 1825 |
| - | EPHYBICSR2 | RW | 0x05EE | Ethernet PHY BIST Control and Status 2 - MR28 | 1826 |
| - | EPHYCDCR | RO | 0x0102 | Ethernet PHY Cable Diagnostic Control - MR30 | 1827 |
| - | EPHYRCR | RW | 0x0000 | Ethernet PHY Reset Control - MR31 | 1828 |
| - | EPHYLEDCFG | RW | 0x0000.0510 | Ethernet PHY LED Configuration - MR37 | 1829 |

## 24.7 Ethernet MAC Register Descriptions

This section lists and describes the MAC registers, in numerical order by address offset. Also see “Ethernet PHY Register Descriptions” on page 1776.

## Register 1: Ethernet MAC Configuration (EMACCFG), offset 0x000

The **EMACCFG** register establishes receive and transmit operating modes. Note that the TWOKPEN bit is only applicable when the JFEN bit is 0.

Ethernet MAC Configuration (EMACCFG)

Base 0x400E.C000
Offset 0x000
Type RW, reset 0x0000.8000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | SADDR | | | TWOKPEN | reserved | CST | reserved | WDDIS | JD | reserved | JFEN | IFG | | | DISCRS |
| Type | RO | RW | RW | RW | RW | RO | RW | RO | RW | RW | RO | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PS | FES | DRO | LOOPBM | DUPM | IPC | DR | reserved | ACS | BL | | DC | TE | RE | PRELEN | |
| Type | RO | RW | RW | RW | RW | RW | RW | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | reserved | RO | 0x0 | 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. |
| 30:28 | SADDR | RW | 0x0 | Source Address Insertion or Replacement Control<br>Bit 30 specifies whether MAC address 0 or 1 registers are used during insertion or replacement for all transmitted frames. Thus for encodings 0x2-0x3, where the most significant bit is 0, the **Ethernet MAC Address 0** registers are used.<br>For encodings 0x6-0x7, the **Ethernet MAC Address 1** registers are used.<br>Bits [29:28] indicate insertion or replacement. If the value is 0x2 insertion is indicated and if the value is 0x3 replacement is indicated. |

| Value | Description |
|---|---|
| 0x0-0x1 | reserved |
| 0x2 | The Ethernet MAC inserts the content of the **Ethernet MAC Address 0 (EMACADDR0x)** registers in the SA field of all transmitted frames. |
| 0x3 | The Ethernet MAC replaces the content of the **Ethernet MAC Address 0 (EMACADDR0x)** registers in the SA field of all transmitted frames. |
| 0x4-0x5 | reserved |
| 0x6 | The MAC inserts the content of the **Ethernet MAC Address 1 (EMACADDR1x)** registers in the source address (SA) field for all transmitted frames. |
| 0x7 | The MAC replaces the content of the **Ethernet MAC Address 1(EMACADDR1x)** registers in the source address (SA) field for all transmitted frames. |

**Note:** Changes in this field take effect only on the start of a frame. If a write of this field occurs while a frame is being transmitted, only the subsequent frame can use the updated value, and the current frame does not.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 27 | TWOKPEN | RW | 0x0 | IEEE 802.3as Support for 2K Packets<br>When set, the MAC considers all frames, up to 2,000 bytes in length, as normal packets. This bit is only valid when the JFEN bit is set to 0.<br>When JFEN is set, configuring TWOKPEN has no effect on Giant Frame status.<br>Value Description<br>0 If the JFEN bit is clear, the MAC considers all received frames larger than 1,518 bytes (1,522 byes tagged) as Giant Frames.<br>1 Frames up to 2 KB are considered normal packets.<br>If the JFEN bit is clear, the MAC considers all received frames larger that 2K bytes as Giant frames. |
| 26 | reserved | RO | 0x0 | 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. |
| 25 | CST | RW | 0x0 | CRC Stripping for Type Frames<br>When set, the last four bytes (Frame Check Sequence FCS) of all frames of Ether type ((Length/Type field greater than or equal to 0x0600) are removed before forwarding the frame to the application.<br>Value Description<br>0 No bytes are removed.<br>1 The last four bytes are removed before forwarding. |
| 24 | reserved | RO | 0x0 | 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. |
| 23 | WDDIS | RW | 0x0 | Watchdog Disable<br>When this bit is set, the MAC disables the internal watchdog counter on the receiver. The MAC can receive frames of up to 16,384 bytes.<br>When this bit is cleared, the MAC does not allow more than 2,048 bytes (10,240 if JFEN is set to 1) of the frame being received. The MAC cuts off any bytes received after 2,048 bytes.<br>Value Description<br>0 Watchdog counter enabled.<br>1 Watchdog counter disabled. |
| 22 | JD | RW | 0x0 | Jabber Disable<br>When this bit is set, the MAC disables the jabber counter on the transmitter. The MAC can transfer frames of up to 16,384 bytes.<br>When this bit is clear, the MAC stops transmission if the application sends out more than 2,048 bytes of data (10,240 if JFEN is set to 1).<br>Value Description<br>0 Jabber counter enabled.<br>1 Jabber counter disabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 21 | reserved | RO | 0x0 | 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. |
| 20 | JFEN | RW | 0x0 | Jumbo Frame Enable<br>When this bit is set, the MAC allows jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.<br>Value Description<br>0 Jumbo frames create giant frame error.<br>1 Jumbo frames allowed without error. |
| 19:17 | IFG | RW | 0x0 | Inter-Frame Gap (IFG)<br>These bits control the minimum IFG between frames during transmission.<br>Value Description<br>0x0 96 bit times<br>0x1 88 bit times<br>0x2 80 bit times<br>0x3 72 bit times<br>0x4 64 bit times<br>0x5 56 bit times<br>0x6 48 bit times<br>0x7 40 bit times<br>In half-duplex mode, the minimum IFG can be configured only for 64 bit times (IFG = 0x4). Lower values are not considered. |
| 16 | DISCRS | RW | 0x0 | Disable Carrier Sense During Transmission<br>When this bit is set, the MAC transmit module ignores carrier sense in half-duplex mode. Thus, errors are not generated when there is a loss of carrier or no carrier during transmission. When this bit is clear, the MAC transmitter generates errors because of carrier sense and can even abort the transmissions.<br>Value Description<br>0 Generate errors for carrier sense errors.<br>1 Ignore carrier sense errors. |
| 15 | PS | RO | 1 | Port Select<br>This bit indicates that a 10/100 Mbps interface is supported on this device. This is a read-only bit. |
| 14 | FES | RW | 0 | Speed<br>This bit indicates the speed of the interface.<br>Value Description<br>0 10 Mbps<br>1 100 Mbps |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | DRO | RW | 0 | Disable Receive Own<br>When this bit is set, the MAC disables the reception of frames while transmitting in half-duplex mode.<br>When this bit is clear, the MAC receives all packets that are given by the PHY while transmitting.<br>Value Description<br>0 All packets are received by MAC.<br>1 Disable reception of frames.<br>**Note:** This bit is not applicable if the MAC is operating in full-duplex mode. |
| 12 | LOOPBM | RW | 0 | Loopback Mode<br>When this bit is set, the MAC operates in the loopback mode at the MII. The MII Receive clock input, EN0RXCK, is required for the loopback to work properly, because the Transmit clock is not looped-back internally.<br>Value Description<br>0 MAC does not operate in loopback mode.<br>1 MAC operates in loopback mode. |
| 11 | DUPM | RW | 0 | Duplex Mode<br>When this bit is set, the MAC operates in the full-duplex mode where it can transmit and receive simultaneously.<br>Value Description<br>0 MAC does not operate in full-duplex mode.<br>1 MAC operates in full-duplex mode. |
| 10 | IPC | RW | 0x0 | Checksum Offload<br>Value Description<br>0 The checksum offload function in the receiver is disabled and the corresponding PCE and IP HCE status bits in the frame status are always cleared.<br>1 Checksum Offload Enable<br>Setting this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | DR | RW | 0x0 | Disable Retry<br>When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current frame transmission and reports a frame abort with excessive collision error in the transmit frame status.<br>When this bit is cleared, the MAC attempts retries based on the settings of the BL field (Bits [6:5]).<br>**Note:** This bit is only applicable in half-duplex mode.<br>Value Description<br>0 MAC retries transmissions based on BL bit field.<br>1 Only one transmission is attempted by the MAC. |
| 8 | reserved | RO | 0x0 | 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. |
| 7 | ACS | RW | 0x0 | Automatic Pad or CRC Stripping<br>When this bit is set, the MAC strips the Pad or Frame Check Sequence (FCS) field on the incoming frames only if the value of the length field is less than 1,536 bytes. All received frames with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.<br>When this bit is cleared, the MAC passes all incoming frames, without modifying them, to the Host.<br>Value Description<br>0 All frames are passed to host unmodified.<br>1 MAC strips FCS field if value of length field is less than 1,536 bytes. |
| 6:5 | BL | RW | 0x0 | Back-Off Limit<br>The Back-Off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. The random integer r takes the value in the range $0 <= r < 2^k$<br>The value of k is programmed in the encodings below and is dependent on the retransmission attempt number, n. This bit is applicable only in the half-duplex mode.<br>Value Description<br>0x0 k= min (n,10), where k is the lowest value when evaluating n or 10.<br>0x1 k= min (n,8), where k is the lowest value when evaluating n or 8.<br>0x2 k= min (n,4), where k is the lowest value when evaluating n or 4.<br>0x3 k= min (n,1), where k is the lowest value when evaluating n or 1. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | DC | RW | 0x0 | Deferral Check<br>When this bit is set, the deferral check function is enabled in the MAC.<br>When the transmit state machine is deferred for more than 24,288 bit times, the MAC issues a Frame Abort status, and sets the excessive deferral error bit (EXDEFF) in the **MAC MMC Transmit Interrupt (EMACMMCTXRIS)** Register.<br>If the Jumbo frame mode (JFEN) is enabled, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active carrier sense signal (CRS) on the MII.<br>The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and then the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0 and it is restarted.<br>When this bit is clear, the deferral check function is disabled and the EMAC defers until the CRS signal goes inactive.<br>**Note:** This bit is only applicable in half-duplex mode.<br>Value Description<br>0 Deferral check function is disabled.<br>1 Deferral check function is enabled. |
| 3 | TE | RW | 0x0 | Transmitter Enable<br>When this bit is set, the transmit state machine of the MAC is enabled for transmission on the MII.<br>When this bit is clear, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames.<br>Value Description<br>0 MAC transmit state machine is disabled.<br>1 MAC transmit state machine is enabled. |
| 2 | RE | RW | 0x0 | Receiver Enable<br>When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the MII.<br>When this bit is clear, the MAC receive state machine is disabled after completion of the reception of the current frame, and does not receive any further frames from the MII.<br>Value Description<br>0 MAC receive state machine is disabled.<br>1 MAC receive state machine is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | PRELEN | RW | 0x0 | Preamble Length for Transmit Frames<br>These bits control the number of preamble bytes that are added to the beginning of every Transmit frame. The preamble reduction occurs only when the MAC is operating in the full-duplex mode.<br><br>Value Description<br>0x0 7 bytes of preamble<br>0x1 5 bytes of preamble<br>0x2 3 bytes of preamble<br>0x3 reserved |

## Register 2: Ethernet MAC Frame Filter (EMACFRAMEFLTR), offset 0x004

The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

Ethernet MAC Frame Filter (EMACFRAMEFLTR)

Base 0x400E.C000
Offset 0x004
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RA | reserved | | | | | | | | | | | | | | VTFE |
| Type | RW | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | HPF | SAF | SAIF | PCF | | DBF | PM | DAIF | HMC | HUC | PR |
| Type | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RA | RW | 0x0 | Receive All<br>When this bit is set, the MAC Receiver module passes all received frames, irrespective of whether they pass the address filter or not, to the application. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word.<br>When this bit is clear, the Receiver module passes only those frames to the application that pass the SA or DA address filter.<br>Value Description<br>0 MAC RX module only passes frames that pass the SA or DA address filter.<br>1 MAC RX module passes all received frames. |
| 30:17 | reserved | RO | 0x0 | 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. |
| 16 | VTFE | RW | 0x0 | VLAN Tag Filter Enable<br>Value Description<br>0 MAC forwards all frames regardless of match status of VLAN Tag.<br>1 MAC drops VLAN tagged frames that do not match VLAN tag comparison. |
| 15:11 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | HPF | RW | 0 | Hash or Perfect Filter<br>When this bit is set, it configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by the HMC or HUC bits in this register.<br>When this bit is clear and the HUC or HMC bit is set, the frame is passed only if it matches the Hash filter.<br>Value Description<br>0 Address filter passes a frame if it matches perfect filtering or hash filtering.<br>1 Address filter passes a frame only if it matches in the hash filter. |
| 9 | SAF | RW | 0 | Source Address Filter Enable<br>When this bit is set, the MAC compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SA match bit of Receive Status Word is set. When SA Match bit is set and the SA filter fails, the MAC drops the frame.<br>When this bit is clear, the MAC forwards the received frame to the application with the updated SA Match bit of the Receive Status Word depending on the SA address comparison.<br>Value Description<br>0 Source address filter disabled.<br>1 Source address filter enabled. |
| 8 | SAIF | RW | 0 | Source Address (SA) Inverse Filtering<br>When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA address filter.<br>When this bit is reset, frames whose SA does not match the SA registers are marked as failing the SA Address filter.<br>Value Description<br>0 Frames whose SA does not match the SA registers are marked as failing.<br>1 Frames whose SA matches the SA registers are marked as failing. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | PCF | RW | 0x0 | Pass Control Frames<br>These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames).<br><br>Value Description<br>0x00 The MAC filters all control frames from reaching application.<br>0x1 MAC forwards all control frames except PAUSE control frames to application even if they fail the address filter.<br>0x2 MAC forwards all control frames to application even if they fail the address Filter.<br>0x3 MAC forwards control frames that pass the address Filter.<br><br>Note that the following conditions should be true for the PAUSE control frames processing:<br><br>■ Condition 1: The MAC is in full-duplex mode and flow control is enabled by setting the RFE bit of MAC Flow Control Register (**EMACFLOWCTL**).<br><br>■ Condition 2: The destination address (DA) of the received frame matches the special multicast address or the **MAC Address 0 (EMACADDR0x)** Register when the UP bit of the (**EMACFLOWCTL**) is set.<br><br>■ Condition 3: The Type field of the received frame is 0x8808 and the OPCODE field is 0x0001.<br><br>This PCF field should be set to 0x1 only when Condition 1 is true; that is, the MAC is programmed to operate in the full-duplex mode and the RFE bit is enabled. Otherwise, the PAUSE frame filtering may be inconsistent. When Condition 1 is false, the PAUSE frames are considered as generic control frames. Therefore, to pass all control frames (including PAUSE control frames) when the full-duplex mode and flow control is not enabled, you should set the PCF field to 0x2 or 0x3 (as required by the application). |
| 5 | DBF | RW | 0 | Disable Broadcast Frames<br>When this bit is set, the address filtering module (AFM) filters all incoming broadcast frames. In addition, it overrides all other filter settings.<br><br>Value Description<br>0 Address filtering module passes all received broadcast frames.<br>1 Address filtering module filters all incoming broadcast frames. |
| 4 | PM | RW | 0 | Pass All Multicast<br>When set, this bit indicates that all received frames with a multicast destination address (DA) (first bit in the destination address field is 1) are passed.<br><br>Value Description<br>0 Filtering of multicast frame depends on HMC bit in this register.<br>1 All received frames with multicast DA are passed. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | DAIF | RW | 0 | Destination Address (DA) Inverse Filtering<br>When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames.<br>Value Description<br>0 Normal filtering of frames is performed.<br>1 Inverse filtering mode is enabled for DA. |
| 2 | HMC | RW | 0 | Hash Multicast<br>Value Description<br>0 MAC performs a perfect destination address (DA) filtering for multicast frames.<br>It compares the DA field with the values programmed in DA registers.<br>1 MAC performs destination address filtering of received multicast frames according to the hash table. |
| 1 | HUC | RW | 0 | Hash Unicast<br>Value Description<br>0 MAC performs a perfect destination address filtering for unicast frames.<br>It compares the DA field with the values programmed in DA registers.<br>1 MAC performs destination address filtering of unicast frames according to the hash table. |
| 0 | PR | RW | 0x0 | Promiscuous Mode<br>When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA or DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set.<br>Value Description<br>0 Incoming frames are filtered.<br>1 All incoming frames are passed. |

## Register 3: Ethernet MAC Hash Table High (EMACHASHTBLH), offset 0x008

The 64-bit Hash table is used for group address filtering. For hash filtering, the contents of the Destination Address (DA) in the incoming frame is passed through the CRC logic, and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (**Ethernet MAC Hash Table High (EMACHASHTBLH)** or **Ethernet MAC Hash Table Low (MACHASHTBLL)**), and the other five bits determine which bit within the register. A hash value of 5b'00000 selects bit 0 of the selected register, and a value of 5b'11111 selects bit 31 of the selected register.

The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

2. Perform bit-wise reversal for the value obtained in Step 1.

3. Take the upper 6 bits from the value obtained in Step 2.

For example, if the DA of the incoming frame is received as 0x1F52419CB6AF, then the internally calculated 6-bit Hash value is 0x2C and Bit 12 of **EMACHASHTBLH** register is checked for filtering. If the DA of the incoming frame is received as 0xA00A98000045, then the calculated 6-bit Hash value is 0x07 and Bit 7 of Hash Table Low register is checked for filtering.

If the corresponding bit value of the register is 0x1, the frame is accepted. Otherwise, it is rejected. If the PM (Pass All Multicast) bit is set in the **Ethernet MAC Frame Filter (EMACFRAMEFLTR)** register, then all multicast frames are accepted regardless of the multicast hash values.

Ethernet MAC Hash Table High (EMACHASHTBLH)

Base 0x400E.C000
Offset 0x008
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HTH | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HTH | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

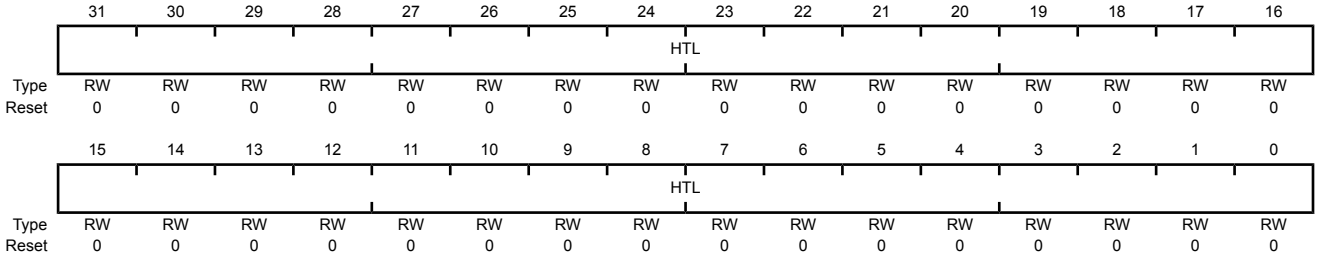
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | HTH | RW | 0x0 | Hash Table High<br>This field contains the upper 32 bits of the hash table. |

## Register 4: Ethernet MAC Hash Table Low (EMACHASHTBLL), offset 0x00C

The **MAC Hash Table Low (EMACHASHTBLL)** register contains the lower 32 bits of the hash table.

Ethernet MAC Hash Table Low (EMACHASHTBLL)

Base 0x400E.C000
Offset 0x00C
Type RW, reset 0x0000.0000



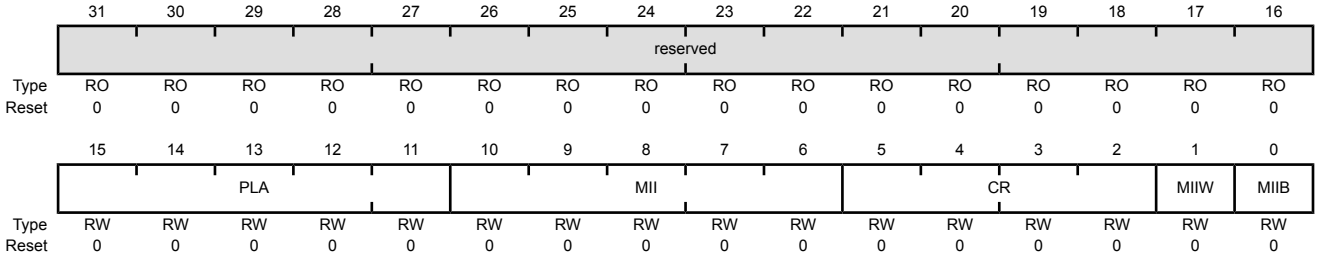

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | HTL | RW | 0x0 | Hash Table Low<br>This field contains the lower 32 bits of the hash table. |

## Register 5: Ethernet MAC MII Address (EMACMIIADDR), offset 0x010

The **Ethernet MAC MII address (EMACMIIADDR)** register controls the management cycles to the PHY through the management interface.

Ethernet MAC MII Address (EMACMIIADDR)

Base 0x400E.C000
Offset 0x010
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | 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. |
| 15:11 | PLA | RW | 0x0 | Physical Layer Address<br>This field gives the PHY address of the MII module. Values 1- 31 (0x1 to 0x1F) are available for external PHY addresses.<br>**Note:** The integrated PHY's address is 0x00. To access the integrated PHY registers, the PLA bits[15:11] must be zeros. |
| 10:6 | MII | RW | 0x0 | MII Register<br>These bits select the desired MII registers in the selected PHY device. |
| 5:2 | CR | RW | 0x0 | Clock Reference Frequency Selection<br>The clock that is sent to the MAC Clock and Status (CSR) registers is the gated System Clock (SYSCLK). The SYSCLK is divided down through the CR field to produce an MDC clock that is approximately between the frequency range 1.0 MHz - 2.5 MHz. The application must program the appropriate CR field based on the System Clock input. |

| Value | Description |
|---|---|
| 0x0 | The frequency of the System Clock is 60 to 100 MHz providing a MDIO clock of SYSCLK/42. |
| 0x1 | The frequency of the System Clock is 100 to 150 MHz providing a MDIO clock of SYSCLK/62. |
| 0x2 | The frequency of the System Clock is 20-35 MHz providing a MDIO clock of System Clock/16. |
| 0x3 | The frequency of the System Clock is 35 to 60 MHz providing a MDIO clock of System Clock/26. |
| 0x4-0xF | reserved |

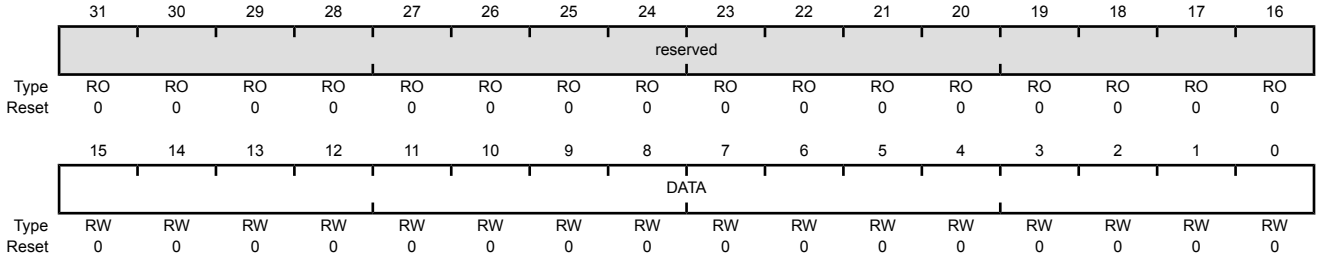
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | MIIW | RW | 0x0 | MII Write<br><br>Value Description<br>0 Read operation is active and read data is placed in the MII Data register (**EMACMIIDATA**).<br>1 PHY is notified that this is a write operation using the MII Data Register (**EMACMIIDATA**). |
| 0 | MIIB | RW | 0x0 | MII Busy<br><br>Indicates whether the MII is busy with a read or write access.<br><br>This bit should read logic 0 before writing to the **EMACMIIADDR** register or the **EMACMIIDATA** register. During a PHY register access, the software sets this bit to indicate that a read or write access is in progress.<br><br>The **EMACMIIDATA** register is invalid until this bit is cleared by the MAC. Therefore, **EMACMIIDATA** should be kept valid until the MAC clears this bit during a PHY Write operation. Similarly for a read operation, the contents of **EMACMIIDATA** are not valid until this bit is cleared.<br><br>Value Description<br>0 **EMACMIIADDR** and **EMACMIIDATA** are available for reads and writes.<br>1 Read or write access is in progress.<br><br>The subsequent read or write operations should happen only after the previous operation is complete. |

## Register 6: Ethernet MAC MII Data Register (EMACMIIDATA), offset 0x014

The **Ethernet MAC MII Data (EMACMIIDATA)** register holds data that is written to and read from the PHY register located at the address specified by the PLA and MII bit fields of the **Ethernet MAC MII Address (EMACMIIADDR)** register.

Ethernet MAC MII Data Register (EMACMIIDATA)

Base 0x400E.C000
Offset 0x014
Type RW, reset 0x0000.0000



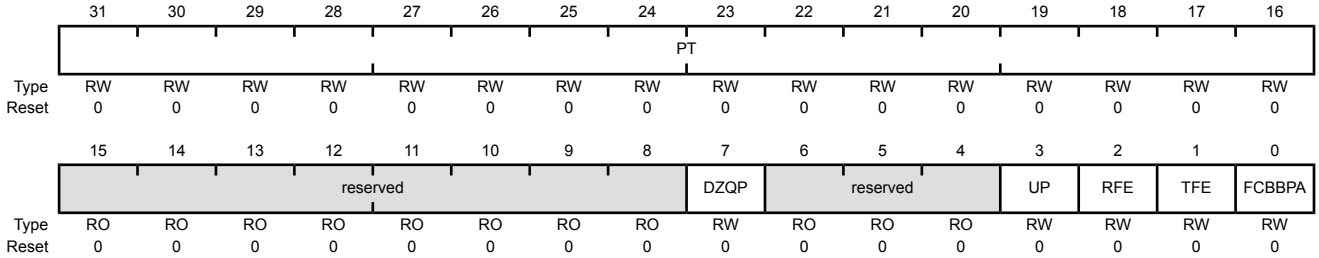

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | DATA | RW | 0x0 | MII Data<br>This field contains the 16-bit data value read from the PHY after a management read operation or the 16-bit data value to be written to the PHY before a management write operation. |

## Register 7: Ethernet MAC Flow Control (EMACFLOWCTL), offset 0x018

The **Ethernet MAC Flow Control (EMACFLOWCTL)** register controls the generation and reception of the control (pause command) frames by the MAC's Flow control module. A write to a register with the FCBBPA (bit 0) set to 1 triggers the Flow Control block to generate a pause control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the pause time (PT) value from this register is used in the pause time field of the control frame. The FCBBPA bit is cleared by the hardware once the control frame is transferred onto the cable. The Host must make sure that the busy bit is clear before writing to the register.

Ethernet MAC Flow Control (EMACFLOWCTL)

Base 0x400E.C000
Offset 0x018
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | PT | RW | 0x0 | Pause Time<br>This field holds the value to be used in the pause time field in the transmit control frame.<br>For example, if these bits are set to 0x0100 then 256 slot times are used in the Pause Time field in the transmit control frame. |
| 15:8 | reserved | RO | 0x0 | 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. |
| 7 | DZQP | RW | 0 | Disable Zero-Quanta Pause<br>When this bit is set, it disables the automatic generation of the Zero-Quanta Pause Control frames on the deassertion of the flow-control signal from the FIFO layer. When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.<br>Value Description<br>0 Automatic Zero-Quanta Pause Control generation is enabled.<br>1 Automatic Zero-Quanta Pause Control generation is disabled. |
| 6:4 | reserved | RO | 0x0 | 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. |

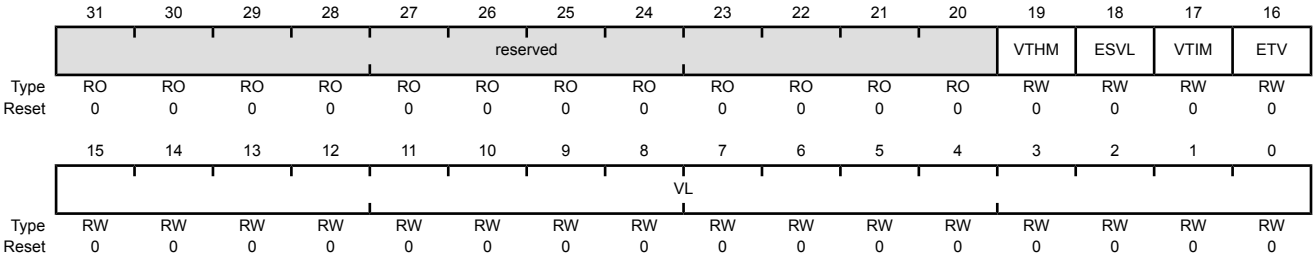
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | UP | RW | 0x0 | Unicast Pause Frame Detect<br><br>Value Description<br>0 MAC detects only a Pause frame with the unique multicast address specified in the 802.3x standard.<br>1 The MAC detects the pause frames with the station's unicast address specified in the **EMACADDR0H** and **EMACADDR0L** register, in addition to detecting pause frames with the unique multicast address. |
| 2 | RFE | RW | 0x0 | Receive Flow Control Enable<br><br>Value Description<br>0 The decode function of the pause frame is disabled.<br>1 The MAC decodes the received pause frame and disables its transmitter for a specified (pause) time. |
| 1 | TFE | RW | 0x0 | Transmit Flow Control Enable<br><br>Value Description<br>0 In full duplex mode, the flow control operation in the MAC is disabled, and the MAC does not transmit any pause frames.<br>In half duplex mode, the back-pressure feature is disabled.<br>1 In the full-duplex mode, the MAC enables the flow control operation to transmit pause frames.<br>In half-duplex mode, the MAC enables the back-pressure operation. |
| 0 | FCBBPA | RW | 0 | Flow Control Busy or Back-pressure Activate<br>In the full-duplex mode, this bit should be read as 0x0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 0x1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC resets this bit to 0x0. The **EMACFLOWCTL** register should not be written to until this bit is cleared.<br><br>Value Description<br>0 No effect<br>1 In the full-duplex mode, a pause control frame is enabled. In half-duplex mode, a back-pressure function is enabled if the TFE bit is set. |

## Register 8: Ethernet MAC VLAN Tag (EMACVLANTG), offset 0x01C

The **Ethernet MAC VLAN Tag (MACVLANTG)** register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with the value 0x8100, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1,518 bytes to 1,522 bytes.

Ethernet MAC VLAN Tag (EMACVLANTG)

Base 0x400E.C000
Offset 0x01C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0 | 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. |
| 19 | VTHM | RW | 0 | VLAN Tag Hash Table Match Enable<br><br>Value Description<br>0 The VLAN Hash Match operation is not performed.<br>1 The most significant four bits of the VLAN tag's CRC are used to index the content of the **MAC VLAN Hash Table (EMACVLANHASH)** register. A value of 1 in the **EMACVLANHASH** register, corresponding to the index, indicates that the frame matched the VLAN hash table.<br>When Bit 16 (ETV) is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison whereas when ETV is reset, the CRC of the 16-bit VLAN tag is used for comparison. |
| 18 | ESVL | RW | 0 | Enable S-VLAN<br><br>Value Description<br>0 The MAC transmitter does not recognize S-VLAN frames as valid VLAN tagged frames.<br>1 The MAC transmitter and receiver considers the S-VLAN (Type = 0x88A8) frames as valid VLAN tagged frames. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | VTIM | RW | 0 | VLAN Tag Inverse Match Enable<br><br>Value Description<br>0 VLAN perfect matching is enabled. The frames with matched VLAN tag are marked as matched.<br>1 VLAN tag inverse matching is enabled. The frames that do not have matching VLAN tag are marked as matched. |
| 16 | ETV | RW | 0 | Enable 12-Bit VLAN Tag Comparison<br><br>Value Description<br>0 All 16 bits of the 15th and 16th bytes of the received VLAN frame are used for comparison and VLAN hash filtering.<br>1 A 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag.<br>Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. Similarly, when enabled, only 12 bits of the VLAN tag in the received frame are used for hash-based VLAN filtering. |
| 15:0 | VL | RW | 0x0000 | VLAN Tag Identifier for Receive Frames<br>This field contains the 802.1Q VLAN tag to identify the VLAN frames and is compared to the 15th and 16th bytes of the frames being received for VLAN frames. The following list describes the bits of this field:<br>■ Bits [15:13]: User Priority<br>■ Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>■ Bits[11:0]: VLAN tag's VLAN Identifier (VID) field<br>When the ETV bit is set, only the VID field (Bits[11:0]) is used for comparison. If VL[11:0] is all zeros when the ETV is set, the MAC does not check the fifteenth and 16th bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 or 0x88A8 as VLAN frames. |

## Register 9: Ethernet MAC Status (EMACSTATUS), offset 0x024

The **Ethernet MAC Status (EMACSTATUS)** register gives the status of all main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.

Ethernet MAC Status (EMACSTATUS)

Base 0x400E.C000
Offset 0x024
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | TXFF | TXFE | reserved | TWC | TRC | | TXPAUSED | TFC | | TPE |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | RXF | | reserved | RRC | | RWC | reserved | RFCFC | | RPE |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:26 | reserved | RO | 0x00 | 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. |
| 25 | TXFF | RO | 0 | TX/RX Controller TX FIFO Full Status<br>Value Description<br>0 The TX/RX Controller TX FIFO is not full.<br>1 The TX/RX Controller TX FIFO is full. Therefore, the TX/RX Controller cannot accept any more frames for transmission. |
| 24 | TXFE | RO | 0 | TX/RX Controller TX FIFO Not Empty Status<br>Value Description<br>0 TX/RX Controller TX FIFO is empty.<br>1 TX/RX Controller TX FIFO is not empty and some data is left for transmission. |
| 23 | reserved | RO | 0x0 | 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. |
| 22 | TWC | RO | 0 | TX/RX Controller TX FIFO Write Controller Active Status<br>Value Description<br>0 TX/RX Controller's TX FIFO write controller is inactive.<br>1 TX/RX Controller's TX FIFO write controller is active and transferring data to the TX FIFO. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 21:20 | TRC | RO | 0x0 | TX/RX Controller's TX FIFO Read Controller Status<br>This field indicates the state of the TX FIFO read controller:<br><br>Value Description<br>0x0 IDLE state<br>0x1 READ state (transferring data to MAC transmitter)<br>0x2 Waiting for TX Status from MAC transmitter<br>0x3 Writing the received TX Status or flushing the TX FIFO |
| 19 | TXPAUSED | RO | 0 | MAC Transmitter PAUSE<br><br>Value Description<br>0 MAC transmitter is not in PAUSE mode.<br>1 Indicates that the MAC transmitter is in the PAUSE condition (in the full-duplex only mode) and hence does not schedule any frame for transmission. |
| 18:17 | TFC | RO | 0x0 | MAC Transmit Frame Controller Status<br>This field indicates the state of the MAC Transmit Frame Controller module:<br><br>Value Description<br>0x0 IDLE state<br>0x1 Waiting for status of previous frame or IFG or backoff period to be over<br>0x2 Generating and transmitting a PAUSE control frame (in the full-duplex mode)<br>0x3 Transferring input frame for transmission |
| 16 | TPE | RO | 0 | MAC MII Transmit Protocol Engine Status<br><br>Value Description<br>0 MAC MII transmit protocol engine is not actively transmitting data.<br>1 MAC MII transmit protocol engine is actively transmitting data and is not in the IDLE state. |
| 15:10 | reserved | RO | 0x00 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9:8 | RXF | RO | 0x0 | TX/RX Controller RX FIFO Fill-level Status<br>This field gives the status of the fill-level of the RX FIFO. The FIFO threshold is programmed by the TCC field in the **Ethernet MAC DMA Operation Mode (EMACDMAOPMODE)** register.:<br>Value Description<br>0x0 RX FIFO Empty<br>0x1 RX FIFO fill level is below the flow-control deactivate threshold<br>0x2 RX FIFO fill level is above the flow-control activate threshold<br>0x3 RX FIFO Full |
| 7 | reserved | RO | 0 | 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. |
| 6:5 | RRC | RO | 0x0 | TX/RX Controller Read Controller State<br>This field gives the state of the RX FIFO read Controller:<br>Value Description<br>0x0 IDLE state<br>0x1 Reading frame data<br>0x2 Reading frame status (or timestamp)<br>0x3 Flushing the frame data and status |
| 4 | RWC | RO | 0 | TX/RX Controller RX FIFO Write Controller Active Status<br>Value Description<br>0 The MTL RX FIFO Write Controller is inactive.<br>1 MTL RX FIFO Write Controller is active and is transferring a received frame to the FIFO. |
| 3 | reserved | RO | 0 | 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. |
| 2:1 | RFCFC | RO | 0x0 | MAC Receive Frame Controller FIFO Status<br>When high, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Frame Controller Module. |
| 0 | RPE | RO | 0x0 | MAC MII Receive Protocol Engine Status<br>Value Description<br>0 MAC MII receive protocol engine is not actively receiving data.<br>1 Indicates that the MAC MII receive protocol engine is actively receiving data and not in IDLE state. |

## Register 10: Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF), offset 0x028

This is the address through which the application writes or reads the remote wake-up frame filter registers. Note that to load values in the **Ethernet MAC Wake-up Frame Filter (EMACRWUFF)** register, the entire register must be written. The **Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF)** register is a pointer to eight wake-up frame filter registers. The **Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF)** register is loaded by sequentially loading the eight register values. Eight sequential writes to this address programs all of the remote wake-up frame filter registers. Similarly, eight sequential reads from the **EMACRWUFF** register reads all wake-up frame registers.

Ethernet MAC Remote Wake-Up Frame Filter (EMACRWUFF)

Base 0x400E.C000
Offset 0x028
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WAKEUPFIL | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WAKEUPFIL | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WAKEUPFIL | RW | 0x0 | Remote Wake-Up Frame Filter |

## Register 11: Ethernet MAC PMT Control and Status Register (EMACPMTCTLSTAT), offset 0x02C

The **Ethernet MAC PMT Control and Status (EMACPMTCTLSTAT)** programs and monitors wake-up events.

Ethernet MAC PMT Control and Status Register (EMACPMTCTLSTAT)

Base 0x400E.C000
Offset 0x02C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WUPFRRST | reserved | | | | RWKPTR | | | reserved | | | | | | | |
| Type | RW | RO | RO | RO | RO | RW | RW | RW | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | GLBLUCAST | reserved | | WUPRX | MGKPRX | reserved | | WUPFREN | MGKPKTEN | PWRDWN |
| Type | RO | RO | RO | RO | RO | RO | RW | RO | RO | RO | RO | RO | RO | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | WUPFRRST | RW | 0 | Wake-Up Frame Filter Register Pointer Reset<br><br>Value Description<br>0 No effect.<br>1 Resets the **MAC Remote Wake-Up Frame Filter (EMACRWUFF)** register pointer to 0x0. It is automatically cleared after one clock cycle. |
| 30:27 | reserved | RO | 0 | 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. |
| 26:24 | RWKPTR | RW | 0 | Remote Wake-Up FIFO Pointer<br>This gives the current value (0 to 7) of the **MAC Remote Wake-Up Frame Filter (EMACRWUFF)** register pointer. The contents of the **EMACRWUFF** Register are transferred to the receive clock domain when a write occurs to that register when this pointer value equals 7. |
| 23:10 | reserved | RO | 0 | 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. |
| 9 | GLBLUCAST | RW | 0 | Global Unicast<br><br>Value Description<br>0 No Effect.<br>1 Enables any unicast packet filtered by the MAC Destination Address Filter (DAF) module's address recognition to be a wake-up frame. |
| 8:7 | reserved | RO | 0 | 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. |

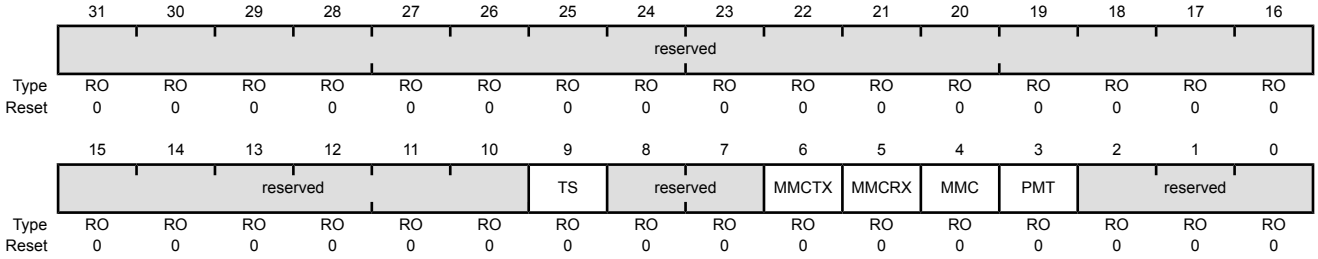
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | WUPRX | RO | 0x0 | Wake-Up Frame Received<br><br>Value Description<br>0 No effect.<br>1 The power management event is generated because of the reception of a wake-up frame.<br>This bit is cleared whenever the register is read. |
| 5 | MGKPRX | RO | 0 | Magic Packet Received<br><br>Value Description<br>0 No effect.<br>1 The power management event is generated because of the reception of a magic packet.<br>This bit is cleared whenever the register is read. |
| 4:3 | reserved | RO | 0x0 | 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. |
| 2 | WUPFREN | RW | 0 | Wake-Up Frame Enable<br><br>Value Description<br>0 Wake-up frame does not affect power management control.<br>1 Generation of a power management event in response to the reception of a wake-up frame is enabled. |
| 1 | MGKPKTEN | RW | 0 | Magic Packet Enable<br><br>Value Description<br>0 Magic packet reception does not affect power management control.<br>1 Generation of a power management event bin response to the reception of a magic packet is enabled. |
| 0 | PWRDWN | RW | 0 | Power Down<br>When set, the MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame. Upon reception, PWRDWN is self-cleared and the power-down mode is disabled. Software can also clear this bit before the expected magic packet or wake-up frame is received. The frames, received by the MAC after this bit is cleared, are forwarded to the application.<br><br>Value Description<br>0 Frames are forwarded to application.<br>1 MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame. |

## Register 12: Ethernet MAC Raw Interrupt Status (EMACRIS), offset 0x038

The **Ethernet MAC Raw Interrupt Status (EMACRIS)** register identifies the events in the MAC that can generate interrupt.

Ethernet MAC Raw Interrupt Status (EMACRIS)

Base 0x400E.C000
Offset 0x038
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.0 | 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. |
| 9 | TS | RO | 0x0 | Timestamp Interrupt Status<br>This bit is cleared by reading the TSS0VF bit in the **MAC timestamp Status Register (EMACTIMSTAT)** register.<br>In this mode, this bit is cleared after the completion of the read of this bit. In all other modes, this bit is reserved.<br>Value Description<br>0 No timestamp interrupt is present.<br>1 When the advanced timestamp feature is enabled, this bit indicates one of two conditions is true:<br>■ The system time value equals or exceeds the value specified in the **EMAC Target Time Seconds Register (EMACTARGSEC)** and **MAC Target Time Nanoseconds (EMACTARGNANO)** registers.<br>■ There is an overflow in the **EMAC Target Time Seconds (EMACTARGSEC)** register.<br>When default timestamping is enabled, this bit indicates that the system time value is equal to or exceeds the value specified in the **EMAC Target Time** registers. In this mode, this bit is cleared after the completion of the read of this bit. |
| 8:7 | reserved | RO | 0x0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | MMCTX | RO | 0 | MMC Transmit Interrupt Status<br>This bit is cleared when all of the bits in the **MAC MMC Transmit Interrupt (EMACMMCTXRIS)** register are clear.<br>Value Description<br>0 No interrupts exist in the **MAC MMC Transmit Interrupt (EMACMMCTXRIS)** register.<br>1 Indicates an interrupt has been generated in the **MAC MMC Transmit Interrupt (EMACMMCTXRIS)** register. |
| 5 | MMCRX | RO | 0 | MMC Receive Interrupt Status<br>This bit is cleared when all of the bits in the **Ethernet MAC MMC Receive Interrupt (EMACMMCRXRIS)** register are clear.<br>Value Description<br>0 No interrupts exist in the **MAC MMC Receive Interrupt (EMACMMCTXRIS)** register.<br>1 Indicates an interrupt has been generated in the **MAC MMC Receive Interrupt (EMACMMCRXRIS)** register. |
| 4 | MMC | RO | 0x0 | MMC Interrupt Status<br>Value Description<br>0 Indicates the MMC-related Interrupt bits [6:5] in this register are clear.<br>1 Indicates that one or more of the MMC-related interrupt bits [6:5] in this register are set. |
| 3 | PMT | RO | 0x0 | PMT Interrupt Status<br>Value Description<br>0 This bit is cleared when both Bits[6:5] are cleared because of a read operation to the **MAC PMT Control and Status Register (EMACPMTCTRLSTAT)** register.<br>1 Indicates a Magic packet or Wake-on-LAN frame is received in the power-down mode (see Bits 5 and 6 in the **MACPMTCTRLSTAT** register). |
| 2:0 | reserved | RO | 0x0 | 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. |

## Register 13: Ethernet MAC Interrupt Mask (EMACIM), offset 0x03C

The **Ethernet MAC Interrupt Mask (EMACIM)** Register bits enables the application to mask the interrupt signal caused by the corresponding event in the **Ethernet MAC Raw Interrupt Status (EMACRIS)** Register.

Ethernet MAC Interrupt Mask (EMACIM)

Base 0x400E.C000
Offset 0x03C
Type RW, reset 0x0000.0000



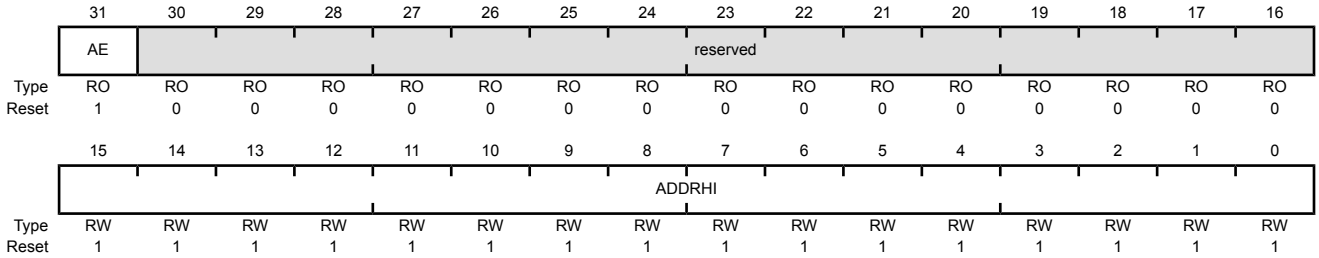

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9 | TSI | RW | 0x0 | Timestamp Interrupt Mask<br>Value Description<br>0 The TSI interrupt status bit in the **MAC Raw Interrupt Status (EMACRIS)** register is not masked and can cause an interrupt.<br>1 The assertion of the TIS interrupt status bit in the **MAC Raw Interrupt Status (EMACRIS)** register is masked and does not cause an interrupt. |
| 8:4 | reserved | RO | 0x0 | 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. |
| 3 | PMT | RW | 0x0 | PMT Interrupt Mask<br>Value Description<br>0 The PMT interrupt status bit in the **MAC Raw Interrupt Status (EMACRIS)** register is not masked and can cause an interrupt.<br>1 The assertion of the PMT interrupt status bit in the **MAC Raw Interrupt Status (EMACRIS)** register is masked and does not cause an interrupt. |
| 2:0 | reserved | RO | 0 | 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. |

## Register 14: Ethernet MAC Address 0 High (EMACADDR0H), offset 0x040

The **Ethernet MAC Address 0 High (EMACADDR0H)** register holds the upper 16 bits of the first 6-byte MAC address of the station. The first (Destination Address) DA byte that is received on the MII interface corresponds to the least significant byte (Bits [7:0]) of the **MAC Address 0 Low Register (EMACADDR0L)** register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the **EMAC Address 0 (EMACADDR0x)** register [47:0] is compared with 0x665544332211.

Ethernet MAC Address 0 High (EMACADDR0H)

Base 0x400E.C000
Offset 0x040
Type RW, reset 0x8000.FFFF



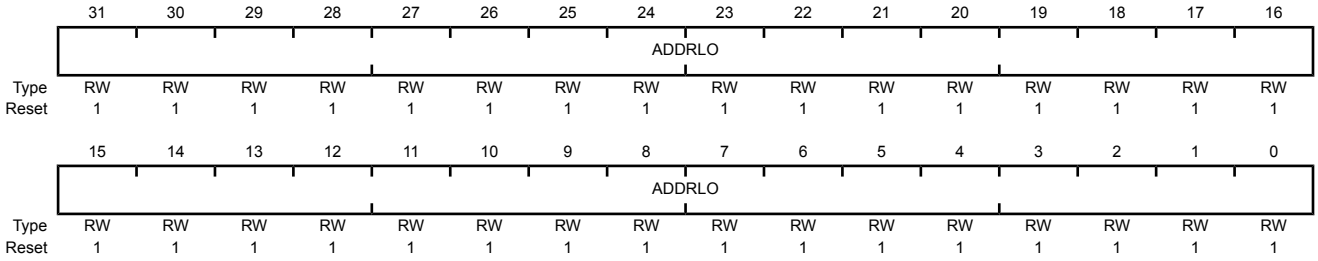

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | AE | RO | 1 | Address Enable<br>This register is always valid, so this bit is set always set to 1. |
| 30:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | ADDRHI | RW | 0xFFFF | MAC Address0 [47:32]<br>This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. The MAC uses this field for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames. |

## Register 15: Ethernet MAC Address 0 Low Register (EMACADDR0L), offset 0x044

The **MAC Address 0 Low Register (EMACADDR0L)** register holds the lower 32 bits of the first 6-byte MAC address of the station.

Ethernet MAC Address 0 Low Register (EMACADDR0L)

Base 0x400E.C000
Offset 0x044
Type RW, reset 0xFFFF.FFFF



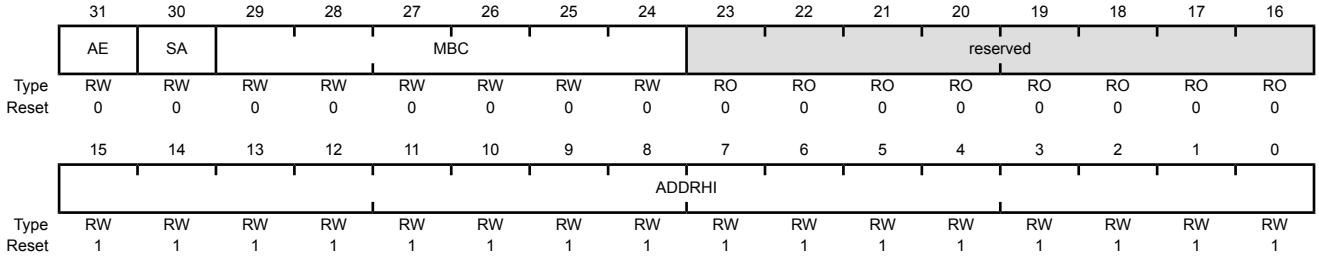

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDRLO | RW | 0xFFFFFFFF | MAC Address0 [31:0]<br>This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames. |

## Register 16: Ethernet MAC Address 1 High (EMACADDR1H), offset 0x048

The **MAC Address 1 High (EMACADDR1H)** register holds the upper 16 bits of the second 6-byte MAC address of the station.

Ethernet MAC Address 1 High (EMACADDR1H)

Base 0x400E.C000
Offset 0x048
Type RW, reset 0x0000.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | AE | RW | 0 | Address Enable<br><br>Value Description<br>0 The address filter module ignores the second address for filtering.<br>1 The address filter module uses the second address for perfect filtering. |
| 30 | SA | RW | 0 | Source Address<br><br>Value Description<br>0 When this bit is reset, MAC Address1[47:0] is used to compare with the DA fields of the received frame.<br>1 When this bit is set, MAC Address1[47:0] is used to compare with the SA fields of the received frame. |
| 29:24 | MBC | RW | 0x00 | Mask Byte Control<br><br>Mask control bits are provided for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:<br><br>■ Bit 29: ADDRHI [15:8] of **EMACADDR1H** Register<br>■ Bit 28: ADDRHI [7:0] of **EMACADDR1H**<br>■ Bit 27: ADDRLO [31:24] of **EMACADDR1L** register<br>■ Bit 26: ADDRLO [23:16] of **EMACADDR1L** register<br>■ Bit 25: ADDRLO [15:8] of **EMACADDR1L** register<br>■ Bit 24: ADDRLO [7:0] of **EMACADDR1L** register<br><br>A group of addresses (known as group address filtering) can be filtered by masking one or more bytes of the address. |

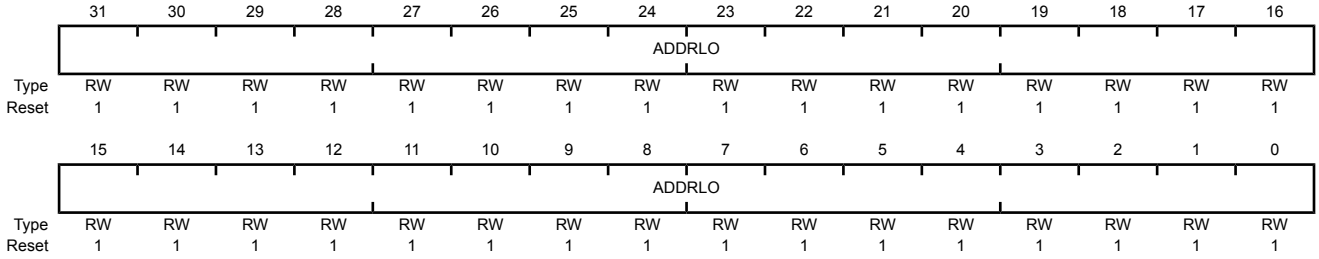
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | ADDRHI | RW | 0xFFFF | MAC Address1 [47:32]<br>This field contains the upper 16 bits (47:32) of the second 6-byte MAC address. |

## Register 17: Ethernet MAC Address 1 Low (EMACADDR1L), offset 0x04C

The **MAC Address 1 Low (EMACADDR1L)** register holds the lower 32 bits of the second 6-byte MAC address of the station.

Ethernet MAC Address 1 Low (EMACADDR1L)

Base 0x400E.C000
Offset 0x04C
Type RW, reset 0xFFFF.FFFF



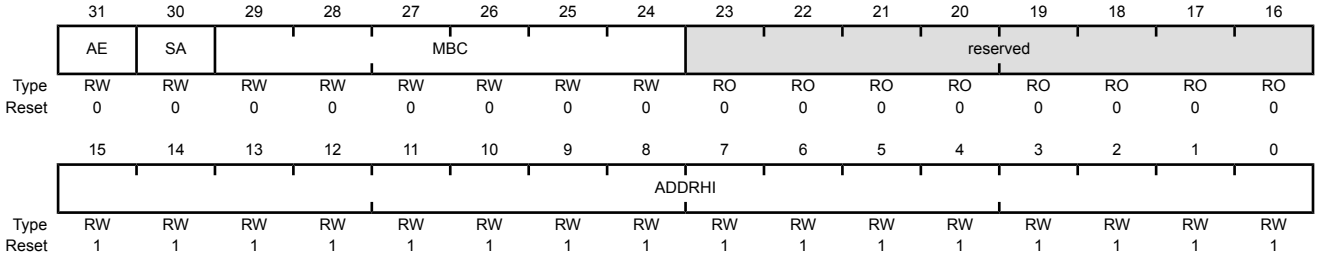

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDRLO | RW | 0xFFFFFFFF | MAC Address1 [31:0]<br>This field contains the lower 32 bits of the second 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process. |

## Register 18: Ethernet MAC Address 2 High (EMACADDR2H), offset 0x050

The **MAC Address 2 High (EMACADDR2H)** register holds the upper 16 bits of the third 6-byte MAC address of the station.

Ethernet MAC Address 2 High (EMACADDR2H)

Base 0x400E.C000
Offset 0x050
Type RW, reset 0x0000.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | AE | RW | 0x0 | Address Enable |

| Value | Description |
|---|---|
| 0 | The address filter module ignores the third address for filtering. |
| 1 | The address filter module uses the third address for perfect filtering. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 30 | SA | RW | 0x0 | Source Address |

| Value | Description |
|---|---|
| 0 | When this bit is reset, the MAC Address2[47:0] is used to compare with the DA fields of the received frame. |
| 1 | When this bit is set, the MAC Address2[47:0] is used to compare with the SA fields of the received frame. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 29:24 | MBC | RW | 0x0 | Mask Byte Control |

Mask control bits are provided for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:

- Bit 29: `ADDRHI` [15:8] of **EMACADDR2H** register
- Bit 28: `ADDRHI` [7:0] of **EMACADDR2H** register
- Bit 27: `ADDRLO` [31:24] of **EMACADDR2L** register
- Bit 26: `ADDRLO` [23:16] of **EMACADDR2L** register
- Bit 25: `ADDRLO` [15:8] of **EMACADDR2L** register
- Bit 24: ADDRLO [7:0] of **EMACADDR2L** register

A group of addresses (known as group address filtering) can be filtered by masking one or more bytes of the address.

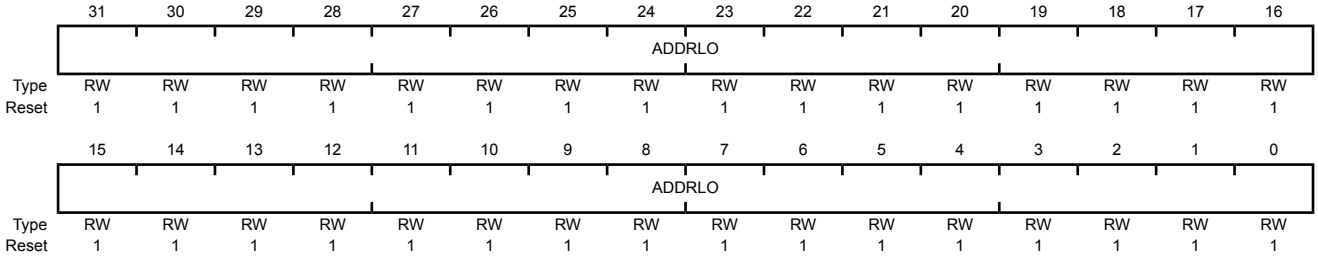
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | ADDRHI | RW | 0xFFFF | MAC Address2 [47:32]<br>This field contains the upper 16 bits [47:32] of the third 6-byte MAC address. |

## Register 19: Ethernet MAC Address 2 Low (EMACADDR2L), offset 0x054

The MAC Address2 Low register holds the lower 32 bits of the third 6-byte MAC address of the station.

Ethernet MAC Address 2 Low (EMACADDR2L)

Base 0x400E.C000
Offset 0x054
Type RW, reset 0xFFFF.FFFF



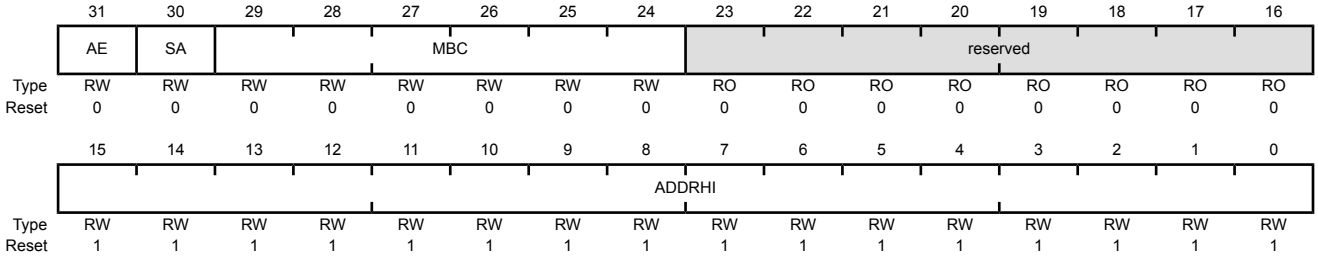

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDRLO | RW | 0xFFFFFFFF | MAC Address2 [31:0]<br>This field contains the lower 32 bits of the third 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process. |

## Register 20: Ethernet MAC Address 3 High (EMACADDR3H), offset 0x058

The **Ethernet MAC Address 3 High (EMACADDR3H)** register holds the upper 16 bits of the fourth 6-byte MAC address of the station.

Ethernet MAC Address 3 High (EMACADDR3H)

Base 0x400E.C000
Offset 0x058
Type RW, reset 0x0000.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | AE | RW | 0x0 | Address Enable<br><br>Value Description<br>0 The address filter module ignores the third address for filtering.<br>1 The address filter module uses the third address for perfect filtering. |
| 30 | SA | RW | 0x0 | Source Address<br><br>Value Description<br>0 When this bit is reset, the MAC Address3[47:0] is used to compare with the DA fields of the received frame.<br>1 When this bit is set, the MAC Address3[47:0] is used to compare with the SA fields of the received frame. |
| 29:24 | MBC | RW | 0x0 | Mask Byte Control<br><br>Mask control bits are provided for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:<br><br>■ Bit 29: ADDRHI [15:8] of **EMACADDR3H** register<br>■ Bit 28: ADDRHI [7:0] of **EMACADDR3H** register<br>■ Bit 27: ADDRLO [31:24] of **EMACADDR3L** register<br>■ Bit 26: ADDRLO [23:16] of **EMACADDR3L** register<br>■ Bit 25: ADDRLO [15:8] of **EMACADDR3L** register<br>■ Bit 24: ADDRLO [7:0] of **EMACADDR3L** register<br><br>A group of addresses (known as group address filtering) can be filtered by masking one or more bytes of the address. |

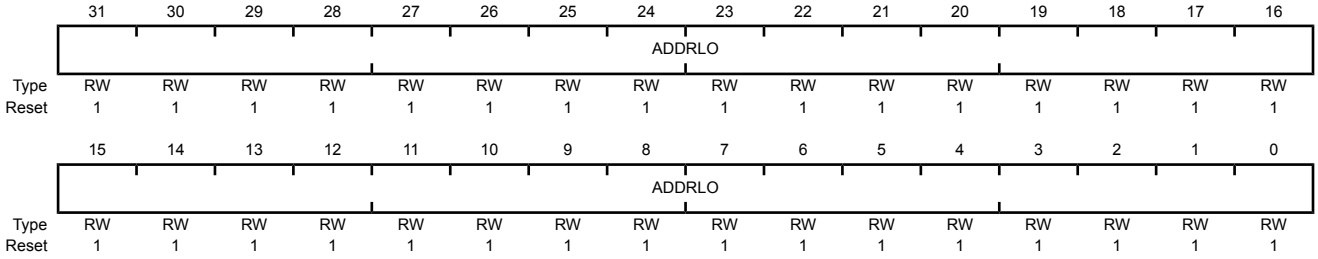
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | ADDRHI | RW | 0xFFFF | MAC Address3 [47:32]<br>This field contains the upper 16 bits [47:32] of the fourth 6-byte MAC address. |

## Register 21: Ethernet MAC Address 3 Low (EMACADDR3L), offset 0x05C

The **Ethernet MAC Address 3 Low (EMACADDR3L)** register holds the lower 32 bits of the fourth 6-byte MAC address of the station.

Ethernet MAC Address 3 Low (EMACADDR3L)

Base 0x400E.C000
Offset 0x05C
Type RW, reset 0xFFFF.FFFF



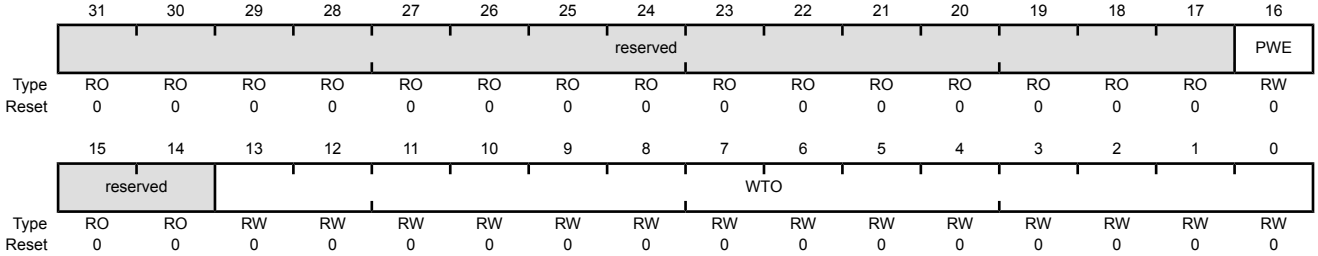

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDRLO | RW | 0xFFFFFFFF | MAC Address3 [31:0]<br>This field contains the lower 32 bits of the fourth 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process. |

## Register 22: Ethernet MAC Watchdog Timeout (EMACWDOGTO), offset 0x0DC

This register controls the watchdog counter for received frames.

Ethernet MAC Watchdog Timeout (EMACWDOGTO)

Base 0x400E.C000
Offset 0x0DC
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0 | 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. |
| 16 | PWE | RW | 0 | Programmable Watchdog Enable<br><br>Value Description<br>0 The watchdog timeout for a received frame is controlled by setting the WD and JE bits in the **EMACCFG** register.<br>1 When the WD bit of the **EMACCFG** register is clear, the WTO field is used as a watchdog timeout for a received frame. |
| 15:14 | reserved | RO | 0 | 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. |
| 13:0 | WTO | RW | 0 | Watchdog Timeout<br><br>When the PWE bit in the **EMACWDOGTO** register is set and the WD bit of the **EMACCFG** register is clear, this field is used as a watchdog timeout value for a received frame. If the length of a received frame exceeds the value of this field, such frame is terminated and declared an error frame.<br><br>**Note:** When the PWE bit is set the value in this field should be more than 1,522 (0x05F2). Otherwise, valid, tagged IEEE 802.3-frames are declared as error frames and are dropped. |

## Register 23: Ethernet MAC MMC Control (EMACMMCCTRL), offset 0x100

The MMC Control register establishes the operating mode of the management counters.

**Note:** The CNTRST bit has higher priority than the CNTPRST. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST is not set.

Ethernet MAC MMC Control (EMACMMCCTRL)

Base 0x400E.C000
Offset 0x100
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | UCDBC | reserved | | CNTPRSTLVL | CNTPRST | CNTFREEZ | RSTONRD | CNTSTPRO | CNTRST |
| RO | RO | RO | RO | RO | RO | RO | RW | RO | RO | RW | RW | RW | RW | RW | RW |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x00 | 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. |
| 8 | UCDBC | RW | 0 | Update MMC Counters for Dropped Broadcast Frames<br>Value Description<br>0 MMC Counters are not updated for dropped broadcast frames.<br>1 MAC updates all related MMC counters for broadcast frames dropped due to setting of DBF bit (Disable Broadcast Frames) of the **MAC Frame Filter(EMACFRAMEFLTR)** register. |
| 7:6 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | CNTPRSTLVL | RW | 0 | Full/Half Preset Level Value<br><br>Value Description<br>0 If CNTPRST is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF.F800 (half value - 2 KB) and all frame-counters get preset to 0x7FFF.FFF0 (half value - 16).<br>1 If CNTPRST is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF.F800 (full value - 2 KB) and all frame-counters gets preset to 0xFFFF_FFF0 (full value - 16).<br><br>This bit, along with CNTPRST, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.<br><br>For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0. |
| 4 | CNTPRST | RW | 0 | Counters Preset<br><br>Value Description<br>0 Normal operation.<br>1 All counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. The CNTPRST bit is cleared automatically after 1 clock cycle.<br><br>This bit, along with CNTPRSTLVL, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. |
| 3 | CNTFREEZ | RW | 0 | MMC Counter Freeze<br><br>When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset, no MMC counter is updated because of any transmitted or received frame. If any MMC counter is read with the RSTONRD bit set, then that counter is also cleared in this mode.<br><br>Value Description<br>0 MMC counters are updated when a frame is transmitted or received.<br>1 When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received frame. |
| 2 | RSTONRD | RW | 0x0 | Reset on Read<br><br>Value Description<br>0 No effect.<br>1 MMC counters are reset to zero after a read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CNTSTPRO | RW | 0 | Counters Stop Rollover<br><br>Value Description<br>0 No effect.<br>1 After reaching maximum value, the MMC counters do not roll over to zero. |
| 0 | CNTRST | RW | 0x0 | Counters Reset<br><br>Value Description<br>0 No effect<br>1 All MMC counters are reset. This bit is cleared automatically after one clock cycle. |

## Register 24: Ethernet MAC MMC Receive Raw Interrupt Status (EMACMMCRXRIS), offset 0x104

The **MAC MMC Receive Interrupt (EMACMMCRXRIS)** register maintains the interrupts that are generated when the following happens:

- Receive statistic counters reach half of their maximum values (0x8000.0000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFF.FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the Counter Stop Rollover (CNTSTPRO bit) in the **MAC MMC Control (EMACMMCCTRL)** register is set, interrupts are set but the counter remains at all-ones. The **EMACMMCRXRIS** register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Ethernet MAC MMC Receive Raw Interrupt Status (EMACMMCRXRIS)

Base 0x400E.C000
Offset 0x104
Type RO, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | UCGF | reserved |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | ALGNERR | CRCERR | reserved | | | | GBF |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0x00 | 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. |
| 17 | UCGF | RO | 0x0 | MMC Receive Unicast Good Frame Counter Interrupt Status<br>Value Description<br>0 The receive **Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI)** register has not reached half of the maximum value or the maximum value.<br>1 The receive **Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI)** register has reached half of the maximum value or the maximum value. |
| 16:7 | reserved | RO | 0 | 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. |

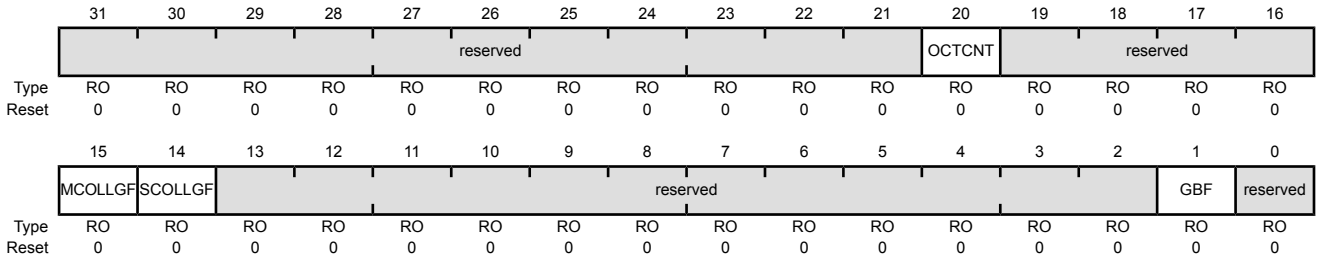
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | ALGNERR | RO | 0x0 | MMC Receive Alignment Error Frame Counter Interrupt Status<br>Value Description<br>0 The **Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR)** register has not reached half of the maximum value or the maximum value.<br>1 The **Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR)** register has reached half of the maximum value or the maximum value. |
| 5 | CRCERR | RO | 0x0 | MMC Receive CRC Error Frame Counter Interrupt Status<br>Value Description<br>0 The **Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR)** register has not reached half of the maximum value or the maximum value.<br>1 The **Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR)** register has reached half of the maximum value or the maximum value. |
| 4:1 | reserved | RO | 0 | 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. |
| 0 | GBF | RO | 0x0 | MMC Receive Good Bad Frame Counter Interrupt Status<br>Value Description<br>0 The **Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB)** register has not reached half of the maximum value or the maximum value.<br>1 The **Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB)** register has reached half of the maximum value or the maximum value. |

## Register 25: Ethernet MAC MMC Transmit Raw Interrupt Status (EMACMMCTXRIS), offset 0x108

The **MAC MMC Transmit Interrupt (EMACMMCTXRIS)** register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and the maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the CNTSTPRO bit is set in the **MAC MMC Control (EMACMMCCTRL)** register, interrupts are set but the counter remains at all-ones. The **EMACMMCTXRIS** register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Ethernet MAC MMC Transmit Raw Interrupt Status (EMACMMCTXRIS)

Base 0x400E.C000
Offset 0x108
Type R, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:21 | reserved | RO | 0x0 | 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. |
| 20 | OCTCNT | RO | 0x0 | Octet Counter Interrupt Status<br>Value Description<br>0 The **Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG)** register has not reached half of the maximum value or the maximum value.<br>1 The **Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG)** register has reached half of the maximum value or the maximum value. |
| 19:16 | reserved | RO | 0 | 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. |

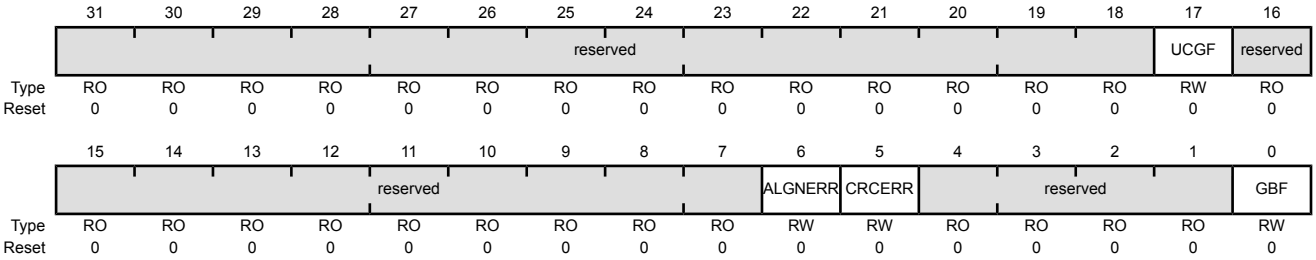
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | MCOLLGF | RO | 0x0 | MMC Transmit Multiple Collision Good Frame Counter Interrupt Status<br>Value Description<br>0 The **Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL)** register has not reached half of the maximum value or the maximum value.<br>1 The **Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL)** register has reached half of the maximum value or the maximum value. |
| 14 | SCOLLGF | RO | 0x0 | MMC Transmit Single Collision Good Frame Counter Interrupt Status<br>Value Description<br>0 The **Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL)** register has not reached half of the maximum value or the maximum value.<br>1 The **Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL)** register has reached half of the maximum value or the maximum value. |
| 13:2 | reserved | RO | 0 | 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. |
| 1 | GBF | RO | 0x0 | MMC Transmit Good Bad Frame Counter Interrupt Status<br>Value Description<br>0 The **Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB)** register has not reached half of the maximum value or the maximum value.<br>1 The **Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB)** register has reached half of the maximum value or the maximum value. |
| 0 | reserved | RO | 0 | 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. |

## Register 26: Ethernet MAC MMC Receive Interrupt Mask (EMACMMCRXIM), offset 0x10C

The **MAC MMC Receive Interrupt Mask (EMACMMCRXIM)** register maintains the masks for the interrupts generated when the receive statistic counters reach half of their maximum value, or maximum value. This register is 32-bits wide.

Ethernet MAC MMC Receive Interrupt Mask (EMACMMCRXIM)

Base 0x400E.C000
Offset 0x10C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:18 | reserved | RO | 0x0 | 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. |
| 17 | UCGF | RW | 0x0 | MMC Receive Unicast Good Frame Counter Interrupt Mask<br>Value Description<br>0 An interrupt is sent to the interrupt controller when the UCGF bit in the **EMACMMCRXRIS** register is set.<br>1 The UCGF interrupt is suppressed and not sent to the interrupt controller. |
| 16:7 | reserved | RO | 0 | 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. |
| 6 | ALGNERR | RW | 0x0 | MMC Receive Alignment Error Frame Counter Interrupt Mask<br>Value Description<br>0 An interrupt is sent to the interrupt controller when the ALGNERR bit in the **EMACMMCRXRIS** register is set.<br>1 The ALGNERR interrupt is suppressed and not sent to the interrupt controller. |
| 5 | CRCERR | RW | 0x0 | MMC Receive CRC Error Frame Counter Interrupt Mask<br>Value Description<br>0 An interrupt is sent to the interrupt controller when the CRCERR bit in the **EMACMMCRXRIS** register is set.<br>1 The CRCERR interrupt is suppressed and not sent to the interrupt controller. |

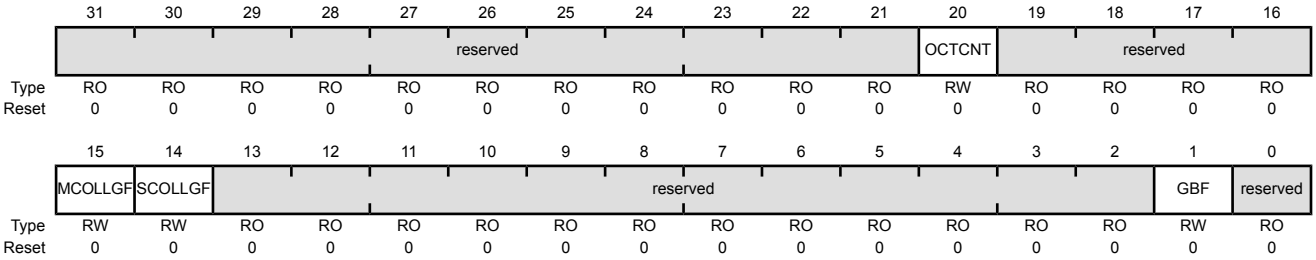
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | GBF | RW | 0x0 | MMC Receive Good Bad Frame Counter Interrupt Mask<br><br>Value Description<br>0 An interrupt is sent to the interrupt controller when the GBF bit in the **EMACMMCRXRIS** register is set.<br>1 The GBF interrupt is suppressed and not sent to the interrupt controller. |

## Register 27: Ethernet MAC MMC Transmit Interrupt Mask (EMACMMCTXIM), offset 0x110

The **MAC MMC Transmit Interrupt Mask (EMACMMCTXIM)** register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.

Ethernet MAC MMC Transmit Interrupt Mask (EMACMMCTXIM)

Base 0x400E.C000
Offset 0x110
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:21 | reserved | RO | 0x0 | 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. |
| 20 | OCTCNT | RW | 0x0 | MMC Transmit Good Octet Counter Interrupt Mask<br>Value Description<br>0 An interrupt is sent to the interrupt controller when the OCTCNT bit in the **EMACMMCTXRIS** register is set.<br>1 The OCTCNT interrupt is suppressed and not sent to the interrupt controller. |
| 19:16 | reserved | RO | 0 | 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. |
| 15 | MCOLLGF | RW | 0x0 | MMC Transmit Multiple Collision Good Frame Counter Interrupt Mask<br>Value Description<br>0 An interrupt is sent to the interrupt controller when the MCOLLGF bit in the **EMACMMCTXRIS** register is set.<br>1 The MCOLLGF interrupt is suppressed and not sent to the interrupt controller. |
| 14 | SCOLLGF | RW | 0x0 | MMC Transmit Single Collision Good Frame Counter Interrupt Mask<br>Value Description<br>0 An interrupt is sent to the interrupt controller when the SCOLLGF bit in the **EMACMMCTXRIS** register is set.<br>1 The SCOLLGF interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13:2 | reserved | RO | 0 | 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. |
| 1 | GBF | RW | 0x0 | MMC Transmit Good Bad Frame Counter Interrupt Mask<br><br>Value Description<br>0 An interrupt is sent to the interrupt controller when the GBF bit in the **EMACMMCTXRIS** register is set.<br>1 The GBF interrupt is suppressed and not sent to the interrupt controller. |
| 0 | reserved | RO | 0 | 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. |

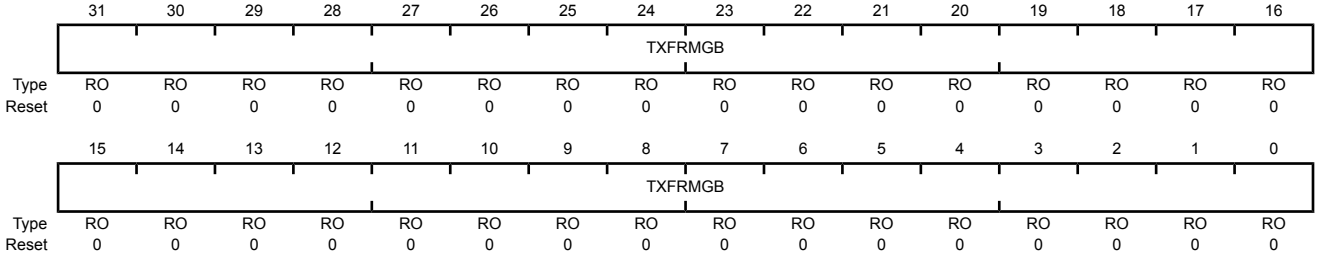
## Register 28: Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB), offset 0x118

The **MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB)** register maintains the number of good and bad frames transmitted, exclusive of retried frames.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Transmit Frame Count for Good and Bad Frames (EMACTXCNTGB)

Base 0x400E.C000
Offset 0x118
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TXFRMGB | RO | 0x0 | This field indicates the number of good and bad frames transmitted, exclusive of retried frames |

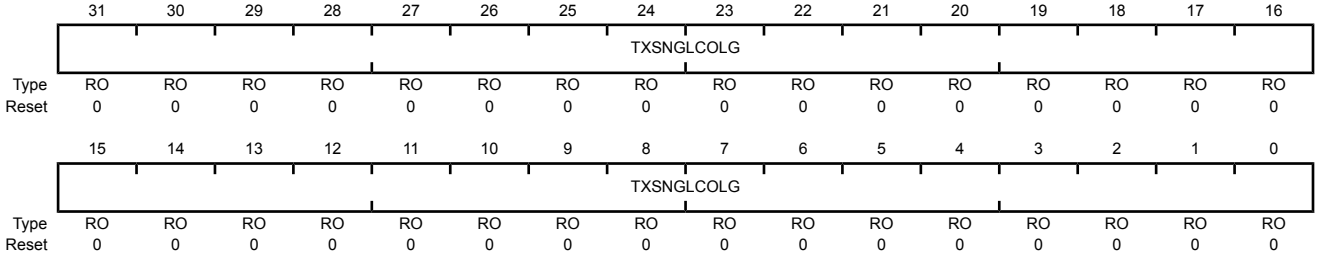
## Register 29: Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL), offset 0x14C

This register maintains the number of successfully transmitted frames after a single collision in the half-duplex mode.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Transmit Frame Count for Frames Transmitted after Single Collision (EMACTXCNTSCOL)

Base 0x400E.C000
Offset 0x14C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TXSNGLCOLG | RO | 0x0 | This field indicates the number of successfully transmitted frames after a single collision in the half-duplex mode. |

## Register 30: Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL), offset 0x150

This register maintains the number of successfully transmitted frames after multiple collisions in the half-duplex mode.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Transmit Frame Count for Frames Transmitted after Multiple Collisions (EMACTXCNTMCOL)

Base 0x400E.C000
Offset 0x150
Type RO, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXMULTCOLG | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXMULTCOLG | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TXMULTCOLG | RO | 0x0 | This field indicates the number of successfully transmitted frames after multiple collisions in the half-duplex mode. |

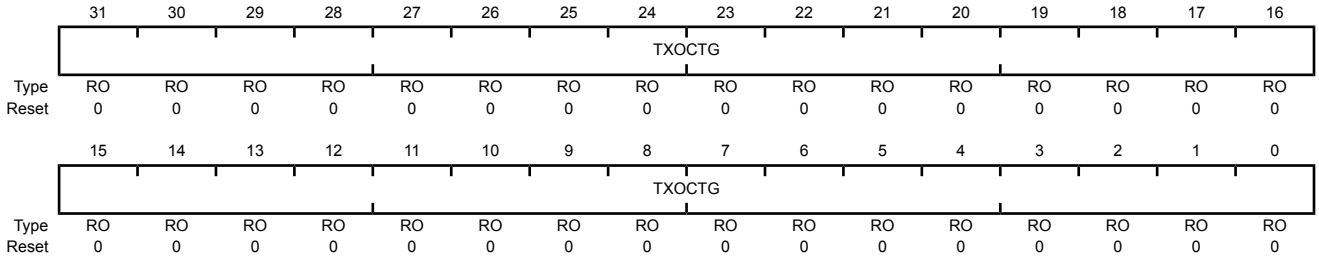
## Register 31: Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG), offset 0x164

This register maintains the number of bytes transmitted, exclusive of preamble, only in good frames.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Transmit Octet Count Good (EMACTXOCTCNTG)

Base 0x400E.C000
Offset 0x164
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TXOCTG | RO | 0x0 | This field indicates the number of bytes transmitted, exclusive of preamble, in good frames. |

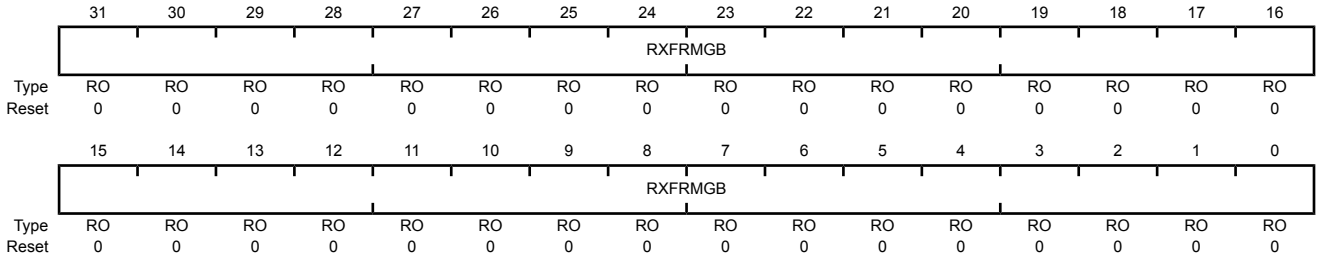
## Register 32: Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB), offset 0x180

This register maintains the number of received good and bad frames.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Receive Frame Count for Good and Bad Frames (EMACRXCNTGB)

Base 0x400E.C000
Offset 0x180
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RXFRMGB | RO | 0x0 | This field indicates the number of received good and bad frames. |

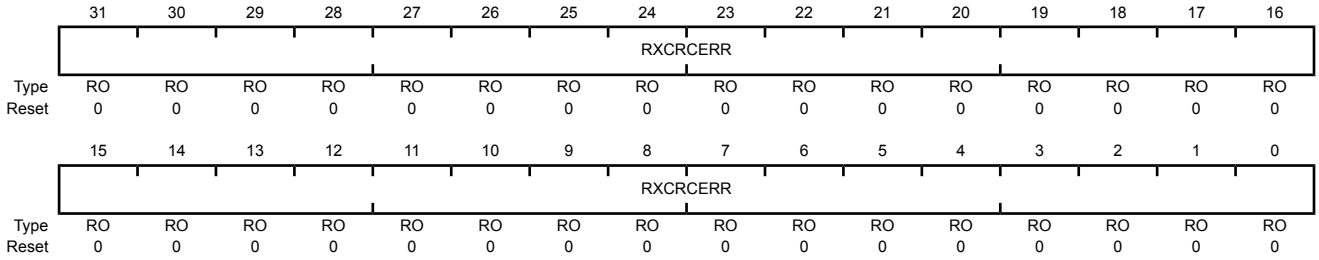
## Register 33: Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR), offset 0x194

This register maintains the number of frames received with CRC error.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Receive Frame Count for CRC Error Frames (EMACRXCNTCRCERR)

Base 0x400E.C000
Offset 0x194
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RXCRCERR | RO | 0x0 | This field indicates the number of frames received with CRC error. |

## Register 34: Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR), offset 0x198

This register maintains the number of frames received with alignment (dribble) error.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Receive Frame Count for Alignment Error Frames (EMACRXCNTALGNERR)

Base 0x400E.C000
Offset 0x198
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXALGNERR | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXALGNERR | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RXALGNERR | RO | 0x0 | This field indicates the number of frames received with alignment (dribble) error. |

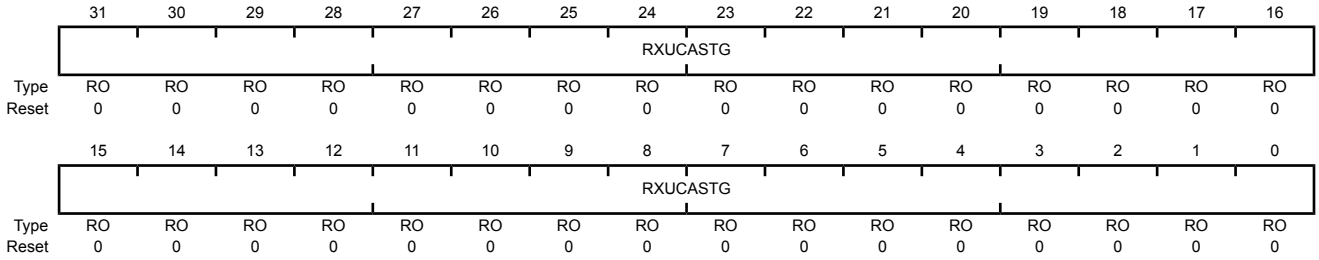
## Register 35: Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI), offset 0x1C4

This register maintains the number of received good unicast frames.

**Note:** This counter is reset to all zeros by setting the CNTRST bit in the **Ethernet MAC MMC Control (EMACMMCCTRL)**, offset 0x100.

Ethernet MAC Receive Frame Count for Good Unicast Frames (EMACRXCNTGUNI)

Base 0x400E.C000
Offset 0x1C4
Type RO, reset 0x0000.0000



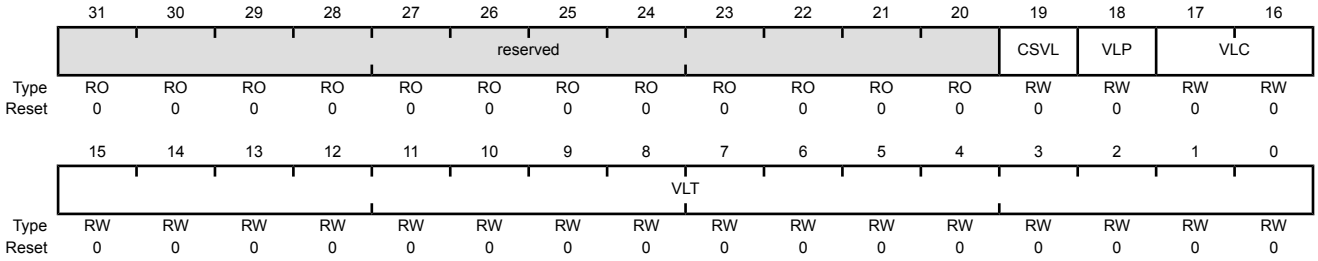

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RXUCASTG | RO | 0x0 | This field indicates the number of received good unicast frames. |

## Register 36: Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP), offset 0x584

The **MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP)** register contains the VLAN tag for insertion or replacement in the transmit frames.

Ethernet MAC VLAN Tag Inclusion or Replacement (EMACVLNINCREP)

Base 0x400E.C000
Offset 0x584
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x0 | 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. |
| 19 | CSVL | RW | 0x0 | C-VLAN or S-VLAN<br>Value Description<br>0 C-VLAN type (0x8100) is inserted or replaced in the transmitted frames.<br>1 S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted frames. |
| 18 | VLP | RW | 0x0 | VLAN Priority Control<br>Value Description<br>0 The control input from the transmit descriptor is used, and the VLC bit field (bits [17:16]) is ignored.<br>1 The VLC bit field is used for VLAN deletion, insertion, or replacement. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17:16 | VLC | RW | 0x0 | VLAN Tag Control in Transmit Frames<br><br>Value Description<br>0x0 No VLAN tag deletion, insertion, or replacement<br>0x1 VLAN tag deletion<br>The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted frames with VLAN tags.<br>0x2 VLAN tag insertion<br>The MAC inserts VLT in bytes 15 and 16 of the frame after inserting the Type value (0x8100/0x88a8) in bytes 13 and 14. This operation is performed on all transmitted frames, irrespective of whether they already have a VLAN tag.<br>0x3 VLAN tag replacement<br>The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted frames (Bytes 13 and 14 are 0x8100/0x88a8).<br><br>**Note:** Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value. |
| 15:0 | VLT | RW | 0x0 | VLAN Tag for Transmit Frames<br>This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority, Bit 12 is the CFI/DEI, and Bits[11:0] are the VLAN tag's VID field. |

## Register 37: Ethernet MAC VLAN Hash Table (EMACVLANHASH), offset 0x588

The 16-bit hash table is used for group address filtering based on VLAN tag when the VTHM bit of the **EMACVLANTG** register is set. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (enabled by ETV in the **EMACVLANTG** register) in the incoming frame is passed through the CRC logic and the upper four bits of the calculated CRC are used to index the contents of the **EMACVLANHASH** register. For example, a hash value of 0x8 selects bit 8 of the VLAN Hash table. The hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).

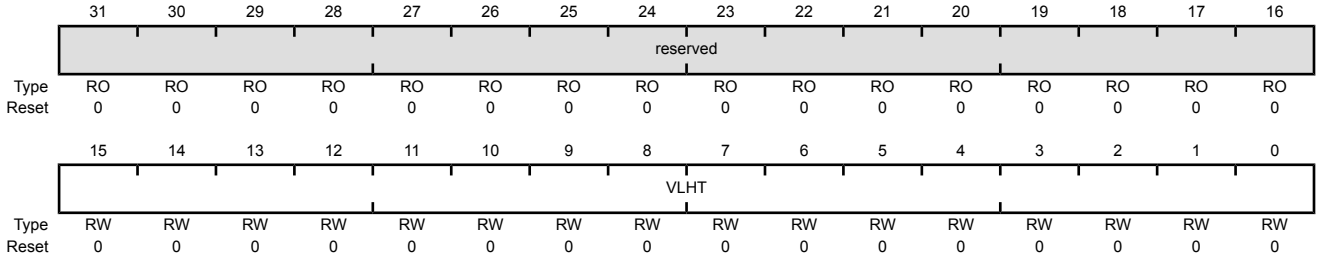
2. Perform bitwise reversal for the value obtained in Step 1.

3. Take the upper four bits from the value obtained in Step 2.

If the corresponding bit value of the register is 0x1, the frame is accepted. Otherwise, it is rejected.

Ethernet MAC VLAN Hash Table (EMACVLANHASH)

Base 0x400E.C000
Offset 0x588
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | 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. |
| 15:0 | VLHT | RW | 0x0 | VLAN Hash Table<br>This field contains the 16-bit VLAN Hash Table. |

## Register 38: Ethernet MAC Timestamp Control (EMACTIMSTCTRL), offset 0x700

This register controls the operation of the system time generator and the processing of Precision Time Protocol (PTP) packets for time-stamping in the receiver.

Ethernet MAC Timestamp Control (EMACTIMSTCTRL)

Base 0x400E.C000
Offset 0x700
Type RW, reset 0x0000.2000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | PTPFLTR | SELPTP | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSMAST | TSEVNT | PTPIPV4 | PTPIPV6 | PTPETH | PTPVER2 | DGTLBIN | ALLF | reserved | | ADDREGUP | INTTRIG | TSUPDT | TSINIT | TSFCUPDT | TSEN |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RO | RO | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x0 | 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. |
| 18 | PTPFLTR | RW | 0x0 | Enable MAC address for PTP Frame Filtering<br>Value Description<br>0 No effect.<br>1 The Destination Address (DA) MAC address, that matches any MAC Address register is used to filter PTP frames when PTP is directly sent over Ethernet. |
| 17:16 | SELPTP | RW | 0x0 | Select PTP packets for Taking Snapshots<br>These bits along with Bits 15 and 14 decide the set of PTP packet types for which a snapshot needs to be taken. |
| 15 | TSMAST | RW | 0x0 | Enable Snapshot for Messages Relevant to Master<br>Value Description<br>0 The snapshot is taken for the messages relevant to the slave node.<br>1 The snapshot is taken only for the messages relevant to the master node. |
| 14 | TSEVNT | RW | 0x0 | Enable Timestamp Snapshot for Event Messages<br>Value Description<br>0 The snapshot is taken for all messages except Announce, Management, and Signaling.<br>1 The timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | PTPIPV4 | RW | 0x1 | Enable Processing of PTP Frames Sent over IPv4-UDP<br>Value Description<br>0 The MAC ignores PTP transported over UDP-IPv4 packets. This bit is set by default.<br>1 The MAC receiver processes PTP packets encapsulated in UDP over IPv4 packets. |
| 12 | PTPIPV6 | RW | 0x0 | Enable Processing of PTP Frames Sent Over IPv6-UDP<br>Value Description<br>0 The MAC ignores PTP transported over UDP-IPv6 packets.<br>1 The MAC receiver processes PTP packets encapsulated in UDP over IPv6 packets. |
| 11 | PTPETH | RW | 0x0 | Enable Processing of PTP Over Ethernet Frames<br>Value Description<br>0 The MAC ignores PTP over Ethernet packets.<br>1 The MAC receiver processes PTP packets encapsulated directly in Ethernet frames. |
| 10 | PTPVER2 | RW | 0x0 | Enable PTP Packet Processing For Version 2 Format<br>Value Description<br>0 PTP packets are processed using the IEEE 1588 version 1 format.<br>1 PTP packets are processed using the IEEE 1588 version 2 format. |
| 9 | DGTLBIN | RW | 0x0 | Timestamp Digital or Binary Rollover Control<br>Value Description<br>0 The TTSL0 field of the **EMACTARGNANO** register, offset 0x720, rolls over after 0x7FFF_FFFF and increments the **EMACTARGSEC** register. The sub-second increment has to be programmed correctly depending on the MOSC frequency and the value of this bit.<br>1 The **EMACTARGNANO** register rolls over after 0x3B9A.C9FF value (that is, 1 nanosecond accuracy) and increments the **EMACTARGSEC** register. |
| 8 | ALLF | RW | 0x0 | Enable Timestamp For All Frames<br>Value Description<br>0 No effect.<br>1 Timestamp snapshots are enabled for all frames received by the MAC. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | reserved | RO | 0x0 | 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 | ADDREGUP | RW | 0x0 | Addend Register Update<br>Value Description<br>0 No effect.<br>1 When set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This is cleared when the update is completed. This register bit should be zero before setting it. |
| 4 | INTTRIG | RW | 0x0 | Timestamp Interrupt Trigger Enable<br>Value Description<br>0 No effect.<br>1 The timestamp interrupt is generated when the System Time becomes greater than the value written in the **Ethernet MAC Target Time Seconds/Nanoseconds (EMACTARGSEC/EMACTARGNANO)** registers. This bit is reset after the generation of the Timestamp Trigger Interrupt. |
| 3 | TSUPDT | RW | 0x0 | Timestamp Update<br>This bit should be read zero before updating it. This bit is reset when the update is completed in hardware. The Timestamp Higher Word register is not updated.<br>Value Description<br>0 No effect.<br>1 When set, the system time is updated (added or subtracted) with the value specified in **EMACTIMSECU** (System Time - Seconds Update Register) and **EMACTIMNANOU** (System Time - Nanoseconds Update Register). |
| 2 | TSINIT | RW | 0x0 | Timestamp Initialize<br>This bit should read zero before updating it.<br>Value Description<br>0x0 This bit is reset when the initialization is complete.<br>0x1 The system time is initialized (overwritten) with the value specified in the **Ethernet MAC System Time-Seconds Update (EMACTIMSECU)** and the **Ethernet MAC System Time-Nanoseconds Update (EMACTIMNANOU)** registers. |

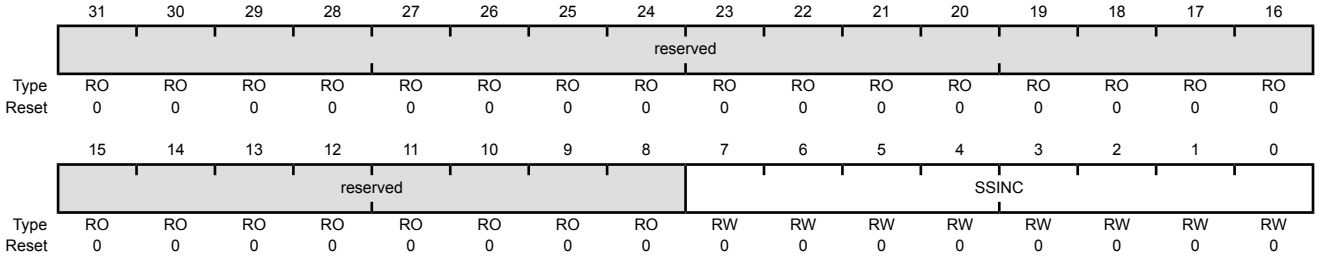
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | TSFCUPDT | RW | 0x0 | Timestamp Fine or Coarse Update<br><br>Value Description<br>0x0 Indicates the system timestamp update should be done using the coarse method.<br>0x1 Indicates that the system times update should be done using the fine update method. |
| 0 | TSEN | RW | 0x0 | Timestamp Enable<br><br>The **EMACTIMSEC** and the **EMACTIMNANO** registers must be initialized after enabling this mode. On the receive side, the MAC processes 1588 frames only if this bit is set.<br><br>Value Description<br>0x0 The timestamp is not added for the transmit and receive frames and the timestamp generator module is also suspended.<br>0x1 The timestamp is added for the transmit and receive frames |

## Register 39: Ethernet MAC Sub-Second Increment (EMACSUBSECINC), offset 0x704

In the Coarse Update mode (enabled by the TSCFUPDT bit in the **MAC Timestamp Control (EMACTIMSTCTRL)** register), the value in the **EMACSUBSECINC** register is added to the system time every clock cycle of slave clock reference, MOSC. In the Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

Ethernet MAC Sub-Second Increment (EMACSUBSECINC)

Base 0x400E.C000
Offset 0x704
Type RW, reset 0x0000.0000



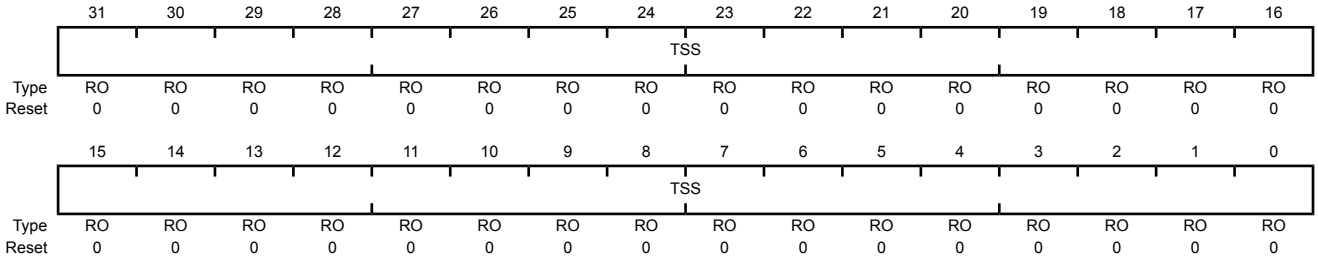

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | SSINC | RW | 0x0 | Sub-second Increment Value<br>The value programmed in this field is accumulated every clock cycle (MOSC) with the contents of the sub-second register. For example, when MOSC is 25 MHz (period is 40 ns), SSINC should be programmed with the value 40 (0x28) when the **Ethernet MAC System Time - Nanoseconds (EMACTIMNANO)** register has an accuracy of 1 ns (DGTLBIN bit is set in **EMACTIMSTCTRL**). When DGTLBIN bit is clear, the **EMACTIMNANO** register has a resolution of ~0.465ns. In this case, a value of 86 (0x56), that is derived by 40ns/0.465, should be programmed in the SSINC field. |

## Register 40: Ethernet MAC System Time - Seconds (EMACTIMSEC), offset 0x708

The **MAC System Time - Seconds (EMACTIMSEC)** register, along with the **MAC System Time - Nanoseconds (EMACTIMNANO)** register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies.

Ethernet MAC System Time - Seconds (EMACTIMSEC)

Base 0x400E.C000
Offset 0x708
Type RO, reset 0x0000.0000



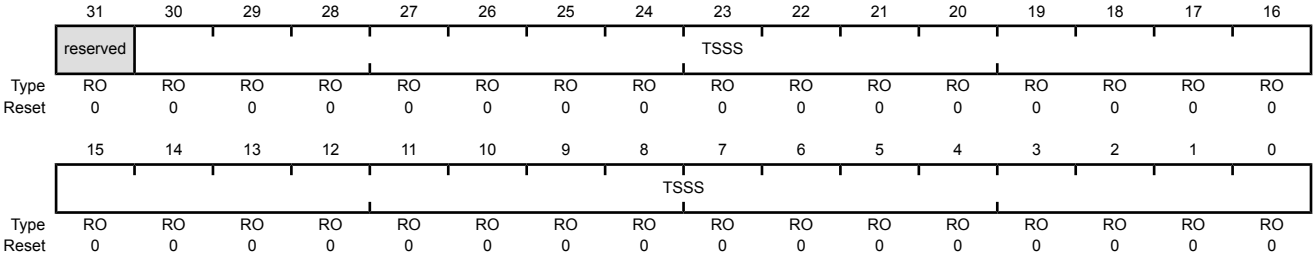

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TSS | RO | 0x0 | Timestamp Second<br>The value in this field indicates the current value in seconds of the system time maintained by the MAC. |

## Register 41: Ethernet MAC System Time - Nanoseconds (EMACTIMNANO), offset 0x70C

The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When DGTLBIN in the **EMACTIMSTCTRL** register is set, each bit represents 1 ns and the maximum value is 0x3B9A.C9FF, after which it rolls-over to zero.

Ethernet MAC System Time - Nanoseconds (EMACTIMNANO)

Base 0x400E.C000
Offset 0x70C
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | reserved | RO | 0x0 | 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. |
| 30:0 | TSSS | RO | 0x0 | Timestamp Sub-Seconds<br>The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When DGTLBIN in the **EMACTIMSTCTRL** register is set, each bit represents 1 ns and the maximum value is 0x3B9A.C9FF, after which it rolls-over to zero. |

## Register 42: Ethernet MAC System Time - Seconds Update (EMACTIMSECU), offset 0x710

The **MAC System Time - Seconds Update (EMACTIMSECU)** register, along with the **MAC System Time - Nanoseconds Update (EMACTIMNANOU)** register, initializes or updates the system time maintained by the MAC. Both of these register must be written before setting the TSINIT or TSUPDT bits in the **EMACTIMSTCTRL** register.

Ethernet MAC System Time - Seconds Update (EMACTIMSECU)

Base 0x400E.C000
Offset 0x710
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSS | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TSS | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

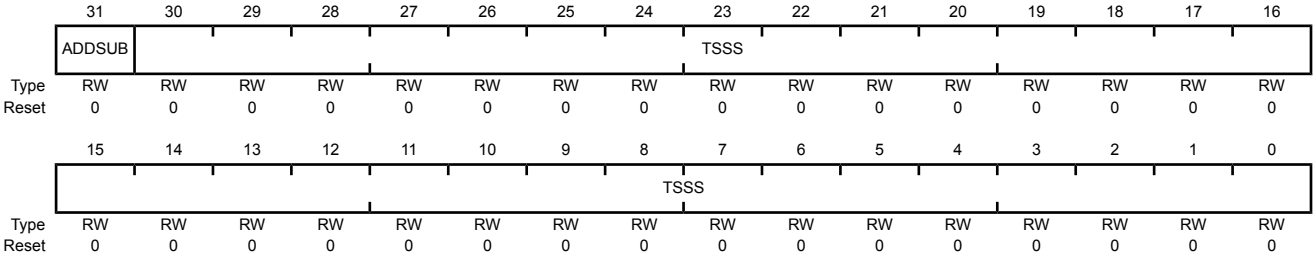
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TSS | RW | 0x0 | Timestamp Second<br>The value in this field indicates the time in seconds to be initialized or added to the system time. |

## Register 43: Ethernet MAC System Time - Nanoseconds Update (EMACTIMNANOU), offset 0x714

This register along with the **EMACTIMSECU** register initializes or updates the system time maintained by the MAC. Both of these register must be written before setting the TSINIT or TSUPDT bits in the **EMACTIMSTCTRL** register.

Ethernet MAC System Time - Nanoseconds Update (EMACTIMNANOU)

Base 0x400E.C000
Offset 0x714
Type RW, reset 0x0000.0000



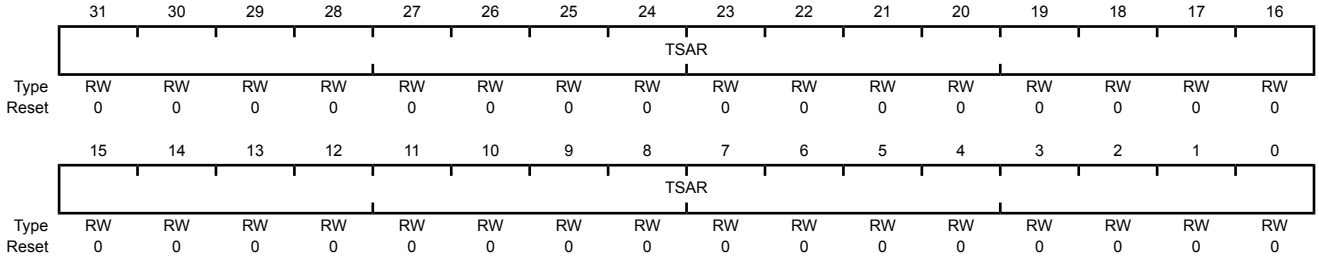

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ADDSUB | RW | 0x0 | Add or subtract time<br><br>Value Description<br>0x0 The time value is added with the contents of the update register.<br>0x1 The time value is subtracted with the contents of the update register. |
| 30:0 | TSSS | RW | 0x0 | Timestamp Sub-Second<br><br>The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When DGTLBIN in the **EMACTIMSTCTRL** register is set, each bit represents 1 ns and the programmed value should not exceed 0x3B9A.C9FF. |

## Register 44: Ethernet MAC Timestamp Addend (EMACTIMADD), offset 0x718

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit is set in the **EMACTIMSTCTRL** register). This register content is added to a 32-bit accumulator every slave clock cycle (MOSC source) and the system time is updated whenever the accumulator overflows.

Ethernet MAC Timestamp Addend (EMACTIMADD)

Base 0x400E.C000
Offset 0x718
Type RW, reset 0x0000.0000



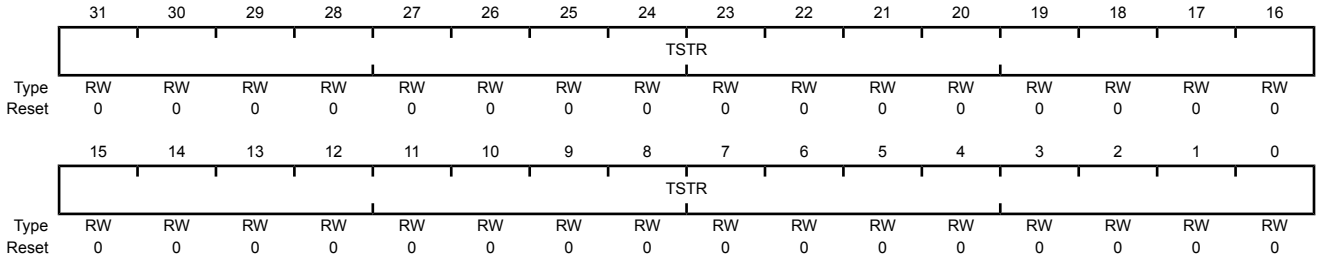

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TSAR | RW | 0x0 | Timestamp Addend Register<br>This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization. |

## Register 45: Ethernet MAC Target Time Seconds (EMACTARGSEC), offset 0x71C

The **MAC Target Time Seconds (EMACTARGSEC)** register, along with the **MAC Target Time Nanoseconds (EMACTARGNANO)** register, is used to schedule an interrupt event.

Ethernet MAC Target Time Seconds (EMACTARGSEC)

Base 0x400E.C000
Offset 0x71C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TSTR | RW | 0x0 | Target Time Seconds Register<br>This register stores the time in seconds. When the timestamp value matches or exceeds both the **MAC Target Time Seconds (EMACTARGSEC)** and **MAC Target Time Nanoseconds (EMACTARGNANO)** registers, then based on the TRGMODS0 bit field in the **MAC PPS Control (EMACPPSCTRL)**, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled). |

## Register 46: Ethernet MAC Target Time Nanoseconds (EMACTARGNANO), offset 0x720

The **MAC Target Time Seconds (EMACTARGSEC)** register, along with the **MAC Target Time Nanoseconds (EMACTARGNANO)** register, is used to schedule an interrupt event.

Ethernet MAC Target Time Nanoseconds (EMACTARGNANO)

Base 0x400E.C000
Offset 0x720
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TRGTBUSY | | | | | | | | TTSLO | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TTSLO | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

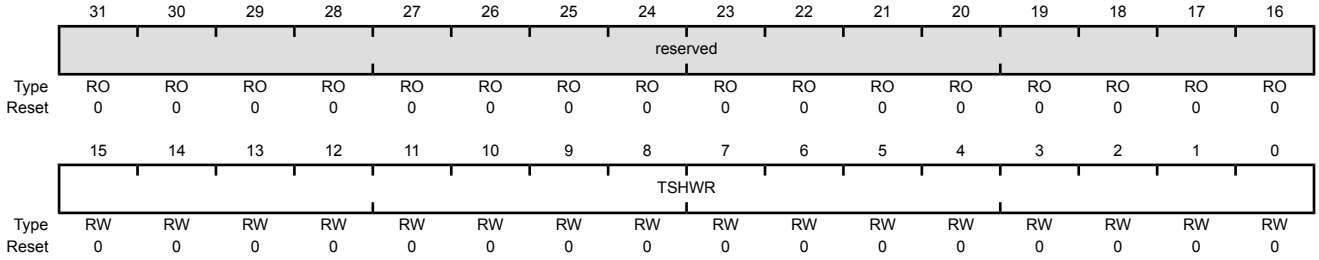
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | TRGTBUSY | RW | 0x0 | Target Time Register Busy<br>This bit is set and cleared by the MAC.<br>Value Description<br>0 The **Ethernet MAC Target Time Seconds/Nanoseconds (EMACTARGSEC/EMACTARGNANO)** registers are not busy.<br>1 The **Ethernet MAC Target Time Seconds/Nanoseconds (EMACTARGSEC/EMACTARGNANO)** registers are busy. This bit is set when the PPSCTRL field in the **EMACPPSCTRL** register is programmed to 0x2 or 0x3 and the MAC is instructed to synchronize the **EMACTARGSEC/EMACTARGNANO** registers to the PTP clock domain.<br>**Note:** The **EMACTARGSEC** and **EMACTARGNANO** registers must not be updated when this bit is read as 1. |
| 30:0 | TTSLO | RW | 0x0 | Target Timestamp Low Register<br>This register stores the time in (signed) nanoseconds. When the value of the timestamp matches both **EMACTARGx** registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGMODS0 field in the **MAC PPS Control (EMACPPSCTRL)** register.<br>This value should not exceed 0x3B9A.C9FF when DGTLBIN is set in the **EMACTIMSTCTRL** register. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value. |

## Register 47: Ethernet MAC System Time-Higher Word Seconds (EMACHWORDSEC), offset 0x724

This register is used to support the most significant 16-bits of the timestamp seconds value.

Ethernet MAC System Time-Higher Word Seconds (EMACHWORDSEC)

Base 0x400E.C000
Offset 0x724
Type RW, reset 0x0000.0000

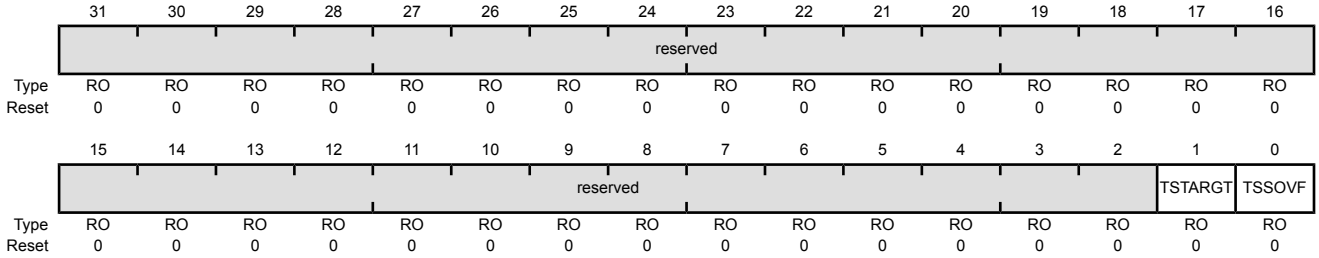



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TSHWR | RW | 0x0 | Target Timestamp Higher Word Register<br>This field contains the most significant 16-bits of the timestamp seconds value. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the System Time - Seconds register. |

## Register 48: Ethernet MAC Timestamp Status (EMACTIMSTAT), offset 0x728

Ethernet MAC Timestamp Status (EMACTIMSTAT)

Base 0x400E.C000
Offset 0x728
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0 | 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. |
| 1 | TSTARGT | RO | 0x0 | Timestamp Target Time Reached<br><br>Value Description<br>0x0 Target Time has not been reached.<br>0x1 Indicates that the value of system time is greater or equal to the value specified in **EMACTARGSEC** (Target Time Seconds Register) and **EMACTARGNANO** (Target Time Nanoseconds Register). |
| 0 | TSSOVF | RO | 0x0 | Timestamp Seconds Overflow<br><br>Value Description<br>0x0 No timestamp overflow has occurred.<br>0x1 Indicates the seconds value of the timestamp has overflowed beyond 0xFFFF.FFFF. |

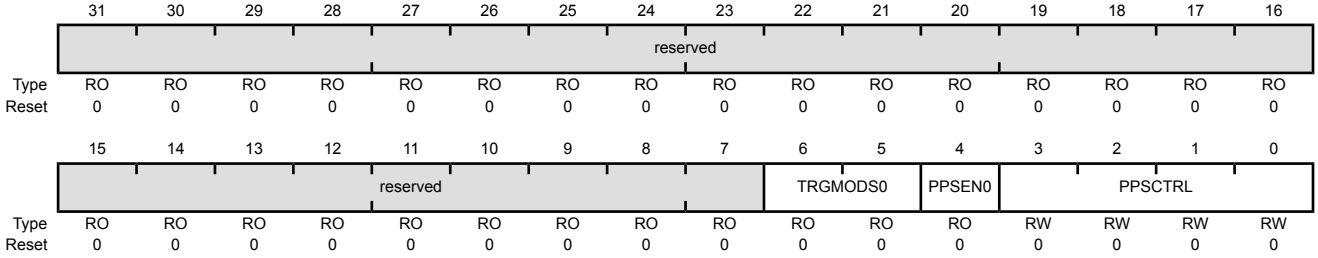
## Register 49: Ethernet MAC PPS Control (EMACPPSCTRL), offset 0x72C

This register is used to control the EN0PPS signal output.

**Note:** The PTP reference clock referred to below is MOSC clock in course update mode and in fine correction mode, is the clock tick at which the system time gets updated.

Ethernet MAC PPS Control (EMACPPSCTRL)

Base 0x400E.C000
Offset 0x72C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0x0 | 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. |
| 6:5 | TRGMODS0 | RO | 0x0 | Target Time Register Mode for PPS0 Output<br>This field indicates the Target Time registers (**EMACTARGSEC** and **EMACTARGNANO**) mode for the EN0PPS output signal:<br>Value Description<br>0x0 Indicates that the Target Time registers are programmed only for generating the interrupt event.<br>0x1 Reserved<br>0x2 Indicates that the Target Time registers are programmed for generating the interrupt event and starting or stopping the generation of the EN0PPS output signal.<br>0x3 Indicates that the Target Time registers are programmed only for starting or stopping the generation of the EN0PPS output signal. No interrupt is asserted. |
| 4 | PPSEN0 | RO | 0x0 | Flexible PPS Output Mode Enable<br>Value Description<br>0x0 Bits[3:0] function as PPS output frequency control (PPSCTRL).<br>0x1 Bits[3:0] function as PPS Command (PPSCMD). |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | PPSCTRL | RW | 0x0 | EN0PPS Output Frequency Control (PPSCTRL) or Command Control (PPSCMD)<br>This bit field has two different functions depending on how the PPSEN0 bit is set. See the values in Table 24-25 on page 1736.<br>If the PPSEN0 bit is set to 0x0, this field functions as a PPS0 output frequency control (PPSCTRL), which controls the frequency of the output signal, EN0PPS. The default value of this field is 0x0 and the PPS output is one pulse every second.<br>If the PPSEN0 bit is 0x1, this field functions as a flexible output command control for the EN0PPS signal. Programming the PPSCTRL bit field with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits are cleared automatically. Software should ensure that these bits are programmed only when they are "all-zeros."<br>Note that in the binary rollover mode, the EN0PPS signal has a duty cycle of 50 percent with these frequencies.<br>In the digital rollover mode, the EN0PPS signal frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:<br>When PPSCTRL =0x1, EN0PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.<br>When PPSCTRL = 0x2, EN0PPS (2 Hz) is a sequence of:<br>■ One clock of 50 percent duty cycle and 537 ms period<br>■ Second clock of 463 ms period (268 ms low and 195 ms high)<br>When PPSCTRL = 0x3, EN0PPS (4 Hz) is a sequence of:<br>■ Three clocks of 50 percent duty cycle and 268 ms period<br>■ Fourth clock of 195 ms period (134 ms low and 61 ms high)<br>This signaling behavior is because of the non-linear toggling of bits in the digital rollover mode in the **Ethernet MAC System Time - Nanoseconds (EMACTIMNANO)** register. |

### Table 24-25. PPSCTRL Bit Field Values

| Value | Description |
|---|---|
| 0x0 | When the PPSEN0 bit = 0x0, the EN0PPS signal is 1 pulse of the PTP reference clock. every second.<br>When the PPSEN0 bit = 0x1, this encoding indicates no command. |
| 0x1 | When the PPSEN0 bit = 0x0, the binary rollover is 2 Hz, and the digital rollover is 1 Hz.<br>When the PPSEN0 bit = 0x1, START single pulse. This command generates a single pulse rising at the start point defined in the **Ethernet MAC Target Time Second/Nanoseconds (EMACTARGX)** registers (MAC offsets 0x71C and 0x720) and a duration defined in the **Ethernet MAC PPS0 Width (EMACPPS0WIDTH)** register (offset 0x764). |
| 0x2 | When the PPSEN0 bit = 0x0, the binary rollover is 4 Hz, and the digital rollover is 2 Hz.<br>When the PPSEN0 bit = 0x1, START pulse train. This command generates the train of pulses rising at the start point defined in the **Ethernet MAC Target Time Second/Nanoseconds (EMACTARGX)** registers (MAC offsets 0x71C and 0x720) and repeated at an interval defined in the **Ethernet MAC PPS0 Width (EMACPPS0WIDTH)** register (offset 0x764). By default, the EN0PPS pulse train is free-running unless stopped by STOP pulse train at a time or STOP pulse train immediately command. |
| 0x3 | When thePPSEN0 bit = 0x0, the binary rollover is 8 Hz, and the digital rollover is 4 Hz, |

**Table 24-25. PPSCTRL Bit Field Values *(continued)***

| Value | Description |
|---|---|
| | When the PPSEN0 bit = 0x1, cancel START. This command cancels the START single pulse and START pulse train commands if the system time has not crossed the programmed start time. |
| 0x4 | When thePPSEN0 bit = 0x0, the binary rollover is 16 Hz, and the digital rollover is 8 Hz.<br>When the PPSEN0 bit = 0x1, STOP pulse train at time. This command stops the train of puses initiated by the START pulse train command after the time programmed in the **Ethernet MAC Target Time Second/Nanoseconds (EMACTARGX)** registers (MAC offsets 0x71C and 0x720). |
| 0x5 | When thePPSEN0 bit = 0x0, the binary rollover is 32 Hz, and the digital rollover is 16 Hz.<br>When the PPSEN0 bit = 0x1, STOP pulse train immediately. This command immediately stops the train of pulses initiated by the START pulse train command. |
| 0x6 | When thePPSEN0 bit = 0x0, the binary rollover is 64 Hz, and the digital rollover is 32 Hz.<br>When the PPSEN0 bit = 0x1, cancel STOP pulse train. This command cancels the STOP pulse train at the time command if the programmed stop time has not elapsed. The EN0PPS pulse train becomes free-running on the successful execution of this command. |
| 0x7 | When thePPSEN0 bit = 0x0, the binary rollover is 128 Hz, and the digital rollover is 64 Hz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0x8 | When thePPSEN0 bit = 0x0, the binary rollover is 256 Hz, and the digital rollover is 128 Hz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0x9 | When thePPSEN0 bit = 0x0, the binary rollover is 512 Hz, and the digital rollover is 256 Hz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0xA | When the PPSEN0 bit = 0x0, the binary rollover is 1.024 kHz, and the digital rollover is 512 Hz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0xB | When thePPSEN0 bit = 0x0, the binary rollover is 2.048 kHz, and the digital rollover is 1.024 kHz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0xC | When thePPSEN0 bit = 0x0, the binary rollover is 4.096 kHz, and the digital rollover is 2.048 kHz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0xD | When thePPSEN0 bit = 0x0, the binary rollover is 8.192 kHz, and the digital rollover is 4.096 kHz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0xE | When thePPSEN0 bit = 0x0, the binary rollover is 16.384 kHz, and the digital rollover is 8.092 kHz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |
| 0xF | When thePPSEN0 bit = 0x0, the binary rollover is 32.768 KHz, and the digital rollover is 16.384 KHz.<br>When the PPSEN0 bit = 0x1, this encoding is reserved. |

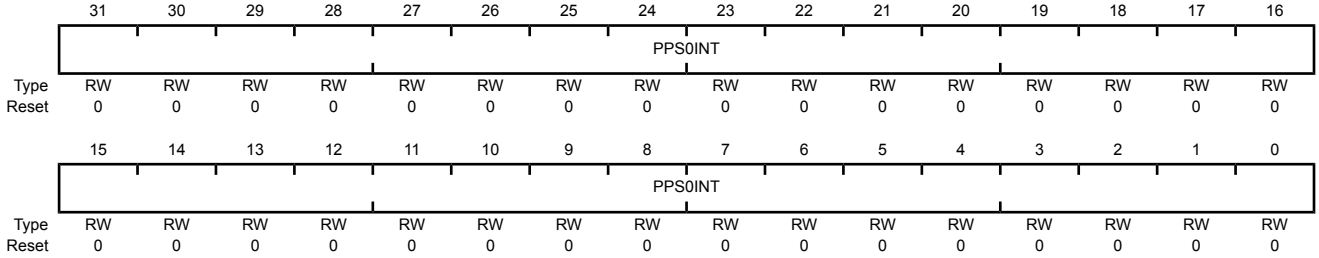
## Register 50: Ethernet MAC PPS0 Interval (EMACPPS0INTVL), offset 0x760

The **MAC PPS0 Interval (EMACPPS0INTVL)** register contains the number of units of sub-second increment value between the rising edges of EN0PPS signal output.

**Note:** The PTP reference clock referred to below is MOSC clock in course update mode and in fine correction mode, is the clock tick at which the system time gets updated.

Ethernet MAC PPS0 Interval (EMACPPS0INTVL)

Base 0x400E.C000
Offset 0x760
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PPS0INT | RW | 0x0 | PPS0 Output Signal Interval<br>These bits store the interval between the rising edges of the EN0PPS signal output in terms of units of sub-second increment value.<br>It must be programmed one value less than the required interval. For example, if the PTP reference clock is 25 MHz (period of 40 ns), and desired interval between rising edges of EN0PPS signal output is 120 ns (that is, three units of sub-second increment value), then you should program value 2 (3 -1) in this register. |

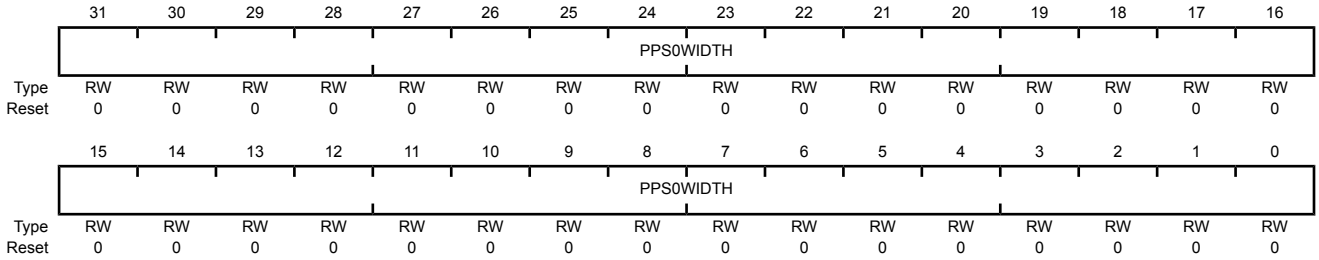
## Register 51: Ethernet MAC PPS0 Width (EMACPPS0WIDTH), offset 0x764

The **MAC PPS0 Width (EMACPPS0WIDTH)** register contains the number of units of sub-second increment value between the rising and corresponding falling edges of the EN0PPS output signal.

**Note:** The PTP reference clock referred to below is MOSC clock in course update mode and in fine correction mode, is the clock tick at which the system time gets updated.

Ethernet MAC PPS0 Width (EMACPPS0WIDTH)

Base 0x400E.C000
Offset 0x764
Type RW, reset 0x0000.0000



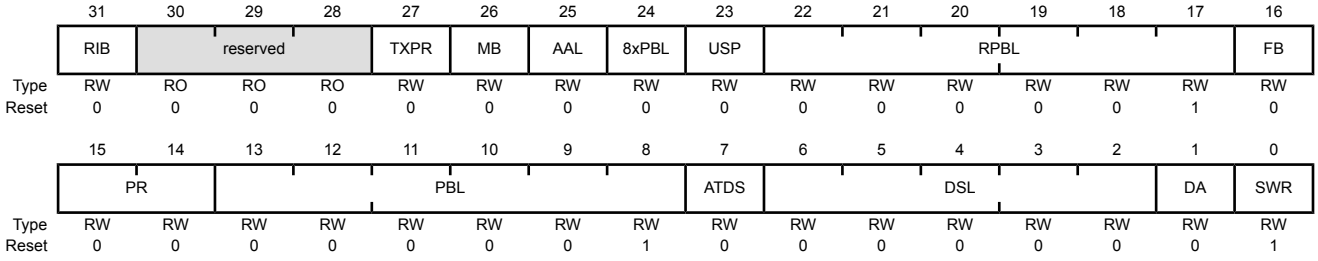

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PPS0WIDTH | RW | 0x0 | EN0PPS Output Signal Width<br>These bits store the width between the rising edges and corresponding falling edge of the of the EN0PPS signal output in terms of units of sub-second increment value.<br>It must be programmed one value less than the required interval. For example, if the PTP reference clock is 25 MHz (period of 40 ns), and the desired interval between rising edges of EN0PPS signal output is 80 ns (that is, two units of sub-second increment value), then you should program value 1 (2 -1) in this register.<br>**Note:** The value programmed in this register must be less than the value programmed in the **Ethernet MAC PPS0 Interval (EMACPPS0INTVL)** register, offset 0x760. |

## Register 52: Ethernet MAC DMA Bus Mode (EMACDMABUSMOD), offset 0xC00

The **Ethernet MAC DMA Bus Mode (EMACDMABUSMODE)** register establishes the operation modes for the DMA.

Ethernet MAC DMA Bus Mode (EMACDMABUSMOD)

Base 0x400E.C000
Offset 0xC00
Type RW, reset 0x0002.0101




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RIB | RW | 0x0 | Rebuild Burst<br><br>Value Description<br>0x0 During a retry, split or loss of bus, the DMA rebuilds the pending beats of any burst transfer with a continuous, uninterrupted burst until the last word, which is a single burst.<br>0x1 During a retry, split or loss of bus, the DMA rebuilds the pending beats of any burst transfer initiated with a defined fixed burst of 1, 4, 8, or 16. |
| 30:28 | reserved | RO | 0x0 | 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. |
| 27 | TXPR | RW | 0 | Transmit Priority<br><br>Value Description<br>0x0 The RX DMA has higher priority than the TX DMA during arbitration for the system bus.<br>0x1 The TX DMA has higher priority than the RX DMA during arbitration for the system bus. |
| 26 | MB | RW | 0 | Mixed Burst<br><br>Value Description<br>0x0 Mixed burst is not enabled.<br>0x1 If the FB bit is 0, the DMA starts all bursts of length more than 16 with a continuous undefined burst.<br>For bursts less than 16, fixed and single bursts are used. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25 | AAL | RW | 0 | Address Aligned Beats<br><br>Value Description<br>0x0 Address aligned transfers are not enabled.<br>0x1 If the FB bit is set, the internal bus interface generates all bursts aligned to the start address least significant bits.<br>If the FB bit is 0, the first burst is not aligned but subsequent bursts are aligned to the address. |
| 24 | 8xPBL | RW | 0 | 8 x Programmable Burst Length (PBL) Mode<br><br>Value Description<br>0x0 8 x PBL mode is inactive.<br>0x1 Bit field RPBL and bit field PBL are multiplied 8 times. Therefore, the DMA transfers the data in bursts of 8, 16, 32, 64, 128, and 256 words. |
| 23 | USP | RW | 0 | Use Separate Programmable Burst Length (PBL)<br><br>Value Description<br>0x0 The PBL value in bits[13:8] is applicable for both the RX and TX DMA engines.<br>0x1 RX DMA is uses the RPBL bit field as its defined programmable burst length and TX DMA uses the PBL bit field as its defined programmable burst length. |
| 22:17 | RPBL | RW | 0x1 | RX DMA Programmable Burst Length (PBL)<br>When the USP bit is 1, this field is used to indicate the maximum number of words to be transferred in one RX DMA transaction. This is the maximum value that is used in a single block read or write.<br>The RX DMA always attempts to burst as specified in the RPBL bit each time it starts a burst transfer on the system bus. The application can program RPBL with values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. This field is valid and applicable only when USP is set high. |
| 16 | FB | RW | 0 | Fixed Burst<br>This bit defines if burst is used during burs transfer operations.<br><br>Value Description<br>0x0 The DMA bursts the entire length during burst transfers except for the last word, which is a single transfer.<br>0x1 The DMA uses only single, or fixed bursts incremented by 4, 8, or 16 during normal bus transfers. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | PR | RW | 0x0 | Priority Ratio<br>These bits control the priority ratio in the weighted round-robin arbitration between the RX DMA and TX DMA. These bits are valid only when the DA bit in this register is clear. The priority ratio is RX:TX or TX:RX depending on whether the TXPR bit in this register is clear or set.<br>Value Description<br>0x0 The Priority Ratio is 1:1.<br>0x1 The Priority Ratio is 2:1.<br>0x2 The Priority Ratio is 3:1.<br>0x3 The Priority Ratio is 4:1. |
| 13:8 | PBL | RW | 0x1 | Programmable Burst Length<br>These bits indicate the maximum number of beats to be transferred in one DMA transaction.<br>This is the maximum value that is used in a single block read or write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable only for TX DMA transactions.<br>If the number of beats to be transferred is more than 32, then perform the following steps:<br>**1.** Set the 8xPBL mode.<br>**2.** Set the PBL value.<br>For example, if the maximum number of beats to be transferred is 64, then first set 8xPBL to 1 and then set PBL to 0x8. |
| 7 | ATDS | RW | 0x0 | Alternate Descriptor Size<br>Value Description<br>0 Descriptor size reverts back to four words<br>1 Alternate descriptors, each eight words in length, are used. |
| 6:2 | DSL | RW | 0x0 | Descriptor Skip Length<br>This bit specifies the number of words to skip between two unchained descriptors.<br>The address skipping starts from the end of current descriptor to the start of next descriptor. When the DSL value is equal to zero, then the descriptor table is taken as contiguous by the DMA in Ring mode. |

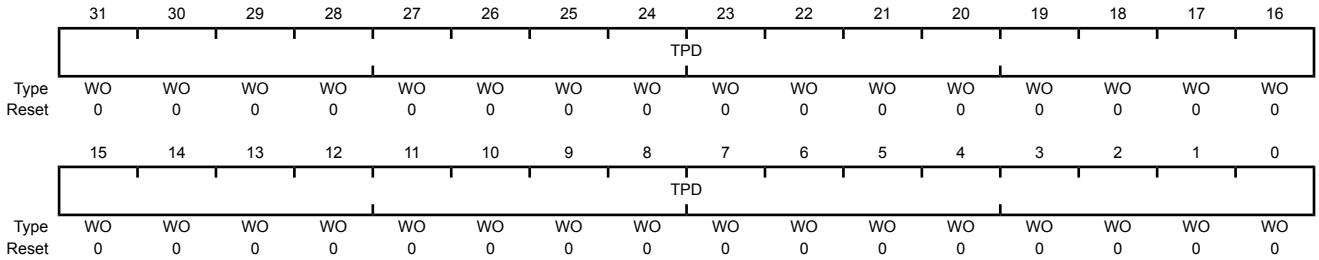
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | DA | RW | 0x0 | DMA Arbitration Scheme<br>This bit specifies the arbitration scheme between the transmit and receive paths of the DMA channel.<br><br>Value Description<br>0 Weighted round-robin with RX:TX or TX:RX.<br>The priority between the paths is according to the priority specified in the PR bit field and priority weights specified in the TXPR bit.<br>1 Fixed priority.<br>The transmit path has priority over receive path when the TXPR bit is set. Otherwise, receive path has priority over the transmit path. |
| 0 | SWR | RW | 0x1 | DMA Software Reset<br>The software reset function is driven by this bit.<br>The reset operation is completed only when all resets in all active clock domains are deasserted. It is essential that all the PHY input clocks are present for the software reset completion. The time of the software reset operation depends on the frequency of the slowest active clock.<br><br>Value Description<br>0 Reset is completed. A value of 0 should be read before reprogramming any MAC registers after a reset<br>1 MAC DMA Controller resets the logic and all internal registers of the MAC. It is cleared automatically after the reset operation has completed in all of the MAC clock domains. |

## Register 53: Ethernet MAC Transmit Poll Demand (EMACTXPOLLD), offset 0xC04

The **MAC Transmit Poll Demand (EMACTXPOLLD)** register enables the TX DMA to check whether or not the DMA owns the current descriptor. The Transmit Poll Demand command is given to wake up the TX DMA if it is in the suspend mode. The TX DMA can go into the SUSPEND mode because of an underflow error in a transmitted frame or the unavailability of descriptors owned by it. This Transmit Poll Demand command can be given at any time and the TX DMA resets this command when it again starts fetching the current descriptor from host memory.

Ethernet MAC Transmit Poll Demand (EMACTXPOLLD)

Base 0x400E.C000
Offset 0xC04
Type WO, reset 0x0000.0000



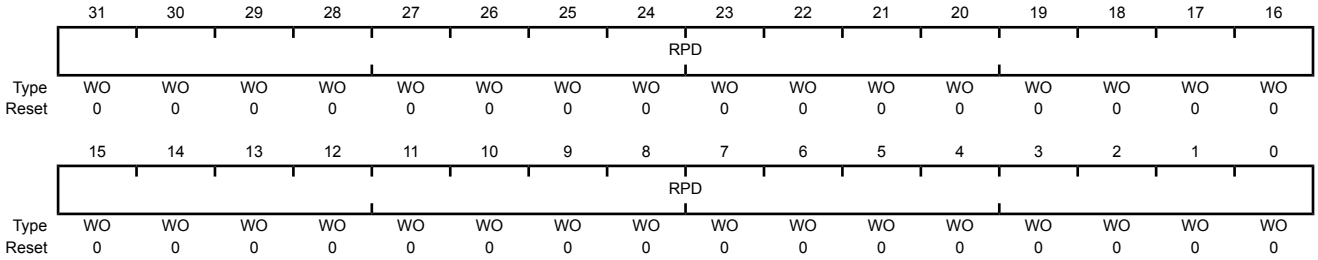

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TPD | WO | 0x0 | Transmit Poll Demand<br>When these bits are written with any value, the DMA reads the current descriptor pointed to by the **MAC Current Host Transmit Descriptor (EMACHOSTXDESC)** register. If that descriptor is not available (owned by the Host), the transmission returns to the SUSPEND state and the TU bit of the **EMACDMARIS** is asserted. If the descriptor is available, the transmission resumes. |

## Register 54: Ethernet MAC Receive Poll Demand (EMACRXPOLLD), offset 0xC08

The **Ethernet MAC Receive Poll Demand (EMACRXPOLLD)** register enables the RX DMA to check for new descriptors. This command is used to wake up the RX DMA from the SUSPEND state. The RX DMA can go into the SUSPEND state only because of the unavailability of descriptors it owns.

Ethernet MAC Receive Poll Demand (EMACRXPOLLD)

Base 0x400E.C000
Offset 0xC08
Type WO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RPD | WO | 0x0 | Receive Poll Demand<br>When these bits are written with any value, the DMA reads the current descriptor pointed to by **MAC Current Host Receive Descriptor (EMACHOSRXDESC)** register. If that descriptor is not available (owned by the Host), the reception returns to the SUSPEND state and the RU bit of **EMACDMARIS** is not asserted. If the descriptor is available, the RX DMA returns to the active state. |

## Register 55: Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR), offset 0xC0C

The **Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR)** register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be word-aligned. The DMA internally converts it to a 32-bit aligned address by making the two lease significant bits zero. Writing to this register is permitted only when the DMA receive transaction has stopped (SR = 0 n the **MAC DMA Operation Mode (EMACDMAOPMODE)** register). When stopped, this register must be written with a new descriptor list address before the receive Start command is given. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the SR bit is set to 0, then the DMA uses the already existing descriptor address.

Ethernet MAC Receive Descriptor List Address (EMACRXDLADDR)

Base 0x400E.C000
Offset 0xC0C
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STRXLIST | | | | | | | | | | | | | | | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | STRXLIST | | | | | | | | | | | | | | reserved | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | STRXLIST | RW | 0x0 | Start of Receive List<br>This field contains the base address of the first descriptor in the Receive Descriptor list. The LSB bits[1:0] are ignored and internally taken as zero by the DMA. Therefore, bits[1:0] are read-only (RO). |
| 1:0 | reserved | RO | 0x0 | 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. |

## Register 56: Ethernet MAC Transmit Descriptor List Address (EMACTXDLADDR), offset 0xC10

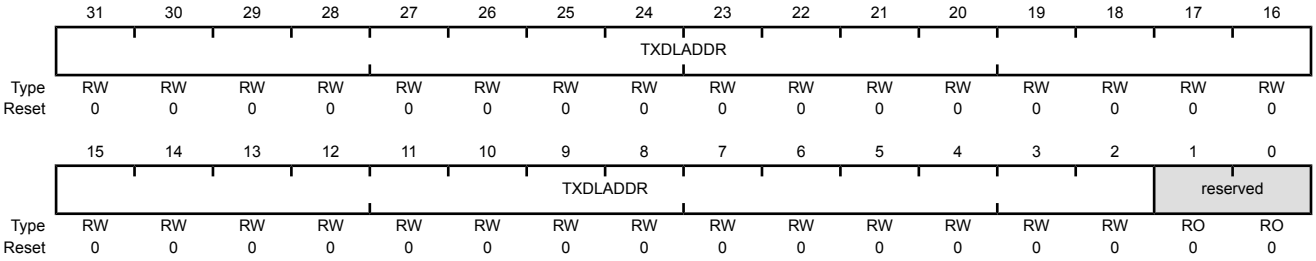
The **MAC Transmit Descriptor List Address (EMACTXDLADDR)** register points to the start of the transmit descriptor list. The descriptor lists reside in the host's physical memory space and must be word-aligned.

This register can only be written when the DMA transmit has stopped (ST = 0 in the **MAC DMA Operation Mode (EMACDMAOPMODE)** register). When the ST bit is set, the DMA takes the uses the newly programmed descriptor base address.

If this register is not changed when the ST bit is set to 0, then the DMA uses the already existing descriptor address.

Ethernet MAC Transmit Descriptor List Address (EMACTXDLADDR)

Base 0x400E.C000
Offset 0xC10
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | TXDLADDR | RW | 0x0 | Start of Transmit List Base Address<br>This field contains the base address of the first descriptor in the transmit descriptor list. The LSB bits (1:0) for 32-bit bus width are ignored and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO). |
| 1:0 | reserved | RO | 0x0 | 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. |

## Register 57: Ethernet MAC DMA Interrupt Status (EMACDMARIS), offset 0xC14

The **MAC DMA Interrupt Status (EMACDMARIS)** register contains all status bits that the DMA reports to the host. The software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits of this register are not cleared when read. Writing a 1 to any bit in the lower half-word of this register clears the bit and writing a 0 has no effect. Status bits [16:0] can be masked by enabling the appropriate mask in the **MAC DMA Interrupt Mask Register (EMACDMAIM)** register.

Ethernet MAC DMA Interrupt Status (EMACDMARIS)

Base 0x400E.C000
Offset 0xC14
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | TT | PMT | MMC | reserved | AE | | | TS | | | RS | | | NIS |
| Type RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW1C |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AIS | ERI | FBI | reserved | | ETI | RWT | RPS | RU | RI | UNF | OVF | TJT | TU | TPS | TI |
| Type RW1C | RW1C | RW1C | RO | RO | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:30 | reserved | RO | 0x0 | 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. |
| 29 | TT | RO | 0x0 | Timestamp Trigger Interrupt Status<br>Software can read the **Ethernet MAC Timestamp Status (EMACTIMSTAT)** register for the exact cause of interrupt and clear its source to reset this bit to 0. When this bit is 1, the interrupt signal from the MAC subsystem is high.<br>Value Description<br>0 No Timestamp interrupt has occurred.<br>1 An interrupt event in the Timestamp module has occurred. |
| 28 | PMT | RO | 0x0 | MAC PMT Interrupt Status<br>Software can read the **PMT Control and Status (EMACPMTCTLSTAT)** register for the exact cause of the interrupt and clear its source to reset this bit to 0. When this bit is 1, the interrupt signal from the MAC subsystem is high.<br>Value Description<br>0 No PMT interrupt has occurred.<br>1 An interrupt event in the PMT module has occurred. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MMC | RO | 0x0 | MAC MMC Interrupt<br>This bit reflects an interrupt event in the MMC module. Software must read the corresponding **EMACMMCTXRIS/EMACMMCRXRIS** register to determine the cause of the interrupt and then clear its source to reset this bit to 0.<br><br>Value Description<br>0 No MMC interrupt has occurred.<br>1 An interrupt in the MMC module has occurred. |
| 26 | reserved | RO | 0 | 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. |
| 25:23 | AE | RO | 0x0 | Access Error<br>This field indicates the type of error that caused a bus error, for example, error response on the internal bus interface. This field is valid only when bit[13] (FBI) is set. This field does not generate an interrupt.<br><br>Value Description<br>0x0 Error during RX DMA Write Data Transfer<br>0x1-0x2 reserved<br>0x3 Error during TX DMA Read Data Transfer<br>0x4 Error during RX DMA Descriptor Write Access<br>0x5 Error during TX DMA Descriptor Write Access<br>0x6 Error during RX DMA Descriptor Read Access<br>0x7 Error during TX DMA Descriptor Read Access |
| 22:20 | TS | RO | 0x0 | Transmit Process State<br>This field indicates the Transmit DMA state. This field does not generate an interrupt.<br><br>Value Description<br>0x0 Stopped; Reset or Stop transmit command processed<br>0x1 Running; Fetching transmit transfer descriptor<br>0x2 Running; Waiting for status<br>0x3 Running; Reading data from host memory buffer and queuing it to transmit buffer (TX FIFO)<br>0x4 Writing Timestamp<br>0x5 reserved<br>0x6 Suspended; Transmit descriptor unavailable or transmit buffer underflow<br>0x7 Running; Closing transmit descriptor |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 19:17 | RS | RO | 0x0 | Received Process State<br>This field indicates the Receive DMA state. This field does not generate an interrupt.<br><br>Value Description<br>0x0 Stopped: Reset or stop receive command issued<br>0x1 Running: Fetching receive transfer descriptor<br>0x2 reserved<br>0x3 Running: Waiting for receive packet<br>0x4 Suspended: Receive descriptor unavailable<br>0x5 Running: Closing receive descriptor<br>0x6 Writing Timestamp<br>0x7 Running: Transferring the receive packet data from receive buffer to host memory |
| 16 | NIS | RW1C | 0x0 | Normal Interrupt Summary<br>Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in **EMACDMAIM** register:<br><br>■ **EMACDMARIS** register, bit [0]: Transmit Interrupt<br>■ **EMACDMARIS** register, bit[2]: Transmit Buffer Unavailable<br>■ **EMACDMARIS** register, bit[6]: Receive Interrupt<br>■ **EMACDMARIS** register, bit[14]: Early Receive Interrupt<br><br>Only unmasked bits (interrupts for which interrupt enable is set in the **EMACDMAIM** register) affect the Normal Interrupt Summary bit.<br>This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit, which causes `NIS` to be set, is cleared. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | AIS | RW1C | 0x0 | Abnormal Interrupt Summary<br>Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the **EMACDMAIM** register:<br>■ **EMACDMARIS** register, bit[1]: Transmit Process Stopped<br>■ **EMACDMARIS** register, bit[3]: Transmit Jabber Timeout<br>■ **EMACDMARIS** register, bit[4]: Receive FIFO Overflow<br>■ **EMACDMARIS** register, bit[5]: Transmit Underflow<br>■ **EMACDMARIS** register, bit[7]: Receive Buffer Unavailable<br>■ **EMACDMARIS** register, bit[8]: Receive Process Stopped<br>■ **EMACDMARIS** register, bit[9]: Receive Watchdog Timeout<br>■ **EMACDMARIS** register, bit[10]: Early Transmit Interrupt<br>■ **EMACDMARIS** register, bit[13]: Fatal Bus Error<br>Only unmasked bits affect the Abnormal Interrupt Summary bit.<br>This bit must be cleared each time a corresponding bit, which causes AIS to be set, is cleared. |
| 14 | ERI | RW1C | 0x0 | Early Receive Interrupt<br>Value Description<br>0 No early receive event has occurred.<br>1 The DMA has filled the first data buffer of the packet. This bit is cleared when software writes a 1 to this bit or if bit[6] (RI) of this register is set. |
| 13 | FBI | RW1C | 0x0 | Fatal Bus Error Interrupt<br>Value Description<br>0 No bus error has occurred.<br>1 A bus error has occurred, as described in the Error Bit field (EB[25:23]). When this bit is set, the corresponding DMA engine disables all of its bus accesses.<br>This bit is cleared by writing a 1 to it. |
| 12:11 | reserved | RO | 0x0 | 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. |
| 10 | ETI | RW1C | 0x0 | Early Transmit Interrupt<br>Value Description<br>0 No early transmit has occurred.<br>1 A frame to be transmitted has been fully transferred to the TX/RX Controller Transmit FIFO.<br>This bit is cleared by writing a 1 to it. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | RWT | RW1C | 0x0 | Receive Watchdog Timeout<br><br>Value Description<br>0 No watchdog timeout event has occurred.<br>1 Indicates a frame with length greater than 2,048 bytes is received (10, 240 when Jumbo Frame mode is enabled).<br>This bit is cleared by writing a 1 to it. |
| 8 | RPS | RW1C | 0x0 | Receive Process Stopped<br><br>Value Description<br>0 No receive process stopped event has occurred.<br>1 Indicates the receive process has entered the stopped state.<br>This bit is cleared by writing a 1 to it. |
| 7 | RU | RW1C | 0x0 | Receive Buffer Unavailable<br><br>Value Description<br>0 No receive buffer unavailable event has occurred.<br>1 Indicates the host owns the next descriptor in the receive list and the DMA cannot acquire it. The receive process is suspended.<br>This bit is cleared by writing a 1 to it.<br><br>To resume processing receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, the receive process resumes when the next recognized incoming frame is received. This bit is set only when the previous receive descriptor is owned by the DMA. |
| 6 | RI | RW1C | 0x0 | Receive Interrupt<br><br>Value Description<br>0 No frame reception complete event has occurred.<br>1 A frame reception is complete. When reception is complete, bit[31] of RDES1 (disable interrupt on completion) is reset in the last descriptor, and the specific frame status information is updated in the descriptor. The reception remains in the Running state.<br>This bit is cleared by writing a 1 to it. |
| 5 | UNF | RW1C | 0x0 | Transmit Underflow<br><br>Value Description<br>0 No transmit underflow event has occurred.<br>1 Indicates the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.<br>This bit is cleared by writing a 1 to it. |

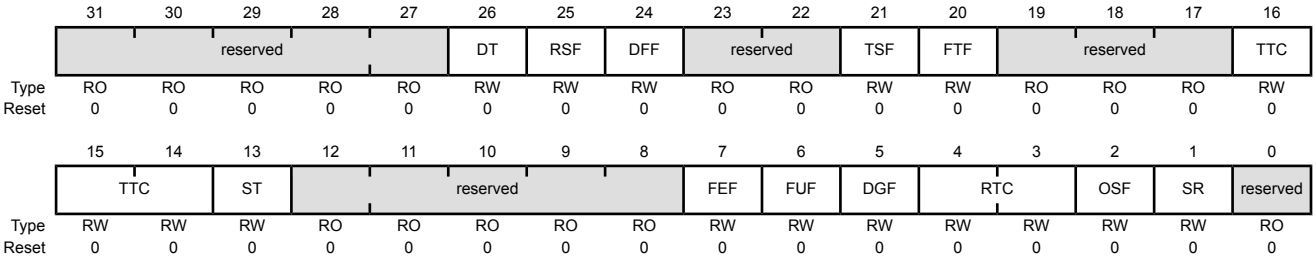
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | OVF | RW1C | 0x0 | Receive Overflow<br><br>Value Description<br>0 No receive overflow event has occurred.<br>1 The receive buffer had an overflow during frame reception. If the partial frame is transferred to the application, the overflow status is set in RDES0[11].<br>This bit is cleared by writing a 1 to it. |
| 3 | TJT | RW1C | 0x0 | Transmit Jabber Timeout<br><br>Value Description<br>0 No transmit jabber timeout event has occurred.<br>1 The Transmit Jabber Timer expired, which happens when the frame size exceeds 2,048 (10,240 bytes when Jumbo frame is enabled).<br>When the Jabber Timeout occurs, the transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.<br>This bit is cleared by writing a 1 to it. |
| 2 | TU | RW1C | 0x0 | Transmit Buffer Unavailable<br><br>Value Description<br>0 No transmit buffer unavailable event has occurred.<br>1 Indicates the host owns the next descriptor in the transmit list and the DMA cannot acquire it. Transmission is suspended. The transmit process state bits (TS[22:20]) explain the transmit process state transitions.<br>This bit is cleared by writing a 1 to it.<br><br>To resume processing Transmit descriptors, the host should change the ownership of the descriptor by setting TDES0[31] and then issue a Transmit Poll Demand command. |
| 1 | TPS | RW1C | 0x0 | Transmit Process Stopped<br><br>Value Description<br>0 Interrupt is inactive.<br>1 Indicates transmission is stopped. |
| 0 | TI | RW1C | 0x0 | Transmit Interrupt<br>This bit indicates that frame transmission is complete. When transmission is complete, the Bit 31 (Interrupt on Completion) of TDES1 is reset in the first descriptor, and the specific frame status information is updated in the descriptor.<br><br>Value Description<br>0 Frame transmission completion event has not occurred.<br>1 Frame transmission is complete.<br>This bit is cleared by writing a 1 to it. |

## Register 58: Ethernet MAC DMA Operation Mode (EMACDMAOPMODE), offset 0xC18

The **MAC DMA Operation Mode (EMACDMAOPMODE)** register establishes the Transmit and Receive operating modes and commands. This register should be the last register to be written as part of the DMA initialization.

Ethernet MAC DMA Operation Mode (EMACDMAOPMODE)

Base 0x400E.C000
Offset 0xC18
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:27 | reserved | RO | 0x0 | 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. |
| 26 | DT | RW | 0x0 | Disable Dropping of TCP/IP Checksum Error Frames<br>Value Description<br>0 All error frames are dropped if the FEF bit is reset.<br>1 The MAC does not drop the frames which only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors only in the encapsulated payload. |
| 25 | RSF | RW | 0x0 | Receive Store and Forward<br>Value Description<br>0 The RX FIFO operates in the cut-through mode, subject to the threshold specified by the RTC bits.<br>1 The TX/RX Controller reads a frame from the RX FIFO only after the complete frame has been written to it, ignoring the RTC bits. |
| 24 | DFF | RW | 0x0 | Disable Flushing of Received Frames<br>Value Description<br>0 RX DMA flushes frames based on receive descriptors or buffers.<br>1 The RX DMA does not flush any frames because of the unavailability of receive descriptors or buffers. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 23:22 | reserved | RO | 0x0 | 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. |
| 21 | TSF | RW | 0x0 | Transmit Store and Forward<br><br>Value Description<br>0 Transmission starts according to TTC bit field.<br>1 Transmission starts when a full frame resides in the TX/RX Controller Transmit FIFO. Additionally, the TTC values specified in TTC bits[16:14] are ignored. This bit should be changed only when the transmission is stopped. |
| 20 | FTF | RW | 0x0 | Flush Transmit FIFO<br><br>This bit is cleared internally when the flushing operation is completed. This register should not be written to until the FTF bit is cleared. The data which has already been accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission.<br><br>Value Description<br>0 This bit indicated normal operation or that the flushing operation has completed.<br>1 The transmit FIFO controller logic is reset to its default values and thus all data in the TX FIFO is lost or flushed.<br>This bit is cleared internally when the flushing operation is complete.<br><br>**Note:** The flush operation is complete only when the TX FIFO is emptied of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. |
| 19:17 | reserved | RO | 0x0 | 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. |
| 16:14 | TTC | RW | 0x0 | Transmit Threshold Control<br><br>These bits control the threshold level of the TX/RX Controller Transmit FIFO. Transmission starts when the frame size within the TX/RX Controller Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when Bit 21 (TSF) is reset.<br><br>Value Description<br>0x0 64 bytes<br>0x1 128 bytes<br>0x2 192 bytes<br>0x3 256 bytes<br>0x4 40 bytes<br>0x5 32 bytes<br>0x6 24 bytes<br>0x7 16 bytes |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | ST | RW | 0x0 | Start or Stop Transmission Command<br>When this bit is set, transmission is placed in the running state. The DMA attempts to acquire the descriptor from the Transmit Descriptor List. Descriptor acquisition is attempted from the current position in the list, which is the Transmit List Base Address set by **Transmit Descriptor List Address (EMACTXDLADDR)** register, or from the position retained when transmission was stopped previously.<br>If the DMA does not own the current descriptor, transmission enters the suspended state and bit[2] (Transmit Buffer Unavailable, TU) of the **MAC DMA Raw Interrupt Status Register (EMACDMARIS)** is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting **EMACTXDLADDR**, then the DMA behavior is unpredictable.<br>When this bit is cleared, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted. To change the list address, you need to program **EMACTXDLADDR** with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state.<br>Value Description<br>0 Transmission process is placed in the stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted.<br>1 Transmission is placed in the running state, and the DMA checks the transmit list at the current position for a frame to be transmitted. |
| 12:8 | reserved | RO | 0x0 | 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. |
| 7 | FEF | RW | 0x0 | Forward Error Frames<br>When this bit is reset, the RX FIFO drops frames with error status (CRC error, collision error, MII_ER, giant frame, watchdog timeout, or overflow). However, if the start byte (write) pointer of a frame is already transferred to the RX controller side (in Threshold mode), then the frame is not dropped.<br>When the FEF bit is set, all frames except runt error frames are forwarded to the DMA. If Bit 25, Receive Store and Forward (RSF), is set and the RX FIFO overflows when a partial frame is written, then the frame is dropped irrespective of the FEF bit setting. However, if the RSF is reset and the RX FIFO overflows when a partial frame is written, then a partial frame may be forwarded to the DMA.<br>Value Description<br>0 The Receive FIFO drops frames with error status<br>1 All frames except runt error frames are forwarded to the DMA. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | FUF | RW | 0x0 | Forward Undersized Good Frames<br><br>Value Description<br>0 The Receive FIFO drops all frames of less than 64 bytes, unless a frame is already transferred because of the lower value of Receive Threshold (RTC) bit field (for example, RTC = 0x1).<br>1 The Receive FIFO forwards undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC. |
| 5 | DGF | RW | 0x0 | Drop Giant Frame Enable<br><br>Value Description<br>0 The MAC does not drop the giant frames in the RX FIFO.<br>1 The MAC drops received frames larger than the computed giant frame limit in the RX FIFO. |
| 4:3 | RTC | RW | 0x0 | Receive Threshold Control<br>These two bits control the threshold level of the RX FIFO. Transfer (request) to DMA starts when the frame size within the RX FIFO is larger than the threshold. In addition, full frames with length less than the threshold are transferred automatically.<br>These bits are valid only when the RSF bit (bit 25) of the **EMACDMAOPMODE** is zero, and are ignored when the RSF bit is set to 1.<br><br>Value Description<br>0x0 64 bytes<br>0x1 32 bytes<br>0x2 96 bytes<br>0x3 128 bytes |
| 2 | OSF | RW | 0x0 | Operate on Second Frame<br><br>Value Description<br>0 DMA processes frames normally.<br>1 DMA processes second frame of the Transmit data even before the status for the first frame is obtained. |

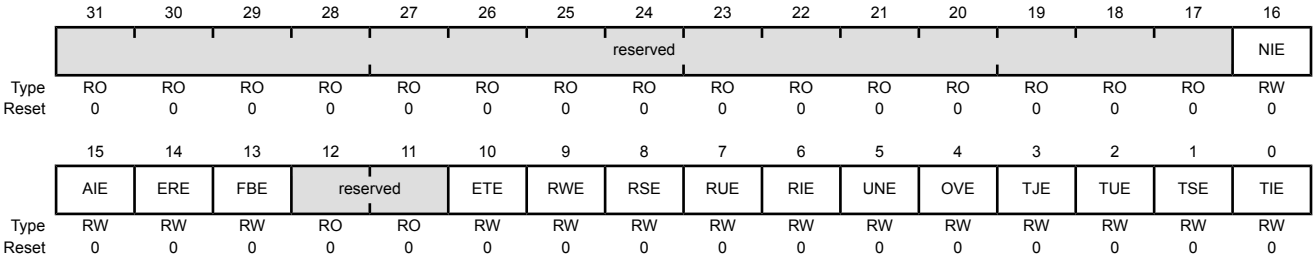
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SR | RW | 0x0 | Start or Stop Receive<br>When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes the incoming frames. The descriptor acquisition is attempted from the current position in the list, which is the address set by the Receive Descriptor List Address Register (**EMACRXDLADDR**) or the position retained when the Receive process was previously stopped. If the DMA does not own the descriptor, reception is suspended and Bit 7 (Receive Buffer Unavailable, RU) of the MAC DMA Interrupt Status Register (**EMACDMARIS**) is set. The Start Receive command is effective only when the reception has stopped. If the command is issued before setting **EMACRXDLADDR**, the DMA behavior is unpredictable.<br>When this bit is cleared, the receive DMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.<br>Value Description<br>0 The receive DMA operation is stopped after the transfer of the current frame<br>1 The Receive process is placed in the Running state. |
| 0 | reserved | RO | 0x0 | 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. |

## Register 59: Ethernet MAC DMA Interrupt Mask Register (EMACDMAIM), offset 0xC1C

The Interrupt Enable register enables the interrupts reported by the MAC DMA Interrupt Status Register (**EMACDMARIS**). Setting a bit to 0x1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

Ethernet MAC DMA Interrupt Mask Register (EMACDMAIM)

Base 0x400E.C000
Offset 0xC1C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:17 | reserved | RO | 0x0 | 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. |
| 16 | NIE | RW | 0x0 | Normal Interrupt Summary Enable<br><br>Value Description<br>0 Normal interrupt summary is masked.<br>1 Normal interrupt summary is enabled.<br><br>This bit enables/masks the ERI, RI, TU, and TI bits in **MAC DMA Interrupt Status Register (EMACDMARIS)** |
| 15 | AIE | RW | 0x0 | Abnormal Interrupt Summary Enable<br><br>Value Description<br>0 Abnormal interrupt summary is disabled.<br>1 Abnormal interrupt summary is enabled.<br><br>This bit enables/masks the TPS, TJT, OVF, UNF, RU, RPS, RWT, ETI and FBI bits in **MAC DMA Interrupt Status Register (EMACDMARIS)** |
| 14 | ERE | RW | 0x0 | Early Receive Interrupt Enable<br><br>Value Description<br>0 Early receive interrupt is disabled.<br>1 Early receive interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 16) must also be set to 0x1. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | FBE | RW | 0x0 | Fatal Bus Error Enable<br><br>Value Description<br>0 Fatal Bus Error Enable Interrupt is disabled.<br>1 Fatal Bus Error Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 12:11 | reserved | RO | 0x0 | 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. |
| 10 | ETE | RW | 0x0 | Early Transmit Interrupt Enable<br><br>Value Description<br>0 Early Transmit Interrupt is disabled.<br>1 Early Transmit Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 9 | RWE | RW | 0x0 | Receive Watchdog Timeout Enable<br><br>Value Description<br>0 The Receive Watchdog Timeout Interrupt is disabled.<br>1 The Receive Watchdog Timeout Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 8 | RSE | RW | 0x0 | Receive Stopped Enable<br><br>Value Description<br>0 Receive Stopped Interrupt is disabled.<br>1 Receive Stopped Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 7 | RUE | RW | 0x0 | Receive Buffer Unavailable Enable<br><br>Value Description<br>0 The Receive Buffer Unavailable Interrupt is disabled.<br>1 The Receive Buffer Unavailable Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 6 | RIE | RW | 0x0 | Receive Interrupt Enable<br><br>Value Description<br>0 The Receive Interrupt is disabled.<br>1 The Receive Interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 15) must also be set to 0x1. |

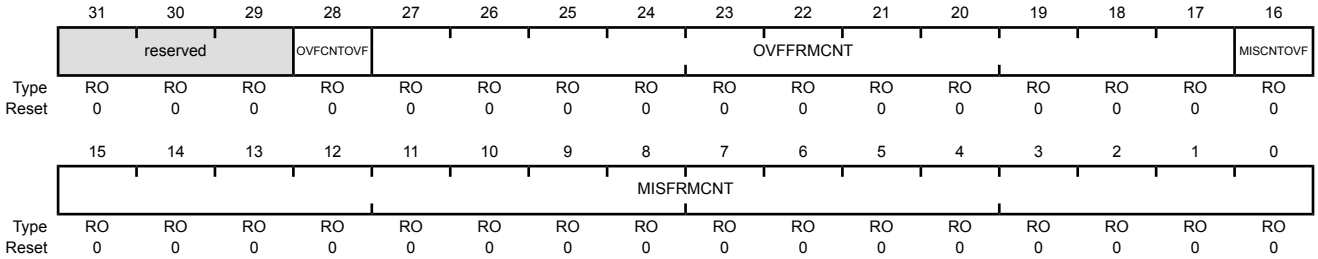
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | UNE | RW | 0x0 | Underflow Interrupt Enable<br><br>Value Description<br>0 Transmit Underflow Interrupt is disabled.<br>1 The Transmit Underflow Interrupt is enabled Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 4 | OVE | RW | 0x0 | Overflow Interrupt Enable<br><br>Value Description<br>0 The Overflow Interrupt is disabled.<br>1 The Receive Overflow Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 3 | TJE | RW | 0x0 | Transmit Jabber Timeout Enable<br><br>Value Description<br>0 Transmit Jabber Timeout Interrupt is disabled.<br>1 Transmit Jabber Timeout Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 2 | TUE | RW | 0x0 | Transmit Buffer Unvailable Enable<br><br>Value Description<br>0 Transmit Buffer Unavailable Interrupt is disabled.<br>1 Transmit Buffer Unavailable Interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 15) must also be set to 0x1. |
| 1 | TSE | RW | 0x0 | Transmit Stopped Enable<br><br>Value Description<br>0 Transmission Stopped Interrupt is disabled.<br>1 Transmission Stopped Interrupt is enabled. Abnormal Interrupt Summary Enable (AIE, bit 15) must also be set to 0x1. |
| 0 | TIE | RW | 0x0 | Transmit Interrupt Enable<br><br>Value Description<br>0 Transmit Interrupt is disabled.<br>1 Transmit Interrupt is enabled. Normal Interrupt Summary Enable (NIE, bit 15) must also be set to 0x1. |

## Register 60: Ethernet MAC Missed Frame and Buffer Overflow Counter (EMACMFBOC), offset 0xC20

The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counters. The counter is used for diagnostic purposes. The MISFRMCNT field indicates missed frames because of the host buffer being unavailable. The OVFFRMCNT field indicates missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

Ethernet MAC Missed Frame and Buffer Overflow Counter (EMACMFBOC)

Base 0x400E.C000
Offset 0xC20
Type RO, reset 0x0000.0000



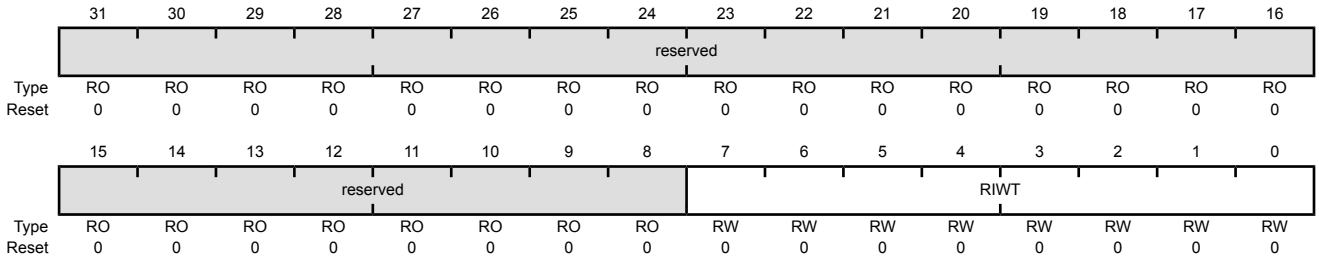

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | OVFCNTOVF | RO | 0 | Overflow Bit for FIFO Overflow Counter<br>This bit is set every time the overflow frame counter (bits [27:17]) overflows; that is, the RX FIFO overflows with the overflow frame counter at maximum value. In such a scenario, the overflow frame counter is reset to all zeros and this bit indicates the rollover happened. |
| 27:17 | OVFFRMCNT | RO | 0x0 | Overflow Frame Counter<br>This field indicates the number of frames missed by the application. This counter is incremented each time the TX/RX Controller indicates overflow.<br>This counter is cleared when the TX/RX Controller accepts data. |
| 16 | MISCNTOVF | RO | 0x0 | Overflow bit for Missed Frame Counter<br>This bit is set every time the Missed Frame Counter (bits [15:0]) overflows; which means the DMA discards an incoming frame because of the Host Receive Buffer being unavailable with the missed frame counter at maximum value. In such a scenario, the Missed Frame Counter is reset to all zeros and this bit indicates that the rollover happened. |
| 15:0 | MISFRMCNT | RO | 0x0 | Missed Frame Counter<br>This field indicates the number of frames missed by the controller because of the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. This counter is cleared when the DMA accepts frames. |

## Register 61: Ethernet MAC Receive Interrupt Watchdog Timer (EMACRXINTWDT), offset 0xC24

This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt, RI (Bit 6), of the **EMACDMARIS** register at EMAC offset 0xC14.

Ethernet MAC Receive Interrupt Watchdog Timer (EMACRXINTWDT)

Base 0x400E.C000
Offset 0xC24
Type RW, reset 0x0000.0000



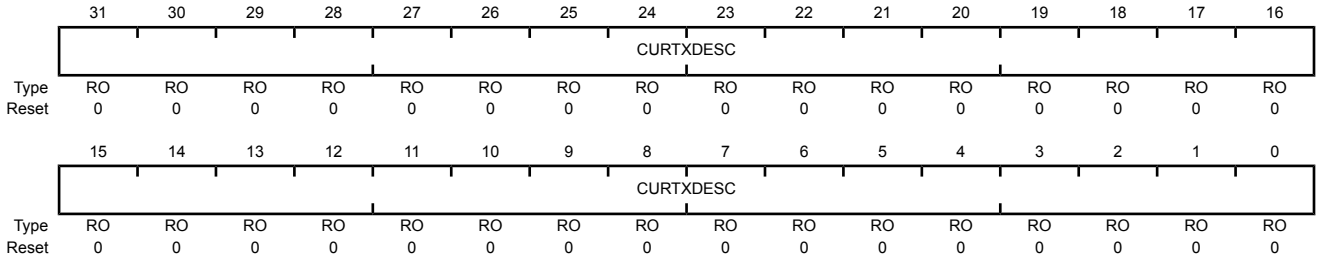

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | RIWT | RW | 0x0 | Receive Interrupt Watchdog Timer Count<br>This field indicates the period in which the receive counter expires. The value in this field is programmed by 256 to calculate the number of system clock periods the timer must count.<br>Watchdog Timer Period = (RIWT * 256) system clocks.<br>When the watchdog timer runs out, the RI bit is set and the timer is stopped.<br>If the RDES[31] bit is clear, the watchdog timer is reset. |

## Register 62: Ethernet MAC Current Host Transmit Descriptor (EMACHOSTXDESC), offset 0xC48

The **MAC Current Host Transmit Descriptor (EMACHOSTXDESC)** register points to the start address of the current Transmit Descriptor read by the DMA.

Ethernet MAC Current Host Transmit Descriptor (EMACHOSTXDESC)

Base 0x400E.C000
Offset 0xC48
Type R, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CURTXDESC | RO | 0x0 | Host Transmit Descriptor Address Pointer<br>This register contains the start address of the current transmit descriptor read by the DMA.<br>This register is cleared on reset. The pointer is updated by the DMA during operation. |

## Register 63: Ethernet MAC Current Host Receive Descriptor (EMACHOSRXDESC), offset 0xC4C

The **MAC Current Host Receive Descriptor (EMACHOSRXDESC)** register points to the start address of the current Receive Descriptor read by the DMA.

Ethernet MAC Current Host Receive Descriptor (EMACHOSRXDESC)

Base 0x400E.C000
Offset 0xC4C
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CURRXDESC | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CURRXDESC | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CURRXDESC | RO | 0x0 | Host Receive Descriptor Address Pointer<br>This register contains the start address of the current receive descriptor read by the DMA.<br>This register is cleared on reset. The pointer is updated by the DMA during operation. |

## Register 64: Ethernet MAC Current Host Transmit Buffer Address (EMACHOSTXBA), offset 0xC50

The **MAC Current Host Transmit Buffer Address (EMACHOSTXBA)** register points to the current Transmit Buffer Address being read by the DMA.

Ethernet MAC Current Host Transmit Buffer Address (EMACHOSTXBA)

Base 0x400E.C000
Offset 0xC50
Type R, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CURTXBUFA | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CURTXBUFA | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

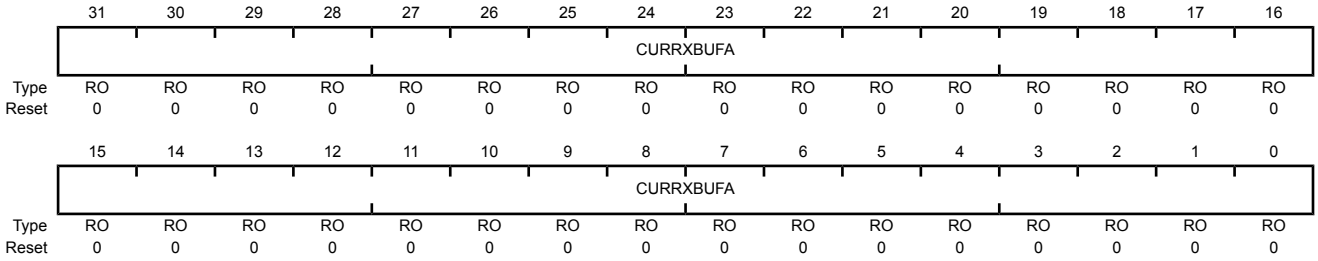
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CURTXBUFA | RO | 0x0 | Host Transmit Buffer Address Pointer<br>This register contains the current transmit buffer address being read by the DMA.<br>This register is cleared on reset. The pointer is updated by the DMA during operation. |

## Register 65: Ethernet MAC Current Host Receive Buffer Address (EMACHOSRXBA), offset 0xC54

The **MAC Current Host Receive Buffer Address (EMACHOSRXBA)** register points to the current receive buffer address being read by the DMA.

Ethernet MAC Current Host Receive Buffer Address (EMACHOSRXBA)

Base 0x400E.C000
Offset 0xC54
Type R, reset 0x0000.0000



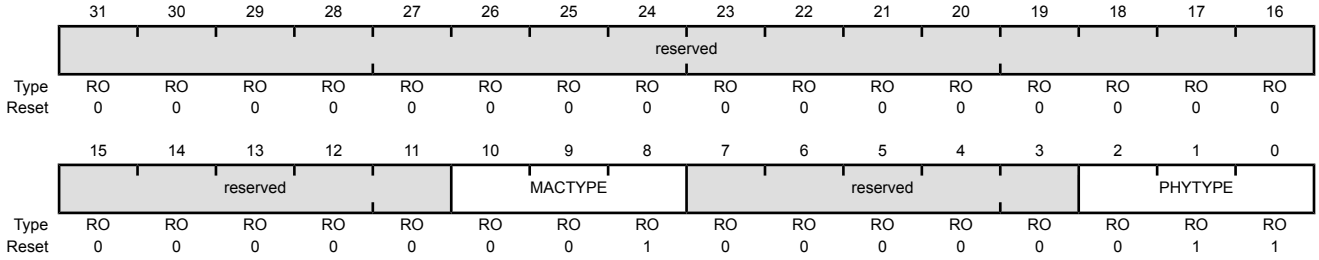

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CURRXBUFA | RO | 0x0 | Host Receive Buffer Address Pointer<br>This register contains the current receive buffer address being read by the DMA.<br>This register is cleared on reset. The pointer is updated by the DMA during operation. |

## Register 66: Ethernet MAC Peripheral Property Register (EMACPP), offset 0xFC0

This register defines the Ethernet MAC and PHY type used.

Ethernet MAC Peripheral Property Register (EMACPP)

Base 0x400E.C000
Offset 0xFC0
Type RO, reset 0x0000.0103




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:11 | reserved | RO | 0x0 | 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. |
| 10:8 | MACTYPE | RO | 0x1 | Ethernet MAC Type<br>Value Description<br>0x0 Stellaris® LM3S-class MAC.<br>0x1 Tiva™ TM4C129x-class MAC.<br>0x2-0x7 Reserved |
| 7:3 | reserved | RO | 0 | 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. |
| 2:0 | PHYTYPE | RO | 0x3 | Ethernet PHY Type<br>This field specifies the type of PHY provided.<br>Value Description<br>0x0 No PHY<br>0x1 Fury class PHY<br>0x2 Tempest/Firestorm class PHY<br>0x3 Snowflake class PHY<br>0x4-0x7 Reserved |

## Register 67: Ethernet MAC Peripheral Configuration Register (EMACPC), offset 0xFC4

The **Ethernet MAC Peripheral Configuration Register (EMACPC)** register configures the MAC and PHY reset and interface parameters.

Ethernet MAC Peripheral Configuration Register (EMACPC)

Base 0x400E.C000
Offset 0xFC4
Type RW, reset 0x0080.040E

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PHYEXT | PINTFS | | | reserved | | DIGRESTART | NIBDETDIS | RXERIDLE | ISOMIILL | LRR | TDRRUN | FASTLDMODE | | | |
| Type | RW | RW | RW | RW | RO | RO | RW | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FASTLDMODE | POLSWAP | MDISWAP | RBSTMDIX | FASTMDIX | MDIXEN | FASTRXDV | FASTLUPD | EXTFD | FASTANEN | FASTANSEL | | ANEN | ANMODE | | PHYHOLD |
| Type | RW | RW | RW | RW | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | PHYEXT | RW | 0 | PHY Select<br>This bit is used to select whether the internal or an external PHY is used.<br><br>Value Description<br>0 Internal PHY<br>1 External PHY |
| 30:28 | PINTFS | RW | 0 | Ethernet Interface Select<br>This field selects the PHY interface used by the MAC.<br>This input is sampled during reset and an update to this register field must result in the MAC undergoing a reset event.<br>This field has the following encoded values:<br><br>Value Description<br>0x0 MII (default) Used for internal PHY or external PHY connected via MII.<br>0x1-0x3 reserved<br>0x4 RMII: Used for external PHY connected via RMII.<br>0x5-0x7 reserved |
| 27:26 | reserved | RO | 0x0 | 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. |
| 25 | DIGRESTART | RW | 0 | PHY Soft Restart<br>This bit allows the user to restart the PHY. Asserting this bit causes the PHY logic and internal register to reset to initial conditions. This bit does not affect the configuration bits provided by the **EMACPC** register, which are stored in the PHY following a chip reset. To initiate the soft reset to the PHY, this bit must be written to a 1 and written again to a 0. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 24 | NIBDETDIS | RO | 0 | Odd Nibble TXER Detection Disable<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the ODDNDETDIS bit of the **Ethernet PHY Configuration 2 (EPHYCFG2)** register, PHY offset 0x00A. |
| 23 | RXERIDLE | RW | 1 | RXER Detection During Idle<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the RXERRIDLE bit of the **Ethernet PHY Configuration 2 (EPHYCFG2)** register, PHY offset 0x00A. |
| 22 | ISOMIILL | RW | 0 | Isolate MII in Link Loss<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the ISOMIILL bit of the **Ethernet PHY Configuration 2 (EPHYCFG2)** register, PHY offset 0x00A. |
| 21 | LRR | RW | 0 | Link Loss Recovery<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the LLR bit of the **Ethernet PHY Configuration 1 (EPHYCFG1)** register, PHY offset 0x009. |
| 20 | TDRRUN | RW | 0 | TDR Auto Run<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the TDRAR bit of the **Ethernet PHY Configuration 1 (EPHYCFG1)** register, PHY offset 0x009. |
| 19:15 | FASTLDMODE | RW | 0 | Fast Link Down Mode<br>These bits are sampled on the deassertion of the PHY reset signal and are used as the default for the FLDWNM bit field of the **Ethernet PHY Configuration 3 (EPHYCFG3)** register, PHY offset 0x00B. |
| 14 | POLSWAP | RW | 0 | Polarity Swap<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the POLSWAP bit of the **Ethernet PHY Configuration 3 (EPHYCFG3)** register PHY offset 0x00B. |
| 13 | MDISWAP | RW | 0 | MDI Swap<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the MDIMDIXS bit of the **Ethernet PHY Configuration 3 (EPHYCFG3)** register, PHY offset 0x00B. |
| 12 | RBSTMDIX | RW | 0 | Robust Auto MDI-X<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the RAMDIX bit of the **Ethernet PHY Configuration 1 (EPHYCFG1)** register, PHY offset 0x009. |
| 11 | FASTMDIX | RO | 0 | Fast Auto MDI-X<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the FAMDIX bit of the **Ethernet PHY Configuration 1 (EPHYCFG1)** register, PHY offset 0x009. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | MDIXEN | RW | 1 | MDIX Enable<br>This bit is sampled on the deassertion of the PHY reset signal and is used to determine whether automatic MDI/MDIX crossover is enabled.<br><br>Value Description<br>0 Disable automatic cross-over.<br>1 Enable automatic cross-over. |
| 9 | FASTRXDV | RW | 0 | Fast RXDV Detection<br>This bit is sampled on the deassertion of the PHY reset signal and is used to select whether fast RXDV detection is enabled in the PHY.<br><br>Value Description<br>0 Disable fast RXDV detection.<br>1 Enable fast RXDV detection. |
| 8 | FASTLUPD | RW | 0 | FAST Link-Up in Parallel Detect<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default value for the FLUPPD bit of the **Ethernet PHY Configuration 2 (EPHYCFG2)** register, PHY offset 0x00A. |
| 7 | EXTFD | RW | 0 | Extended Full Duplex Ability<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default value for the EXTFD bit of the **Ethernet PHY Configuration 2 (EPHYCFG2)** register, PHY offset 0x00A. |
| 6 | FASTANEN | RW | 0 | Fast Auto Negotiation Enable<br>This bit is sampled on the deassertion of the PHY reset signal and is used as the default for the FASTANEN bit of the **Ethernet PHY Configuration 1 (EPHYCFG1)** register, PHY offset 0x009. |
| 5:4 | FASTANSEL | RW | 0 | Fast Auto Negotiation Select<br>These bits are sampled on the deassertion of the PHY reset signal and are used as the defaults for the FANSEL bit field of the **Ethernet PHY Configuration 1 (EPHYCFG1)** register, PHY offset 0x009. |
| 3 | ANEN | RW | 1 | Auto Negotiation Enable<br>This bit is sampled on the deassertion of the PHY reset signal and is to select whether auto-negotiation is enabled.<br><br>Value Description<br>0 Auto-negotiation disabled.<br>1 Auto-negotiation enabled. |

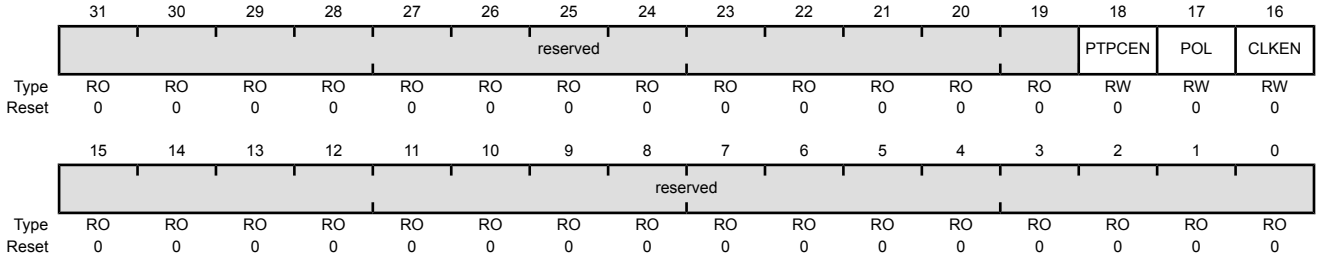
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2:1 | ANMODE | RW | 3 | Auto Negotiation Mode<br>These bits are sampled on the deassertion of the PHY reset signal and are used to determine the auto-negotiation mode of the PHY.<br><br>Value Description<br>0x0 When ANEN = 0x0, the mode is 10Base-T, Half-Duplex<br>When ANEN=0x1, the mode is 10Base-T, Half/Full-Duplex<br>0x1 When ANEN = 0x0, the mode is 10Base-T, Full-Duplex<br>When ANEN=0x1, the mode is 100Base-TX, Half/Full-Duplex<br>0x2 When ANEN = 0x0, the mode is 100Base-TX, Half-Duplex<br>When ANEN=0x1, the mode is 10Base-T,Half-Duplex 100Base-TX, Half-Duplex<br>0x3 When ANEN = 0x0, the mode is 100Base-TX, Full-Duplex<br>When ANEN=0x1, the mode is 10Base-T,Half/Full-Duplex 100Base-TX, Half/Full-Duplex |
| 0 | PHYHOLD | RW | 0 | Ethernet PHY Hold<br>This bit is sampled on the deassertion of the PHY reset signal and is used to keep the PHY from transmitting energy on the line.<br><br>Value Description<br>0 PHY transmits energy on the line.<br>1 PHY is held off from transmitting energy on the line. |

## Register 68: Ethernet MAC Clock Configuration Register (EMACCC), offset 0xFC8

The following register is used to configure the clocks of the Ethernet Controller.

Ethernet MAC Clock Configuration Register (EMACCC)

Base 0x400E.C000
Offset 0xFC8
Type RO, reset 0x0000.0000



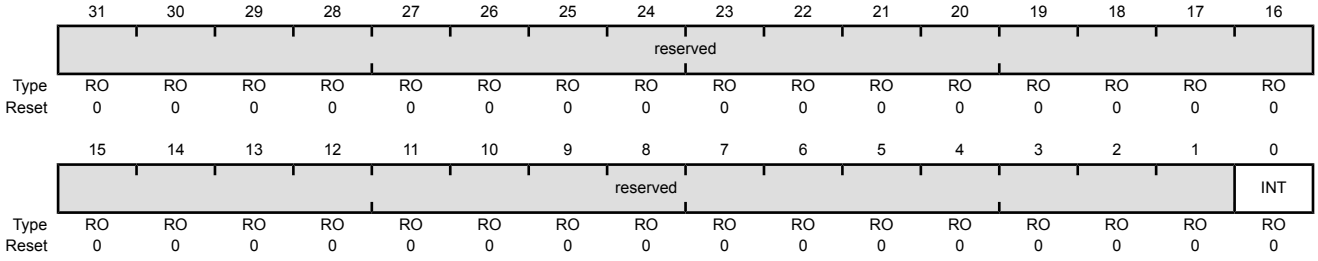

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 18 | PTPCEN | RW | 0 | PTP Clock Reference Enable<br>The PTP clock reference is MOSC.<br>This bit enables the MOSC to drive the PTP clock reference of the Ethernet MAC.<br><br>Value Description<br>0 PTP clock reference is disabled.<br>1 PTP clock reference is enabled. |
| 17 | POL | RW | 0 | LED Polarity Control<br>This bit controls the polarity of the LED outputs coming from the Ethernet PHY.<br><br>Value Description<br>0 LEDs are active high.<br>1 LEDs are active low. |
| 16 | CLKEN | RW | 0 | EN0RREF_CLK Signal Enable<br>Note that when using the RMII interface, this bit must be set to a 1.<br><br>Value Description<br>0 EN0RREF_CLK signal is disabled<br>1 EN0RREF_CLK signal is enabled |
| 15:0 | reserved | RO | 0 | 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. |

## Register 69: Ethernet PHY Raw Interrupt Status (EPHYRIS), offset 0xFD0

The **Ethernet PHY Raw Interrupt Status (EPHYRIS)** register is used to mask the interrupt from the Ethernet PHY, which is either from the internal integrated PHY or an external PHY.

Ethernet PHY Raw Interrupt Status (EPHYRIS)

Base 0x400E.C000
Offset 0xFD0
Type RO, reset 0x0000.0000



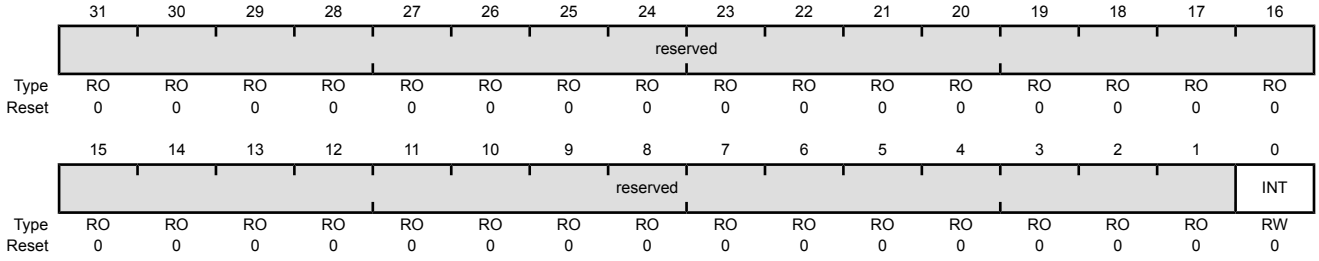

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | INT | RO | 0 | Ethernet PHY Raw Interrupt Status<br>Value Description<br>0 No interrupt has triggered.<br>1 The Ethernet PHY has signaled an interrupt using the EN0INTR input. |

## Register 70: Ethernet PHY Interrupt Mask (EPHYIM), offset 0xFD4

The **Ethernet PHY Interrupt Mask (EPHYIM)** register is used to mask the interrupt from the Ethernet PHY, which is either from the internal integrated PHY or an external PHY.

Ethernet PHY Interrupt Mask (EPHYIM)

Base 0x400E.C000
Offset 0xFD4
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | INT | RW | 0 | Ethernet PHY Interrupt Mask |

| Value | Description |
|---|---|
| 0 | An Ethernet PHY interrupt is suppressed and not sent to the interrupt controller. |
| 1 | An Ethernet PHY interrupt is sent to the interrupt controller when the INT bit is set in the **EPHYRIS** register.<br>This interrupt could be generated from either the integrated PHY or an external PHY through the EN0INTR signal depending on the configuration chosen. |

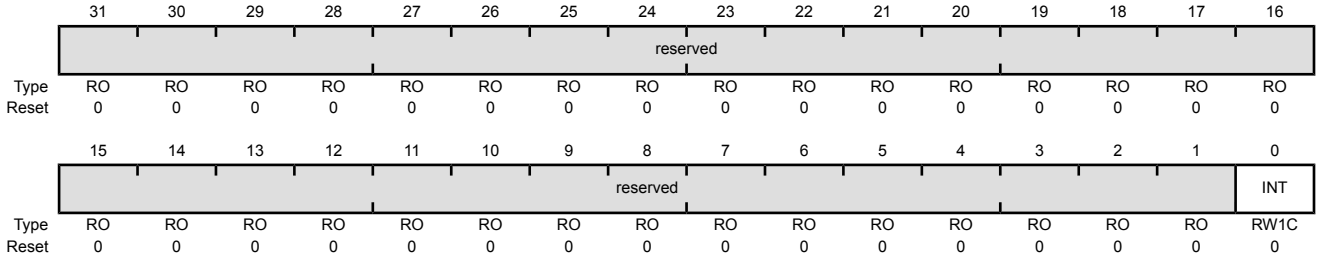
### Register 71: Ethernet PHY Masked Interrupt Status and Clear (EPHYMISC), offset 0xFD8

The **Ethernet Masked Interrupt Status and Clear (EPHYMISC)** register displays the masked interrupt status of the Ethernet PHY, which is either from the internal integrated PHY or an external PHY. This register can be written to clear the **EPHYRIS** register.

This register is used for clearing the **EPHYRIS** register bits.

Ethernet PHY Masked Interrupt Status and Clear (EPHYMISC)

Base 0x400E.C000
Offset 0xFD8
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | 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. |
| 0 | INT | RW1C | 0 | Ethernet PHY Status and Clear register<br>Reading this register provides a result which is the logical AND of the **EPHYRIS** and **EPHYIM** registers. A write of 1 to a bit of this register clears the corresponding bit in the **EPHYRIS** register.<br>**Note:** The Ethernet MAC interrupt is an OR'd summary of both the masked **EMACRIS** register output and this register. When an Ethernet MAC interrupt is asserted, software mush check both the **EMACRIS** and **EMACIM** registers along with this register. |

## 24.8 Ethernet PHY Register Descriptions

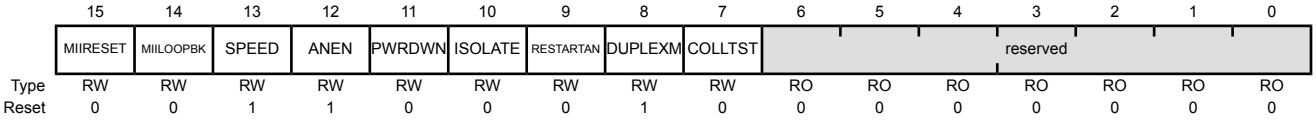
This section lists and describes the PHY registers, in numerical order by address. Also see “Ethernet MAC Register Descriptions” on page 1657.

## Register 72: Ethernet PHY Basic Mode Control - MR0 (EPHYBMCR), address 0x000

This register describes the basic mode controls available to the EPHY. The reset state of the ANEN bit is controlled by the **EMACPC** register.

Ethernet PHY Basic Mode Control - MR0 (EPHYBMCR)

Base n/a
Address 0x000
Type RW, reset 0x3100




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | MIIRESET | RW | 0 | MII Register reset<br>Writing a 1 to this bit resets the contents of the MII-related registers, **EPHYBMCR** (0x000) **EPHYANA** (0x004) and **EPHYANNPTR** (0x007) registers . When the reset operation is done, this bit is cleared to 0 automatically.<br>Value Description<br>0 Normal operation.<br>1 Initiate MII Reset / Reset in Process. |
| 14 | MIILOOPBK | RW | 0 | MII Loopback<br>When MII loopback mode is activated, the transmitter data presented on MII TXD is looped back to MII RXD internally.<br>Value Description<br>0 Normal operation.<br>1 MII Loopback enabled. |
| 13 | SPEED | RW | 1 | Speed Select<br>When auto-negotiation is disabled writing to this bit allows the port speed to be selected.<br>Value Description<br>0 10Mbs<br>1 100Mbs |
| 12 | ANEN | RW | 1 | Auto-Negotiate Enable<br>Value Description<br>0 Auto-Negotiation Disabled – the SPEED bit and the DUPLEXM bit determine the port speed and duplex mode.<br>1 Auto-Negotiation Enabled – the SPEED bit and the DUPLEXM bit of this register are ignored when this bit is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | PWRDWN | RW | 0 | Power Down<br>Setting this bit powers down the PHY. Only minimal register functionality is enabled during the power down condition.<br>Value Description<br>0 Normal operation<br>1 Power-down modes are enabled: General Power Down Mode, Active Sleep Mode and Passive Sleep Mode (see **Ethernet PHY Specific Control (EPHYSCR)** register, offset 0x011. |
| 10 | ISOLATE | RW | 0 | Port Isolate<br>Value Description<br>0 Normal operation.<br>1 Isolates the Port from the MII with the exception of the serial management. |
| 9 | RESTARTAN | RW | 0 | Restart Auto-Negotiation<br>Value Description<br>0 Normal operation.<br>1 Auto-Negotiation process is re-initiated. If Auto-Negotiation is disabled (ANEN = 0), this bit is ignored. This bit is self-clearing and returns a value of 1 until Auto-Negotiation is initiated, whereupon it self-clears. Operation of the Auto-Negotiation process is not affected by the management entity clearing this bit.. |
| 8 | DUPLEXM | RW | 1 | Duplex Mode<br>When auto-negotiation is disabled writing to this bit allows the port-duplex capability to be selected.<br>Value Description<br>0 Half Duplex operation.<br>1 Full Duplex Operation |
| 7 | COLLTST | RW | 0 | Collision Test<br>When set, this bit causes the EN0COL signal to be asserted in response to the assertion of EN0TXEN within 512 bit times. The EN0COL signal is deasserted within four bit times in response to the deassertion of EN0TXEN.<br>Value Description<br>0 Normal operation<br>1 Collision test enabled. |
| 6:0 | reserved | RO | 0 | 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. |

## Register 73: Ethernet PHY Basic Mode Status - MR1 (EPHYBMSR), address 0x001

This register reflects the basic mode features that are available in the EPHY.

Ethernet PHY Basic Mode Status - MR1 (EPHYBMSR)

Base n/a
Address 0x001
Type RO, reset 0x7849

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | 100BTXFD | 100BTXHD | 10BTFD | 10BTHD | reserved | | | | MFPRESUP | ANC | RFAULT | ANEN | LINKSTAT | JABBER | EXTEN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | reserved | RO | 0 | 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. |
| 14 | 100BTXFD | RO | 1 | 100Base-TX Full Duplex Capable<br>Value Description<br>0 Device does not support 100Base-TX in full duplex mode.<br>1 Device supports 100Base-TX in full duplex mode. |
| 13 | 100BTXHD | RO | 1 | 100Base-TX Half Duplex Capable<br>Value Description<br>0 The device does not support 100Base-TX in half duplex mode.<br>1 The device supports 100Base-TX in half duplex mode. |
| 12 | 10BTFD | RO | 1 | 10 Base-T Full Duplex Capable<br>Value Description<br>0 The device does not support 10Base-T in full duplex mode.<br>1 Device able to perform 10Base-T in full duplex mode. |
| 11 | 10BTHD | RO | 1 | 10 Base-T Half Duplex Capable<br>Value Description<br>0 Device not able to perform 10Base-T in half duplex mode.<br>1 Device able to perform 10Base-T in half duplex mode. |
| 10:7 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | MFPRESUP | RO | 1 | Preamble Suppression Capable<br>Value Description<br>0 The device does not perform management transactions with preambles suppressed.<br>1 The device performs management transactions with preambles suppressed. For this mode, the 32-bits of preamble are needed only once after reset, invalid opcode or invalid turnaround. |
| 5 | ANC | RO | 0 | Auto-Negotiation Complete<br>Value Description<br>0 Auto-Negotiation process not complete (either still in process, disabled, or reset)<br>1 Auto-Negotiation process complete. |
| 4 | RFAULT | RO | 0 | Remote Fault<br>Value Description<br>0 No remote fault condition detected.<br>1 Remote Fault condition detected (cleared on read or by reset). Fault criteria: Far End Fault Indication or notification from Link Partner of Remote Fault. |
| 3 | ANEN | RO | 1 | Auto Negotiation Enabled<br>Value Description<br>0 Device is not able to perform Auto-Negotiation.<br>1 Device is able to perform Auto-Negotiation. |
| 2 | LINKSTAT | RO | 0 | Link Status<br>Value Description<br>0 Link not established.<br>1 Valid link established (for either 10 or 100Mbs operation). |
| 1 | JABBER | RO | 0 | Jabber Detect<br>This bit is implemented with a latching function, such that the occurrence of a jabber condition causes it to set until it is cleared by a read from this register, by the management interface, or by a reset.<br>Value Description<br>0 No Jabber condition detected.<br>1 Jabber condition detected |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | EXTEN | RO | 1 | Extended Capability Enable |

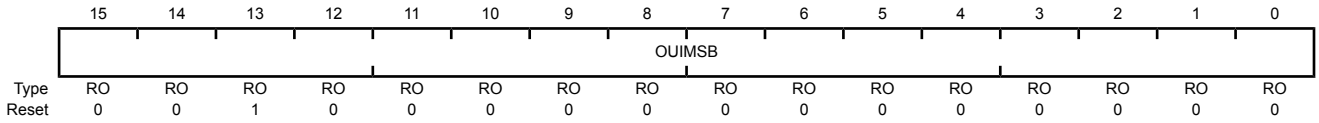
| Value | Description |
|---|---|
| 0 | Basic register set capabilities only. |
| 1 | Extended register capabilities. |

## Register 74: Ethernet PHY Identifier Register 1 - MR2 (EPHYID1), address 0x002

The **Ethernet PHY Identifier 1 and 2 (EPHYIDn)** registers together form a unique identifier. The identifier consists of a concatenation of the Organizationally Unique Identifier (OUI), the vendor's model number and the model revision number. A PHY may return a value of zero in each of the 32 bits of the PHY Identifier if desired. The PHY Identifier is intended to support network management. The Texas Instruments IEEE-assigned OUI is 0x080028. This OUI field is broken into two fields in the **EPHYID1** and **EPHYID2** register. The most significant OUI field, OUIMSB in the **EPHYID1** register, is equal to 0x0002 (the two most significant bits of the OUI are ignored). The least significant OUI field, OUILSB in the **EPHYID2** register, is equal to 0x28.

Ethernet PHY Identifier Register 1 - MR2 (EPHYID1)

Base n/a
Address 0x002
Type R, reset 0x2000



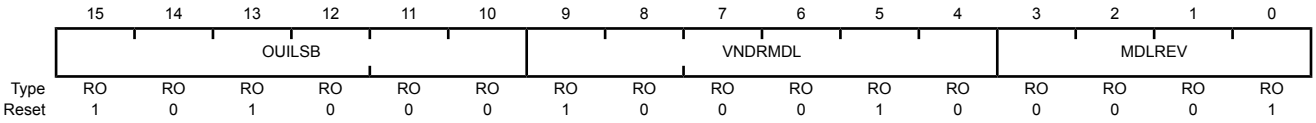

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:0 | OUIMSB | RO | 0x2000 | OUI Most Significant Bits<br>OUI[21:6] = 0x200. The most significant two bits of the OUI are ignored (the IEEE standard refers to these as bits 1 and 2). |

## Register 75: Ethernet PHY Identifier Register 2 - MR3 (EPHYID2), address 0x003

The **Ethernet PHY Identifier 1 and 2 (EPHYIDn)** registers together form a unique identifier. The identifier consists of a concatenation of the Organizationally Unique Identifier (OUI), the vendor's model number and the model revision number. A PHY may return a value of zero in each of the 32 bits of the PHY Identifier if desired. The PHY Identifier is intended to support network management. The Texas Instruments IEEE-assigned OUI is 0x080028. This OUI field is broken into two fields in the **EPHYID1** and **EPHYID2** register. The most significant OUI field, `OUIMSB` in the **EPHYID1** register, is equal to 0x0002 (the two most significant bits of the OUI are ignored). The least significant OUI field, `OUILSB` in the **EPHYID2** register, is equal to 0x28.

Ethernet PHY Identifier Register 2 - MR3 (EPHYID2)

Base n/a
Address 0x003
Type R, reset 0xA221



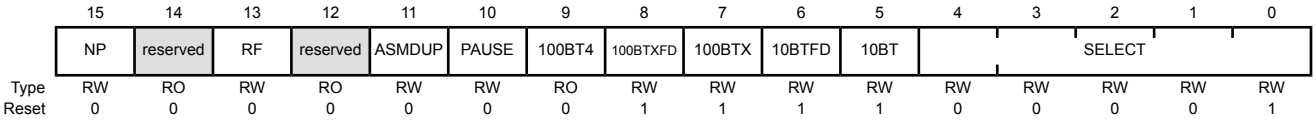

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:10 | OUILSB | RO | 0x28 | OUI Least Significant Bits<br>Bits 19 to 24 of the OUI (0x080028) are mapped from bits 15 to 10 of this register respectively. |
| 9:4 | VNDRMDL | RO | 0x22 | Vendor Model Number<br>The six bits of vendor model number are mapped from bits 9 to 4 (most significant bit to bit 9). |
| 3:0 | MDLREV | RO | 0x1 | Model Revision Number<br>Four bits of the vendor model revision number are mapped from bits 3 to 0 (most significant bit to bit 3). This field is incremented for all major device changes. |

## Register 76: Ethernet PHY Auto-Negotiation Advertisement - MR4 (EPHYANA), address 0x004

This register contains the advertised abilities of this device as they are transmitted to its link partner during Auto-Negotiation.

Ethernet PHY Auto-Negotiation Advertisement - MR4 (EPHYANA)

Base n/a
Address 0x004
Type RW, reset 0x01E1



| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NP | reserved | RF | reserved | ASMDUP | PAUSE | 100BT4 | 100BTXFD | 100BTX | 10BTFD | 10BT | SELECT | | | | |
| Type | RW | RO | RW | RO | RW | RW | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | NP | RW | 0 | Next Page Indication<br><br>Value Description<br>0 Next Page Transfer not desired.<br>1 Next Page Transfer desired. |
| 14 | reserved | RO | 0 | 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. |
| 13 | RF | RW | 0 | Remote Fault<br><br>Value Description<br>0 No Remote Fault detected.<br>1 Advertises that this device has detected a Remote Fault. |
| 12 | reserved | RO | 0 | 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. |
| 11 | ASMDUP | RW | 0 | Asymmetric PAUSE support for Full Duplex Links<br>Encoding and resolution of PAUSE bits is defined in IEEE 802.3 Annex 28B, Tables 28B-2 and 28B-3, respectively. Pause resolution status is reported in the Pause Status bits [13:12], of the **Ethernet PHY Control (EPHYCTL)** register.<br><br>Value Description<br>0 Asymmetric PAUSE not implemented.<br>1 Asymmetric PAUSE implemented. Advertise that the MAC has implemented both the optional MAC control sublayer and the pause function as specified in clause 31 and annex 31B of IEEE802.3u. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | PAUSE | RW | 0 | PAUSE Support for Full Duplex Links<br>The PAUSE bit indicates that the device is capable of providing the symmetric PAUSE functions as defined in Annex 31B.<br>Encoding and resolution of PAUSE bits is defined in IEEE 802.3 Annex 28B, Tables 28B-2 and 28B-3, respectively. Pause resolution status is reported in the **Ethernet PHY Control (EPHYCTL)**register.<br>Value Description<br>0 MAC PAUSE not implemented<br>1 MAC PAUSE implemented. Advertise that the MAC has implemented both the optional MAC control sub-layer and the pause function as specified in clause 31 and annex 31B of 802.3u. |
| 9 | 100BT4 | RO | 0 | 100Base-T4 Support<br>Value Description<br>0 100Base-T4 not supported by the internal PHY.<br>1 100Base-T4 is supported by the internal PHY. |
| 8 | 100BTXFD | RW | 1 | 100Base-TX Full Duplex Support<br>Value Description<br>0 100Base-TX Full Duplex not supported by the internal PHY.<br>1 100Base-TX Full Duplex is supported by the internal PHY. |
| 7 | 100BTX | RW | 1 | 100Base-TX Support<br>Value Description<br>0 100Base-TX not supported by the internal PHY.<br>1 100Base-TX is supported by the internal PHY. |
| 6 | 10BTFD | RW | 1 | 10Base-T Full Duplex Support<br>Value Description<br>0 10Base-T Full Duplex not supported by the internal PHY.<br>1 10Base-T Full Duplex is supported by the internal PHY. |
| 5 | 10BT | RW | 1 | 10Base-T Support<br>Value Description<br>0 10Base-T not supported by the internal PHY.<br>1 10Base-T is supported by the internal PHY. |
| 4:0 | SELECT | RW | 0x1 | Protocol Selection<br>These bits contain the binary encoded protocol selector supported by this port. 0x1 indicates that this device supports IEEE 802.3u. |

## Register 77: Ethernet PHY Auto-Negotiation Link Partner Ability - MR5 (EPHYANLPA), address 0x005

This register contains the advertised abilities of the Link Partner as received during Auto-Negotiation. The content changes after the successful auto-negotiation if next-pages are supported.

Ethernet PHY Auto-Negotiation Link Partner Ability - MR5 (EPHYANLPA)

Base n/a
Address 0x005
Type RO, reset 0x0000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NP | ACK | RF | reserved | ASMDUP | PAUSE | 100BT4 | 100BTXFD | 100BTX | 10BTFD | 10BT | SELECT | | | | |
| Type RO | RO | RO | RO | RW | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | NP | RO | 0 | Next Page Indication<br>Value Description<br>0 Link Partner does not desire Next Page Transfer.<br>1 Link Partner desires Next Page Transfer. |
| 14 | ACK | RO | 0 | Acknowledge<br>Value Description<br>0 Not acknowledged. The Auto-Negotiation state machine will automatically control the this bit based on the incoming FLP bursts.<br>1 Link Partner acknowledges reception of the ability data word. |
| 13 | RF | RO | 0 | Remote Fault<br>Value Description<br>0 No remote fault indicated by link partner.<br>1 Remote fault indicated by link partner. |
| 12 | reserved | RO | 0x0 | 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. |
| 11 | ASMDUP | RW | 0 | Asymmetric PAUSE<br>Value Description<br>0 Asymmetric pause is not supported by the Link Partner.<br>1 Asymmetric pause is supported by the Link Partner. |
| 10 | PAUSE | RO | 0 | PAUSE<br>Value Description<br>0 Pause function is not supported by the Link Partner.<br>1 Pause function is supported by the Link Partner. |

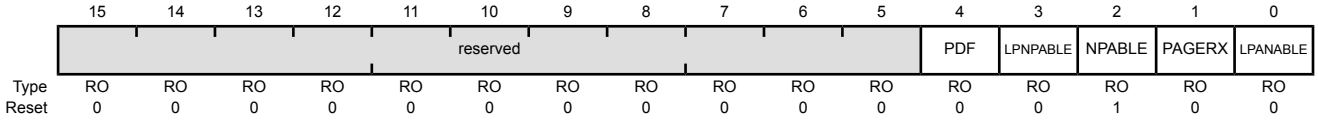
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | 100BT4 | RO | 0 | 100Base-T4 Support<br><br>Value Description<br>0 100Base-T4 is not supported by the Link Partner.<br>1 100Base-T4 is supported by the Link Partner. |
| 8 | 100BTXFD | RO | 0 | 100Base-TX Full Duplex Support<br><br>Value Description<br>0 100Base-TX Full Duplex is not supported by the Link Partner.<br>1 100Base-TX Full Duplex is supported by the Link Partner. |
| 7 | 100BTX | RO | 0 | 100Base-TX Support<br><br>Value Description<br>0 100Base-TX is not supported by the Link Partner.<br>1 100Base-TX is supported by the Link Partner. |
| 6 | 10BTFD | RO | 0 | 10Base-T Full Duplex Support<br><br>Value Description<br>0 10Base-T Full Duplex is not supported by the Link Partner.<br>1 10Base-T Full Duplex is supported by the Link Partner. |
| 5 | 10BT | RO | 0 | 10Base-T Support<br><br>Value Description<br>0 10Base-T is not supported by the Link Partner.<br>1 10Base-T is supported by the Link Partner |
| 4:0 | SELECT | RO | 0x1 | Protocol Selection<br>Link Partner's binary encoded protocol selector. |

## Register 78: Ethernet PHY Auto-Negotiation Expansion - MR6 (EPHYANER), address 0x006

This register contains additional local device and link partner status information.

Ethernet PHY Auto-Negotiation Expansion - MR6 (EPHYANER)

Base n/a
Address 0x006
Type RO, reset 0x0004



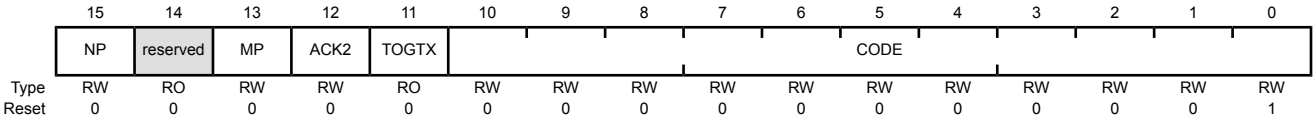

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:5 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | PDF | RO | 0 | Parallel Detection Fault<br>Value Description<br>0 A fault has not been detected.<br>1 A fault has been detected via the Parallel Detection function. |
| 3 | LPNPABLE | RO | 0 | Link Partner Next Page Able<br>Value Description<br>0 Link Partner does not support next page.<br>1 Link Partner does support next page. |
| 2 | NPABLE | RO | 1 | Next Page Able<br>Value Description<br>0 Indicates local device is not able to send additional next pages.<br>1 Indicates local device is able to send additional next pages |
| 1 | PAGERX | RO | 0 | Link Code Word Page Received<br>Value Description<br>0 Link Code Word has not been received.<br>1 Link Code Word has been received. This bit is cleared when read. |
| 0 | LPANABLE | RO | 0 | Link Partner Auto-Negotiation Able<br>Value Description<br>0 Indicates that the Link Partner does not support Auto-Negotiation.<br>1 Indicates that the Link Partner supports Auto-Negotiation. |

## Register 79: Ethernet PHY Auto-Negotiation Next Page TX - MR7 (EPHYANNPTR), address 0x007

This register contains the next page information sent by this device to its link partner during Auto-Negotiation.

Ethernet PHY Auto-Negotiation Next Page TX - MR7 (EPHYANNPTR)

Base n/a
Address 0x007
Type RW, reset 0x2001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | NP | RW | 0 | Next Page Indication<br><br>Value Description<br>0 No other next-page transfer desired.<br>1 Another next page desired. |
| 14 | reserved | RO | 0x0 | 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. |
| 13 | MP | RW | 0 | Message Page<br><br>Value Description<br>0 Unformatted Page.<br>1 Message Page. |
| 12 | ACK2 | RW | 0 | Acknowledge 2<br><br>Value Description<br>0 Cannot comply with message.<br>1 Can comply with message. |
| 11 | TOGTX | RO | 0 | Toggle<br><br>Toggle is used by the arbitration function within auto-negotiation to synchronize with the Link Partner during next page exchange. This bit always takes the opposite value of the toggle bit in the previously exchanged Link Code Word.<br><br>Value Description<br>0 Value of toggle bit in previously transmitted Link Code Word was 1.<br>1 Value of toggle bit in previously transmitted Link Code Word was 0. |

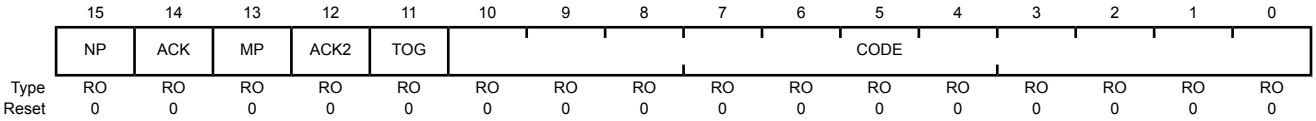
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10:0 | CODE | RW | 0x1 | Code<br>This field represents the code field of the next page transmission. If the MP bit is set (bit 13 of this register), then the code is interpreted as a Message Page, as defined in annex 28C of IEEE 802.3u.<br>Otherwise, the code is interpreted as an Unformatted Page, and the interpretation is application specific.<br>The default value of the CODE field represents a Null Page as defined in Annex 28C of IEEE 802.3u. |

## Register 80: Ethernet PHY Auto-Negotiation Link Partner Ability Next Page - MR8 (EPHYANLNPTR), address 0x008

This register contains the next page information sent by this device to its Link Partner during Auto-Negotiation.

Ethernet PHY Auto-Negotiation Link Partner Ability Next Page - MR8 (EPHYANLNPTR)

Base n/a
Address 0x008
Type RO, reset 0x0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | NP | RO | 0 | Next Page Indication<br><br>Value Description<br>0 No other next page transfer desired.<br>1 Another next page desired. |
| 14 | ACK | RO | 0 | Acknowledge<br><br>The auto-negotiation state machine automatically controls this bit based on the incoming fast-link pulse (FLP) bursts. Software should not attempt to write to this bit.<br><br>Value Description<br>0 Not acknowledged.<br>1 Link Partner acknowledges reception of the ability data word. |
| 13 | MP | RO | 0 | Message Page<br><br>Value Description<br>0 Unformatted Page.<br>1 Message Page. |
| 12 | ACK2 | RO | 0 | Acknowledge 2<br><br>Acknowledge2 is used by the next page function to indicate that Local Device has the ability to comply with the message received.<br><br>Value Description<br>0 Link Partner cannot comply to next-page message.<br>1 Link Partner has the ability to comply to next-page message. |

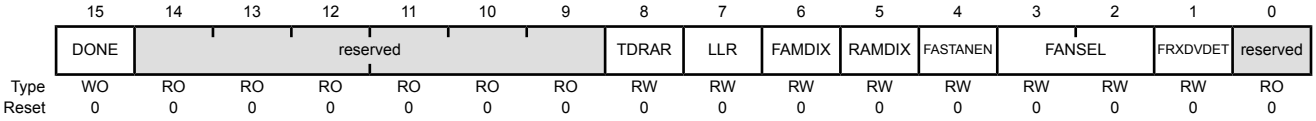
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | TOG | RO | 0 | Toggle<br>Toggle is used by the Arbitration function within Auto-Negotiation to synchronize with the Link Partner during Next Page exchange. This bit always takes the opposite value of the Toggle bit in the previously exchanged Link Code Word<br><br>Value Description<br>0 Value of toggle bit in previously transmitted Link Code Word was 1.<br>1 Value of toggle bit in previously transmitted Link Code Word was 0. |
| 10:0 | CODE | RO | 0x0 | Code<br>This field represents the code field of the next page transmission. If the MP bit is set (bit 13 of this RW register), then the code is interpreted as a Message Page, as defined in annex 28C of IEEE 802.3u. Otherwise, the code is interpreted as an Unformatted Page, and the interpretation is application specific.<br>The default value of the CODE represents a Null Page as defined in Annex 28C of IEEE 802.3u. |

## Register 81: Ethernet PHY Configuration 1 - MR9 (EPHYCFG1), address 0x009

This register configuration for the Ethernet PHY. These configuration values are programmed by the system processor after a POR. The DONE bit in the **EPHYCFG1** register is set when configuration is complete. This register is used when the user requires a configuration different from what is provided in the **EMACPC** register.

Ethernet PHY Configuration 1 - MR9 (EPHYCFG1)

Base n/a
Address 0x009
Type RW, reset 0x0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | DONE | WO | 0 | Configuration Done<br>This bit reads as a zero. The application must write a one to this bit to terminate the PHYHOLD mode set in the **EMACPC** register and wake up the EPHY.<br>Value Description<br>0 Configuration process is not complete.<br>1 Configuration process is complete, and the PHY can continue and complete its internal reset sequence. |
| 14:9 | reserved | RO | 0 | 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. |
| 8 | TDRAR | RW | 0 | TDR Auto-Run at Link Down<br>Value Description<br>0 Disable automatic execution of TDR.<br>1 Enable execution of TDR procedure after link down event. |
| 7 | LLR | RW | 0 | Link Loss Recovery<br>Value Description<br>0 Normal Link Loss operation<br>Link status goes down approximately 250 µs from signal loss.<br>1 Enable Link Loss Recovery mechanism<br>This mode allows recovery from short interference and continues to hold the link up for a period of an additional few µs until the short interference is gone and the signal is OK. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | FAMDIX | RW | 0 | Fast Auto MDI/MDIX<br>If both link partners are configured to work in Force 100Base-TX mode (auto-negotiation is disabled), this mode enables automatic MDI/MDIX resolution in a short time.<br>Value Description<br>0 Normal Auto MDI/MDIX mode.<br>1 Enable Fast Auto MDI/MDIX mode. |
| 5 | RAMDIX | RW | 0 | Robust Auto MDI/MDIX<br>If link partners are configured to operational modes that are not supported by normal Auto MDI/MDIX mode (like Auto-Neg versus Force 100Base-TX or Force 100Base-TX versus Force 100Base-TX), this Robust Auto MDI/MDIX mode allows MDI/MDIX resolution and prevents deadlock.<br>Value Description<br>0 Normal Auto MDI/MDIX mode.<br>1 Enable Robust Auto MDI/MDIX resolution. |
| 4 | FASTANEN | RW | 0 | Fast Auto Negotiation Enable<br>Adjusting these bits reduces the time it takes to auto-negotiate between two PHYs.<br>**Note:** When using this option care must be taken to maintain proper operation of the system. While shortening these timer intervals may not cause problems in normal operation, there are certain situations where this may lead to problems.<br>Value Description<br>0 Disable Fast Auto-Negotiation mode<br>The PHY auto-negotiates using the normal timer setting<br>1 Enable Fast Auto-Negotiation mode<br>The PHY auto-negotiates using the timer setting according to the FANSEL bits (bits [3:2] of this register) |

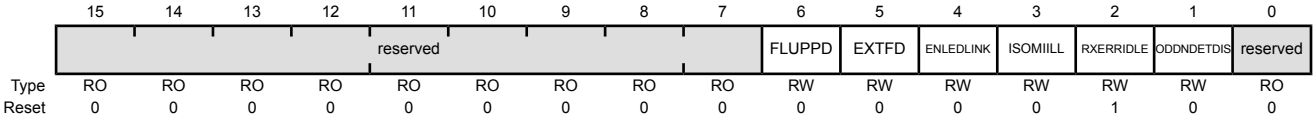
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | FANSEL | RW | 0 | Fast Auto-Negotiation Select Configuration<br>This bit is preconfigured at reset by the **EMACPC** register. For custom configuration see “Custom Configuration” on page 1653.<br>Adjusting these bits reduces the time it takes to auto-negotiate between two PHYs. In Fast AN mode, both PHYs should be configured to the same configuration. These two bits define the duration for each state of the auto-negotiation process according to the table above. The new duration time must be enabled by setting bit 4 (FASTANEN) of this register.<br>**Note:** Using this mode in cases where both link partners are not configured to the same fast auto-negotiation configuration might produce scenarios with unexpected behavior.<br>Value Description<br>0x0 Break Link Timer: 80 ms<br>Link Fail Inhibit Timer: 50 ms<br>Auto-Negate Wait Timer: 35 ms<br>0x1 Break Link Timer: 120 ms<br>Link Fail Inhibit Timer:75 ms<br>Auto-Negate Wait Timer: 50 ms<br>0x2 Break Link Timer: 240 ms<br>Link Fail Inhibit Timer:150 ms<br>Auto-Negate Wait Timer: 100 ms<br>0x3 reserved |
| 1 | FRXDVDET | RW | 0 | FAST RXDV Detection<br>Enabling this feature allows the PHY to pass data to the MII interface earlier. This bit is set to disabled if using the **EMACPC** register bits to program the PHY configuration.<br>Value Description<br>0 Disable fast RXDV detection. The PHY operates in normal mode where an internally asserted RXDV signal is sent to the MII interface after detection of /J/K/.<br>1 An internal RXDV signal is asserted high on receive packet due to detection of /J/ symbol only. Fast RXDV detection allows the PHY to pass data to the MII interface earlier.<br>If a consecutive /K/ does not appear, RXERR is generated. |
| 0 | reserved | RO | 0 | 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. |

## Register 82: Ethernet PHY Configuration 2 - MR10 (EPHYCFG2), address 0x00A

Fields in this register are used to configure the Ethernet PHY. These configuration values are programmed by the system processor after a POR. The DONE bit in the **EPHYCFG1** register is set when configuration is complete. This register is used when the user requires a configuration different from what is provided in the **EMACPC** register.

Ethernet PHY Configuration 2 - MR10 (EPHYCFG2)

Base n/a
Address 0x00A
Type RW, reset 0x0004




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:7 | reserved | RO | 0x0 | 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. |
| 6 | FLUPPD | RW | 0x0 | Fast Link-Up in Parallel Detect Mode<br>In fast-auto MDI-X and in robust-auto MDI-X modes (bits 6 and 5 in register **EPHYCFG1**), this bit is automatically set.<br>Value Description<br>0 Normal Parallel Detection link establishment<br>1 Enable Fast Link-Up time During Parallel Detection |
| 5 | EXTFD | RW | 0 | Extended Full-Duplex Ability<br>Encodes the type of PHY attached.<br>Value Description<br>0 Disable extended full-duplex ability. Decision to work in full-duplex or half-duplex mode follows IEEE specification.<br>1 Force full-duplex while working with link partner in forced 100B-TX. When the PHY is set to Auto-Negotiation or Force 100B-TX and the link partner is operated in Force 100B-TX, the link is always full duplex |
| 4 | ENLEDLINK | RW | 0 | Enhanced LED Functionality<br>Value Description<br>0 LED Link is ON when link is established.<br>1 LED Link is ON only when link is established in 100B-TX Full Duplex mode. |

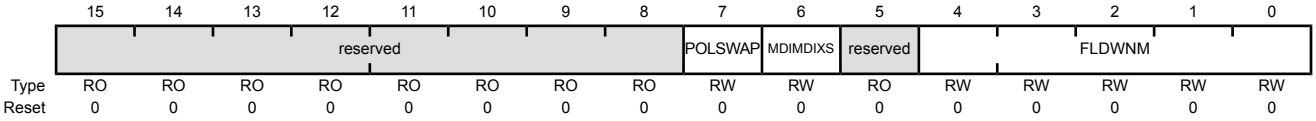
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | ISOMIILL | RW | 0 | Isolate MII outputs when Enhanced Link is not Achievable<br><br>Value Description<br>0 Normal MII outputs operation<br>1 When no link established in 100B-TX and Full Duplex conditions, MII outputs are tied low. |
| 2 | RXERRIDLE | RW | 1 | Detection of Receive Symbol Error During IDLE State<br><br>Value Description<br>0 Disable detection of Receive symbol error during IDLE state.<br>1 Enable detection of Receive symbol error during IDLE state. |
| 1 | ODDNDETDIS | RW | 0 | Detection of Transmit Error<br><br>Value Description<br>0 Enable detection of deassertion of the internal transmit enable on an odd-nibble boundary. In this case the internal transmit enable is extended by one additional transmit clock cycle and behaves as if the internal transmit error was asserted during that additional cycle.<br>1 Disable detection of internal transmit error in odd-nibble boundary. |
| 0 | reserved | RO | 0 | 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. |

## Register 83: Ethernet PHY Configuration 3 - MR11 (EPHYCFG3), address 0x00B

Fields in this register are used to configure the Ethernet PHY. These configuration values are programmed by the system processor after a POR. The DONE bit in the **EPHYCFG1** register is set when configuration is complete. This register is used when the user requires a configuration different from what is provided in the **EMACPC** register.

Ethernet PHY Configuration 3 - MR11 (EPHYCFG3)

Base n/a
Address 0x00B
Type RW, reset 0x0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:8 | reserved | RO | 0x0 | 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. |
| 7 | POLSWAP | RW | 0 | Polarity Swap<br><br>Value Description<br>0 MDI pairs normal (Receive on EN0RXIN/EN0RXIP pair, Transmit on EN0TXON/EN0TXOP pair)<br>1 Inverted polarity on both pairs.<br><br>To enable the port mirror function, set this bit and bit 6 (MDIMDIXS) to 1. |
| 6 | MDIMDIXS | RW | 0 | MDI/MDIX Swap<br><br>Value Description<br>0 MDI pairs normal (Receive on EN0RXIN/EN0RXIP pair, Transmit on EN0TXON/EN0TXOP pair)<br>1 Swap MDI pairs (Receive on EN0TXON/EN0TXOP pair, Transmit on EN0RXIN/EN0RXIP pair)<br><br>To enable the port mirroring function, set bit 7 and this bit to 1. |
| 5 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4:0 | FLDWNM | RW | 0x0 | Fast Link Down Modes<br>The Fast Link Down function is an OR of the following five options. The application can enable combinations of these conditions.<br>Bit 4: Drop the link due to descrambler sync loss. If this bit is enabled, the link is dropped when the receiver loses synchronization of the transmitter.<br>Bit 3: Drop the link based on RX Error count of the MII interface. When a predefined number of 32 RX Error occurrences in a 10µs interval is reached, the link is dropped.<br>Bit 2: Drop the link based on MLT3 Errors count (Violation of the MLT3 coding in the DSP output). When a predefined number of 20 MLT3 Error occurrences in a 10µs interval is reached, the link is dropped.<br>Bit 1: Drop the link based on Low SNR Threshold. When a predefined number of 20 Threshold crossing occurrences in a 10µs interval is reached, the link is dropped.<br>Bit 0: Drop the link based on Signal/Energy loss indication – When the Energy detector indicates Energy Loss, the link is dropped. Typical reaction time is 10µs. |

## Register 84: Ethernet PHY Register Control - MR13 (EPHYREGCTL), address 0x00D

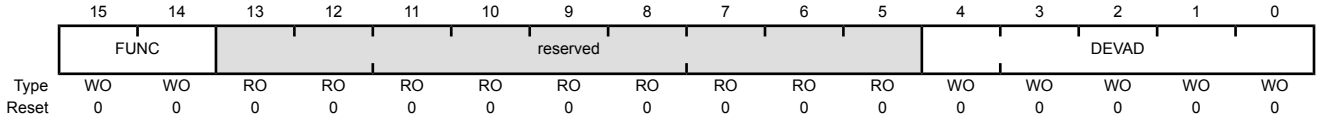
**EPHYREGCTL** (0x00D) and **EPHYADDAR** (0x00E) registers allow read/write access to the extended register set using indirect addressing. The modes for the FUNC field are as follows:

- **EPHYREGCTL[15:14]** = 0x0: A write to **EPHYADDAR** modifies the extended register set address register. This address register must be initialized in order to access any of the registers within the extended register set.
- **EPHYREGCTL[15:14]** = 0x1: A read/write to **EPHYADDAR** operates on the register within the extended register set selected (pointed to) by the value in the address register. The address register contents (pointer) remain unchanged.
- **EPHYREGCTL[15:14]** = 0x2: A read/write to **EPHYADDAR** operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for both reads and writes, the value in the address register is incremented.
- **EPHYREGCTL[15:14]** = 0x3: A read/write to **EPHYADDAR** operates on the register within the extended register set selected (pointed to) by the value in the address register. After that access is complete, for write accesses only, the value in the address register is incremented. For read accesses, the value of the address register remains unchanged.

This register is the MDIO Manageable Device (MMD) access control. In general, register **EPHYREGCTL [4:0]** is the device address, DEVAD, that directs any accesses of **EPHYADDAR** (0x00E) register to the appropriate MMD. It also contains selection bits for auto-increment of the data register. This register contains the device address to be written to access the extended registers. Write 0x1F into bits [4:0] of this register. It also contains selection bits [15:14] for the address auto-increment mode of **EPHYADDAR**.

Ethernet PHY Register Control - MR13 (EPHYREGCTL)

Base n/a
Address 0x00D
Type WO, reset 0x0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | FUNC | WO | 0 | Function<br>Value Description<br>0x0 Address<br>0x1 Data, no post increment<br>0x2 Data, post increment on read and write<br>0x3 Data, post increment on write only |
| 13:5 | reserved | RO | 0 | 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. |

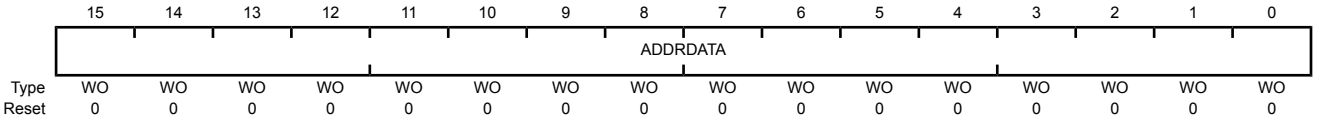
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4:0 | DEVAD | WO | 0x0 | Device Address<br>In general, these bits [4:0] are the device address DEVAD that directs any accesses of **EPHYADDAR** register (0x00E) to the appropriate MMD. The PHY uses the vendor specific DEVAD [4:0] = 0x1F for accesses. All accesses through registers **EPHYREGCR** and **EPHYADDAR** should use this DEVAD. Transactions with other DEVAD are ignored. |

## Register 85: Ethernet PHY Address or Data - MR14 (EPHYADDAR), address 0x00E

This register is the address/data MMD register. It is used in conjunction with **EPHYREGCTL** register (PHY offset 0x00D) to provide the access by an indirect read/write mechanism to the extended register set.

Ethernet PHY Address or Data - MR14 (EPHYADDAR)

Base n/a
Address 0x00E
Type WO, reset 0x0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:0 | ADDRDATA | WO | 0x0000 | Address or Data<br>If the **EPHYREGCTL** register bits [15:14] = 0x0, this register holds the MDIO Manageable Device (MMD) DEVAD's address register, otherwise holds the MMD DEVAD's data register. See **EPHYREGCTL** register for more information about DEVAD field. |

## Register 86: Ethernet PHY Status - MR16 (EPHYSTS), address 0x010

This register provides quick access to commonly accessed PHY control status and general information.

Ethernet PHY Status - MR16 (EPHYSTS)

Base n/a
Address 0x010
Type RO, reset 0x0002

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | MDIXM | RXLERR | POLSTAT | FCSL | SD | DL | PAGERX | MIIREQ | RF | JD | ANS | MIILB | DUPLEX | SPEED | LINK |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | reserved | RO | 0 | 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. |
| 14 | MDIXM | RO | 0 | MDI-X Mode<br>This is a read-only status as reported by the Auto-Negation state machine:<br>This bit is affected by the settings of the MDIXEN and FORCEMDIX bits in the **EPHYCTL** register. When MDIX is enabled, but not forced, this bit updates dynamically as the Auto-MDIX algorithm swaps between MDI and MDI-X configurations.<br>Value Description<br>0 MDI pairs normal (Receive on TPRD pair, Transmit on TPTD pair)<br>1 MDI pairs swapped (Receive on TPTD pair, Transmit on TPRD pair) |
| 13 | RXLERR | RO | 0 | Receive Error Latch<br>Value Description<br>0 No receive error event has occurred.<br>1 Receive error event has occurred since last read of **EPHYRXERCNT** register (PHY offset 0x015).<br>This bit is cleared on a read of the **EPHYRXERCNT** register. |
| 12 | POLSTAT | RO | 0 | Polarity Status<br>Value Description<br>0 Correct Polarity detected.<br>1 Inverted Polarity detected.<br>This bit is a duplication of bit 4 (POLSTAT) in the **EPHY10BTSC** register (PHY offset 0x01A). This bit is cleared upon a read of the **EPHY10BTSC** register, but not on a read of the **EPHYSTS** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | FCSL | RO | 0 | False Carrier Sense Latch<br><br>Value Description<br>0 No False Carrier event has occurred.<br>1 False Carrier event has occurred since last read of **EPHYFCSCR** register (0x014).<br><br>This bit is cleared on a read of the **EPHYFCSR** register. |
| 10 | SD | RO | 0 | Signal Detect<br>This bit displays the active high 100Base-TX unconditional Signal Detect indication from the PMD (Physical Layer Medium Dependent).<br>This bit is latched low and held until it is read, based upon the occurrence of the corresponding event. |
| 9 | DL | RO | 0 | Descrambler Lock<br>This bit displays the active high 100Base-TX Descrambler Lock indication from PMD.<br>This bit is latched low and held until it is read, based upon the occurrence of the corresponding event. |
| 8 | PAGERX | RO | 0 | Link Code Page Received<br><br>Value Description<br>0 Link Code Word Page has not been received.<br>1 A new Link Code Word Page has been received. This is a duplicate of Page Received (bit 1) in the **EPHYANER** register and it is cleared on read of the **EPHYANER** register (0x006).<br><br>This bit is not cleared upon a read of the **EPHYSTS** register. |
| 7 | MIIREQ | RO | 0 | MII Interrupt Pending<br><br>Value Description<br>0 No interrupt pending<br>1 Indicates that an internal interrupt is pending. Interrupt source can be determined by reading the **EPHYMISR1** Register (PHY offset 0x012). Reading the **EPHYMISR1** clears this Interrupt bit indication. |
| 6 | RF | RO | 0 | Remote Fault<br><br>Value Description<br>0 No remote fault condition detected.<br>1 Remote Fault condition detected. Criteria for a fault is when there is notification from Link Partner of Remote Fault via Auto-Negotiation. Cleared on read of **EPHYBMSR** register (PHY offset 0x001) or by reset. |

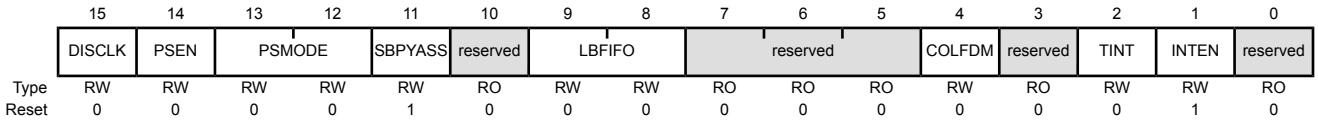
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | JD | RO | 0 | Jabber Detect<br>This bit will not be cleared upon a read of the **EPHYSTS** register.<br><br>Value Description<br>0 No Jabber.<br>1 Jabber condition detected. This bit has meaning only in 10 Mb/s mode. This bit is a duplicate of the Jabber Detect bit in the **EPHYBMSR** register (PHY offset 0x001). |
| 4 | ANS | RO | 0 | Auto-Negotiation Status<br><br>Value Description<br>0 Auto-Negotiation not complete.<br>1 Auto-Negotiation complete. |
| 3 | MIILB | RO | 0 | MII Loopback Status<br><br>Value Description<br>0 Normal operation.<br>1 Loopback active (enabled). |
| 2 | DUPLEX | RO | 0 | Duplex Status<br><br>Value Description<br>0 Half Duplex Mode<br>1 Full Duplex Mode<br><br>This bit indicates duplex status and is determined from Auto-Negotiation or Forced Modes. Therefore, it is only valid if Auto-Negotiation is enabled and complete and there is a valid link or if Auto-Negotiation is disabled and there is a valid link. |
| 1 | SPEED | RO | 1 | Speed Status<br><br>Value Description<br>0 100 Mb/s mode.<br>1 10 Mb/s mode.<br><br>This bit indicates the status of the speed and is determined from Auto-Negotiation or Forced Modes. It is only valid if Auto-Negotiation is enabled and complete and there is a valid link or if Auto-Negotiation is disabled and there is a valid link. |
| 0 | LINK | RO | 0 | Link Status<br><br>Value Description<br>0 Link is not established.<br>1 Valid link is established (for either 10 or 100 Mb/s operation). This bit is a duplicate of the Link Status bit in the **EPHYBMSR** register (PHY offset 0x001).<br><br>This bit is not cleared upon a read of the **EPHYSTS** register. |

## Register 87: Ethernet PHY Specific Control- MR17 (EPHYSCR), address 0x011

This register implements the PHY Specific Control register. This register allows access to general functionality inside the PHY to enable operation in reduced power modes and control the interrupt mechanism.

Ethernet PHY Specific Control- MR17 (EPHYSCR)

Base n/a
Address 0x011
Type RW, reset 0x0103



| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DISCLK | PSEN | PSMODE | | SBPYASS | reserved | LBFIFO | | reserved | | | COLFDM | reserved | TINT | INTEN | reserved |
| Type | RW | RW | RW | RW | RW | RO | RW | RW | RO | RO | RO | RW | RO | RW | RW | RO |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | DISCLK | RW | 0 | Disable CLK |

| Value | Description |
|---|---|
| 0 | Normal mode of operation |
| 1 | Disable internal clocks |

Clocks can be disabled only in IEEE power down mode.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 14 | PSEN | RW | 0 | Power Saving Modes Enable |

| Value | Description |
|---|---|
| 0 | Normal mode of operation |
| 1 | Enable power saving modes |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13:12 | PSMODE | RW | 0 | Power Saving Modes |

| Value | Description |
|---|---|
| 0x0 | Normal: Normal operation mode. PHY is fully functional |
| 0x1 | IEEE Power Down<br>Low Power mode that shuts down all internal circuitry other than SMI functionality. Power could be dropped further by setting high DISCLK bit in this register and disabling internal clocks circuitries. |
| 0x2 | Active Sleep<br>Low Power Active Wake-On-LAN (WOL) mode that shuts down all internal circuitry other than SMI and energy detect functionality. In this mode the PHY sends NLP every 1.4 seconds to wake up the link-partner. Automatic power-up is done when link partner is detected. |
| 0x3 | Passive Sleep<br>Low Power WOL mode that shuts down all internal circuitry besides SMI and energy detect functionality. Automatic power-up is done when link partner is detected. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | SBPYASS | RW | 0x1 | Scrambler Bypass<br><br>Value Description<br>0 Scrambler bypass disabled.<br>1 Scrambler bypass enabled. |
| 10 | reserved | RO | 0 | 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. |
| 9:8 | LBFIFO | RW | 0 | Loopback FIFO Depth<br><br>Value Description<br>0x0 Four nibble FIFO<br>0x1 Five nibble FIFO<br>0x2 Six nibble FIFO<br>0x3 Eight nibble FIFO<br><br>This FIFO is used to adjust RX (recovered) clock rate to TX clock rate. FIFO depth must be set based on expected maximum packet size and clock accuracy. Default value sets to 5 nibbles. |
| 7:5 | reserved | RO | 0 | 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. |
| 4 | COLFDM | RW | 0 | Collision in Full-Duplex Mode<br><br>Value Description<br>0 Disable collision indication in full-duplex mode. Collision indication is active in half-duplex mode only.<br>1 Enable collision signaling in full-duplex mode. |
| 3 | reserved | RO | 0 | 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. |
| 2 | TINT | RW | 0 | Test Interrupt<br><br>Value Description<br>0 Do not generate interrupt<br>1 Generate an interrupt<br><br>Forces the PHY to generate an interrupt to facilitate interrupt testing. Interrupts will continue to be generated as long as this bit remains set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INTEN | RW | 1 | Interrupt Enable<br><br>Value Description<br>0 Disable event based interrupts<br>1 Enable event based interrupts<br><br>Enables interrupt dependent on the event enables in the **EPHYMISR** register (0x012). |
| 0 | reserved | RO | 0 | 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. |

## Register 88: Ethernet PHY MII Interrupt Status 1 - MR18 (EPHYMISR1), address 0x012

This register contains events status and enables for the interrupt function. If an event has occurred since the last read of this register, the corresponding status bit is set. If the corresponding enable bit in the register is set, an interrupt is generated if the event occurs. The INTEN bit (Bit 1) in the **EPHYSCR** register (0x011) must also be set to allow interrupts. The status indications in this register are set even if the interrupt is not enabled.

Ethernet PHY MII Interrupt Status 1 - MR18 (EPHYMISR1)

Base n/a
Address 0x012
Type RW, reset 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | LINKSTAT | SPEED | DUPLEXM | ANC | FCHF | RXHF | reserved | | LINKSTATEN | SPEEDEN | DUPLEXMEN | ANCEN | FCHFEN | RXHFEN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | reserved | RO | 0 | 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. |
| 13 | LINKSTAT | RO | 0 | Change of Link Status Interrupt<br>Reading this bit clears the interrupt and thus, the status bit.<br><br>Value Description<br>0 No change of link status.<br>1 Change of link status interrupt is pending. |
| 12 | SPEED | RO | 0 | Change of Speed Status Interrupt<br>Reading this bit clears the interrupt and thus, the status bit.<br><br>Value Description<br>0 No change of speed status.<br>1 Change of speed status interrupt is pending. |
| 11 | DUPLEXM | RO | 0 | Change of Duplex Status Interrupt<br>Reading this bit clears the interrupt and thus, the status bit.<br><br>Value Description<br>0 No change of duplex status.<br>1 Duplex status change interrupt is pending. |
| 10 | ANC | RO | 0 | Auto-Negotiation Complete Interrupt<br>Reading this bit clears the interrupt and thus, the status bit.<br><br>Value Description<br>0 No Auto-negotiation complete event is pending.<br>1 Auto-negotiation complete interrupt is pending. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | FCHF | RO | 0 | False Carrier Counter Half-Full Interrupt<br>Reading this bit clears the interrupt and thus, the status bit.<br><br>Value Description<br>0 False carrier counter half-full event is not pending.<br>1 False carrier counter (Register **EPHYFCSCR**, address 0x014) exceeds half-full and an interrupt is pending. |
| 8 | RXHF | RO | 0 | Receive Error Counter Half-Full Interrupt<br>Reading this bit clears the interrupt and thus, the status bit.<br><br>Value Description<br>0 False carrier counter half-full event is not pending.<br>1 Receive error counter (Register **EPHYRXERCNT**, address 0x015) exceeds half-full and an interrupt is pending. |
| 7:6 | reserved | RO | 0 | 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 | LINKSTATEN | RW | 0 | Link Status Interrupt Enable<br><br>Value Description<br>0 Change of link status interrupt disabled.<br>1 Enable interrupt on change of link status. |
| 4 | SPEEDEN | RW | 0 | Speed Change Interrupt Enable<br><br>Value Description<br>0 Change of speed status interrupt disabled.<br>1 Enable interrupt on change of speed status |
| 3 | DUPLEXMEN | RW | 0 | Duplex Status Interrupt Enable<br><br>Value Description<br>0 Change of duplex status interrupt disabled.<br>1 Enable interrupt on change of duplex status |
| 2 | ANCEN | RW | 0 | Auto-Negotiation Complete Interrupt Enable<br><br>Value Description<br>0 Auto-negotiation complete interrupt disabled.<br>1 Enable interrupt on Auto-negotiation complete event |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | FCHFEN | RW | 0 | False Carrier Counter Register half-full Interrupt Enable<br><br>Value Description<br>0 False Carrier Counter Register half-full interrupt disabled.<br>1 Enable interrupt on False Carrier Counter Register half-full event |
| 0 | RXHFEN | RW | 0 | Receive Error Counter Register Half-Full Event Interrupt<br><br>Value Description<br>0 Receive Error Counter Register half-full interrupt disabled.<br>1 Enable interrupt on Receive Error Counter Register half-full event |

## Register 89: Ethernet PHY MII Interrupt Status 2 - MR19 (EPHYMISR2), address 0x013

This register contains additional event status and enables for the interrupt function. If an event has occurred since the last read of this register, the corresponding status bit is set. If the corresponding enable bit in the register is set, an interrupt is generated if the event occurs. The INTEN bit (bit 1) in the **EPHYSCR** register (PHY offset 0x011) must also be set to allow interrupts. The status indications in this register are set even if the interrupt is not enabled.

Ethernet PHY MII Interrupt Status 2 - MR19 (EPHYMISR2)

Base n/a
Address 0x013
Type RW, reset 0x0000

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | ANERR | PAGERX | LBFIFO | MDICO | SLEEP | POLINT | JABBER | reserved | ANERREN | PAGERXEN | LBFIFOEN | MDICOEN | SLEEPEN | POLINTEN | JABBEREN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | reserved | RO | 0 | 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. |
| 14 | ANERR | RO | 0 | Auto-Negotiation Error Interrupt<br>Reading this bit clears the interrupt.<br><br>Value Description<br>0 No Auto-negotiation error event pending.<br>1 Auto-negotiation error interrupt is pending. |
| 13 | PAGERX | RO | 0 | Page Receive Interrupt<br>Reading this bit clears the interrupt.<br><br>Value Description<br>0 Page has not been received.<br>1 Page has been received. |
| 12 | LBFIFO | RO | 0 | Loopback FIFO Overflow/Underflow Event Interrupt<br>Reading this bit clears the interrupt.<br><br>Value Description<br>0 No FIFO Overflow/Underflow event pending.<br>1 FIFO Overflow/Underflow event interrupt pending. |
| 11 | MDICO | RO | 0 | MDI/MDIX Crossover Status Changed Interrupt<br>Reading this bit clears the interrupt.<br><br>Value Description<br>0 MDI crossover status has not changed.<br>1 MDI crossover status changed interrupt is pending. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | SLEEP | RO | 0 | Sleep Mode Event Interrupt<br>Reading this bit clears the interrupt.<br><br>Value Description<br>0 No sleep mode event pending.<br>1 Sleep Mode event interrupt is pending. |
| 9 | POLINT | RO | 0 | Polarity Changed Interrupt<br>Reading this bit clears the interrupt.<br><br>Value Description<br>0 No Data polarity event pending.<br>1 Data polarity changed interrupt pending. |
| 8 | JABBER | RO | 0 | Jabber Detect Event Interrupt<br>Reading this bit clears the interrupt.<br><br>Value Description<br>0 No Jabber detect event pending<br>1 Jabber detect event interrupt pending. |
| 7 | reserved | RO | 0 | 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. |
| 6 | ANERREN | RW | 0 | Auto-Negotiation Error Interrupt Enable<br><br>Value Description<br>0 Auto-Negotiation error event interrupt disabled.<br>1 Enable interrupt on Auto-Negotiation error event |
| 5 | PAGERXEN | RW | 0 | Page Receive Interrupt Enable<br><br>Value Description<br>0 Page Receive error event interrupt disabled.<br>1 Page receive event interrupt enabled. |
| 4 | LBFIFOEN | RW | 0 | Loopback FIFO Overflow/Underflow Interrupt Enable<br><br>Value Description<br>0 Loopback FIFO overflow/underflow event interrupt disabled<br>1 Loopback FIFO overflow/underflow event interrupt enabled. |

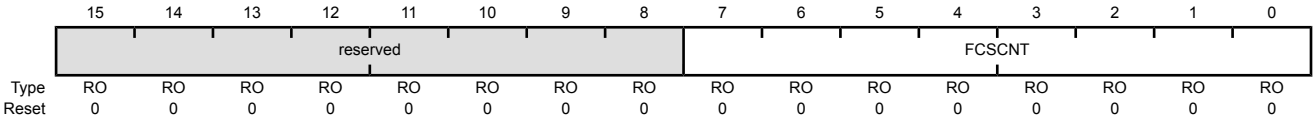
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | MDICOEN | RW | 0 | MDI/MDIX Crossover Status Changed Interrupt Enable<br>Value Description<br>0 MDI/MDIX Crossover Change Status interrupt disabled<br>1 MDI/MDIX Crossover Change Status interrupt enabled. |
| 2 | SLEEPEN | RW | 0 | Sleep Mode Event Interrupt Enable<br>Value Description<br>0 Sleep mode interrupt disabled.<br>1 Sleep Mode event interrupt enabled. |
| 1 | POLINTEN | RW | 0 | Polarity Changed Interrupt Enable<br>Value Description<br>0 Polarity Change interrupt is disabled.<br>1 Polarity change interrupt is enabled. |
| 0 | JABBEREN | RO | 0 | Jabber Detect Event Interrupt Enable<br>Value Description<br>0 Jabber detect interrupt disabled.<br>1 Jabber detect interrupt enabled. |

## Register 90: Ethernet PHY False Carrier Sense Counter - MR20 (EPHYFCSCR), address 0x014

This counter provides information required to implement the False Carriers attribute within the MAU managed object class of Clause 30 of the IEEE 802.3u specification.

Ethernet PHY False Carrier Sense Counter - MR20 (EPHYFCSCR)

Base n/a
Address 0x014
Type RO, reset 0x0000



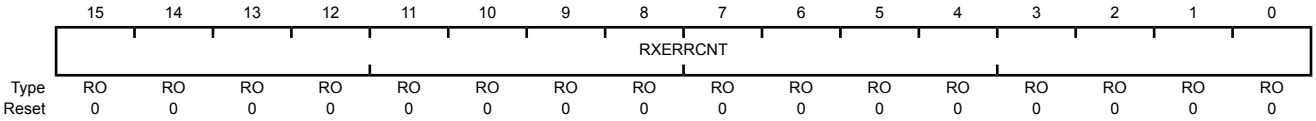

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:8 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | FCSCNT | RO | 0x00 | False Carrier Event Counter<br>This 8-bit counter increments on every false carrier event. This counter stops when it reaches its maximum count (0xFF). When the counter exceeds half full (0x7F), an interrupt event is generated. This register is cleared on read. |

## Register 91: Ethernet PHY Receive Error Count - MR21 (EPHYRXERCNT), address 0x015

This counter provides information required to implement the Symbol Error During Carrier attribute within the PHY managed object class of Clause 30 of the IEEE 802.3u specification.

Ethernet PHY Receive Error Count - MR21 (EPHYRXERCNT)

Base n/a
Address 0x015
Type RO, reset 0x0000.0000



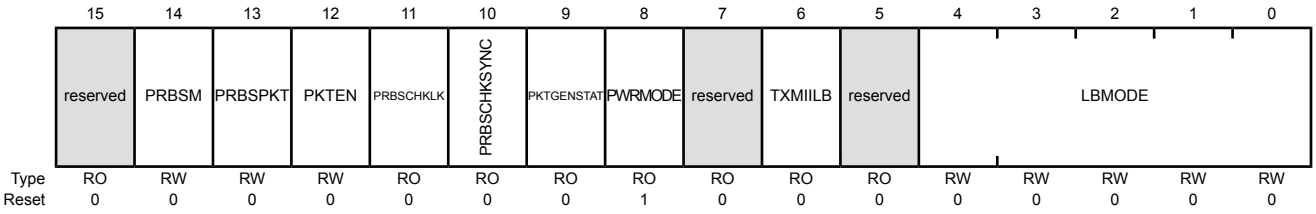

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:0 | RXERRCNT | RO | 0x0 | Receive Error Count<br>When a valid carrier is present (while EN0RXDV is active), and there is at least one occurrence of an invalid data symbol, this 16-bit counter increments for each receive error detected. The **EPHYRXERCNT** counter does not count in MII loopback mode. The counter stops when it reaches its maximum count of 0xFFFF. When the counter exceeds half-full (0x7FFF), an interrupt is generated. This register is cleared on read. |

## Register 92: Ethernet PHY BIST Control - MR22 (EPHYBISTCR), address 0x016

This register is used for Build-In Self Test (BIST) configuration. The BIST functionality provides Pseudo Random Bit Stream (PRBS) mechanism including packet generation generator and checker. Selection of the exact loopback point in the signal chain is also done in this register.

Ethernet PHY BIST Control - MR22 (EPHYBISTCR)

Base n/a
Address 0x016
Type RW, reset 0x0100




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | reserved | RO | 0 | 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. |
| 14 | PRBSM | RW | 0 | PRBS Single/Continuous Mode<br>Value Description<br>0 Single mode selected. When BIST Error Counter reaches its max value, the PRBS checker stops counting.<br>1 Continuous mode selected. When the PRBS counters reach maximum count value, the counter starts counting from zero again. |
| 13 | PRBSPKT | RW | 0 | Generated PRBS Packets<br>Value Description<br>0 When packet generator is enabled, generate a single packet with constant data. PRBS generation and check is disabled.<br>1 When packet generator is enabled, generate continuous packets with PRBS data. When packet generator is disabled, PRBS checker is still enabled. |
| 12 | PKTEN | RW | 0 | Packet Generation Enable<br>Value Description<br>0 Disable packet generator<br>1 Enable packet generation with PRBS data |
| 11 | PRBSCHKLK | RO | 0 | PRBS Checker Lock Indication<br>Value Description<br>0 PRBS checker is not locked<br>1 PRBS checker is locked and synchronized on received bit stream |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10 | PRBSCHKSYNC | RO | 0 | PRBS Checker Lock Sync Loss Indication<br>Value Description<br>0 PRBS checker has not lost synchronization.<br>1 PRBS checker has lost synchronization on received bit stream. This is an error indication. |
| 9 | PKTGENSTAT | RO | 0 | Packet Generator Status Indication<br>Value Description<br>0 Packet Generator is disabled.<br>1 Packet Generator is active and generate packets. |
| 8 | PWRMODE | RO | 1 | Power Mode Indication<br>Value Description<br>0 Indicates that the PHY is in one of the sleep modes, either active or passive.<br>1 Indicates that the PHY is in normal power mode. |
| 7 | reserved | RO | 0 | 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. |
| 6 | TXMIILB | RO | 0 | Transmit Data in MII Loopback Mode<br>Value Description<br>0 Data is not transmitted to the line in MII loopback<br>1 Enable transmission of the data from the MAC received on the TX pins to the line in parallel to the MII loopback to RX pins. This bit may be set only in MII Loopback mode, which is enabled by setting the MIILOOPBK bit in the **EPHYBMCR** register, offset EPHY.0x00. |
| 5 | reserved | RO | 0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4:0 | LBMODE | RW | 0 | Loopback Mode Select<br>The PHY provides several options for Loopback that test and verify various functional blocks within the PHY. |

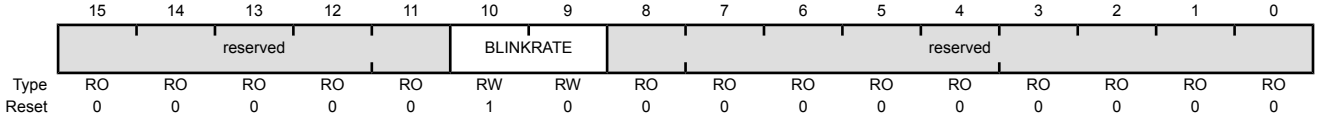
| Value | Description |
|---|---|
| 0x00 | reserved |
| 0x01 | Near-end loopback: PCS Input Loopback |
| 0x02 | Near-end loopback: PCS Output Loopback (In 100Base-TX only) |
| 0x03 | reserved |
| 0x04 | Near-end loopback: Digital Loopback |
| 0x05-0x07 | reserved |
| 0x08 | Near-end loopback: Analog Loopback (requires 100Ω termination) |
| 0x09-0x0F | reserved |
| 0x10 | Far-end Loopback: Reverse Loopback |

## Register 93: Ethernet PHY LED Control - MR24 (EPHYLEDCR), address 0x018

This register provides the ability to control the blink rate on the LED outputs.

Ethernet PHY LED Control - MR24 (EPHYLEDCR)

Base n/a
Address 0x018
Type RW, reset 0x0400



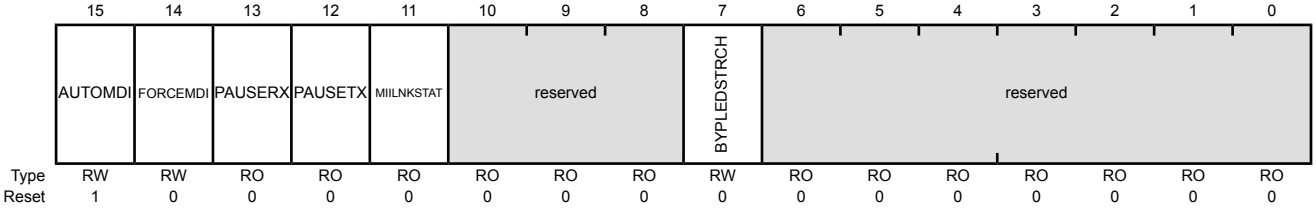

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:11 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10:9 | BLINKRATE | RW | 0x2 | LED Blinking Rate (ON/OFF duration):<br>Value Description<br>0x0 20 Hz (50 ms)<br>0x1 10 Hz (100 ms)<br>0x2 5 Hz (200 ms)<br>0x3 2 Hz (500 ms) |
| 8:0 | reserved | RO | 0 | 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. |

## Register 94: Ethernet PHY Control - MR25 (EPHYCTL), address 0x019

This register provides the ability to control and set general functionality inside the PHY.

Ethernet PHY Control - MR25 (EPHYCTL)

Base n/a
Address 0x019
Type RW, reset 0x8000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | AUTOMDI | RW | 1 | Auto-MDIX Enable |

| Value | Description |
|---|---|
| 0 | Disable Auto- negotiation, Auto-MDIX capability. |
| 1 | Enable Auto-negotiation, Auto-MDIX capability. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 14 | FORCEMDI | RW | 0 | Force MDIX |

| Value | Description |
|---|---|
| 0 | Normal operation. (Transmit on TPTD pair, Receive on TPRD pair) |
| 1 | Force MDI pairs to cross. (Receive on TPTD pair, Transmit on TPRD pair) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | PAUSERX | RO | 0 | Pause Receive Negotiated Status |

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Indicates that pause receive should be enabled in the MAC.<br>This bit is set based on bits [11:10] in the **EPHYANA** register and in the **EPHYANLPAR** register settings.<br>This function is enabled according to IEEE 802.3 Annex 28B Table 28B-3, Pause Resolution, only if the Auto-Negotiated Highest Common Denominator is a full-duplex technology. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | PAUSETX | RO | 0 | Pause Transmit Negotiated Status |

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Indicates that pause transmit should be enabled in the MAC.<br>This bit is set based on bits [11:10] in the **EPHYANA** register and in the **EPHYANLPAR** register settings.<br>This function is enabled according to IEEE 802.3 Annex 28B Table 28B-3, Pause Resolution, only if the Auto-Negotiated Highest Common Denominator is a full-duplex technology. |

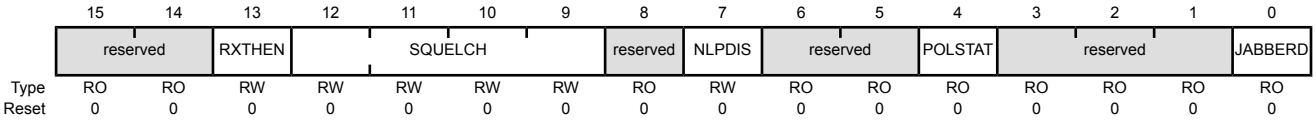
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | MIILNKSTAT | RO | 0 | MII Link Status<br>Value Description<br>0 No active link of 100BT full-duplex has been established using Auto-Negotiation.<br>1 100BT Full-duplex Link is active and was established using Auto-Negotiation. |
| 10:8 | reserved | RO | 0 | 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. |
| 7 | BYPLEDSTRCH | RW | 0 | Bypass LED Stretching<br>This bypasses LED stretching and allows the LEDs to reflect normal operation values.<br>Value Description<br>0 Normal LED operation.<br>1 Bypass LED stretching. |
| 6:0 | reserved | RO | 0 | 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. |

## Register 95: Ethernet PHY 10Base-T Status/Control - MR26 (EPHY10BTSC), address 0x01A

This register provides the ability to control and read status of the PHY's internal 10Base-T functionality.

Ethernet PHY 10Base-T Status/Control - MR26 (EPHY10BTSC)

Base n/a
Address 0x01A
Type RW, reset 0x0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:14 | reserved | RO | 0 | 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. |
| 13 | RXTHEN | RW | 0 | Lower Receiver Threshold Enable<br>Value Description<br>0 Normal 10Base-T operation.<br>1 Enable 10Base-T lower receiver threshold to allow operation with longer cables |
| 12:9 | SQUELCH | RW | 0x0 | Squelch Configuration<br>Used to set the Peak Squelch ON threshold for the 10Base-T receiver. Every value corresponds to a 50-mV increase. The squelch threshold range is from 200 mV to 600 mV. The default Squelch threshold is set to 200 mV.<br>Values 0xA to 0xF are reserved. |
| 8 | reserved | RO | 0 | 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. |
| 7 | NLPDIS | RW | 0 | Normal Link Pulse (NLP) Transmission Control<br>Value Description<br>0 Enable transmission of Normal Link Pulses (NLPs).<br>1 Disable transmission of Normal Link Pulses (NLPs). |
| 6:5 | reserved | RO | 0 | 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. |

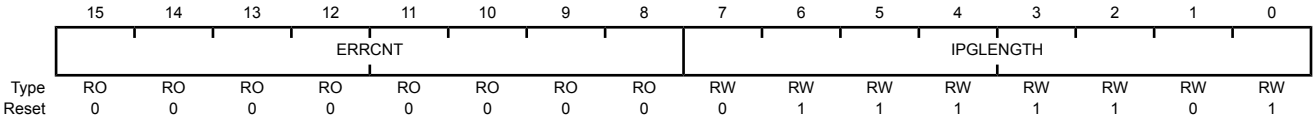
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | POLSTAT | RO | 0 | 10 Mb Polarity Status<br>This bit is a duplication of POLSTAT bit in the **EPHYSTS** register, offset 0x0010. Both bits are cleared on a read of the Ethernet PHY 10Base-T Status/Control (EPHY10BTSC), but not upon a read of the **EPHYSTS** register.<br>Value Description<br>0 Correct Polarity detected.<br>1 Inverted Polarity detected. |
| 3:1 | reserved | RO | 0 | 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. |
| 0 | JABBERD | RO | 0 | Jabber Disable<br>Note that this function is applicable only in 10Base-T.<br>Value Description<br>0 Jabber function enabled.<br>1 Jabber function disabled. |

## Register 96: Ethernet PHY BIST Control and Status 1 - MR27 (EPHYBICSR1), address 0x01B

This register provides the total number of error bytes that are received by the PRBS checker and defines the Inter-Packet Gap (IPG) for the packet generator.

Ethernet PHY BIST Control and Status 1 - MR27 (EPHYBICSR1)

Base n/a
Address 0x01B
Type RW, reset 0x007D



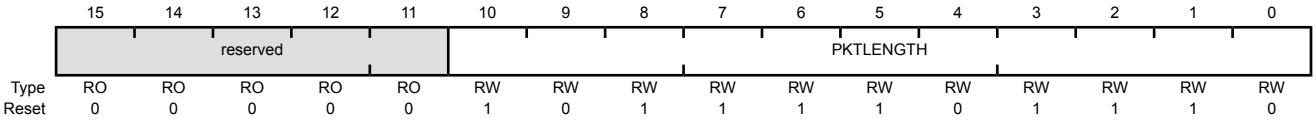

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:8 | ERRCNT | RO | 0x00 | BIST Error Count<br>This field holds the number of erroneous bytes that were received by the PRBS checker. The value in this register is locked when a write is done to bit 15.<br>When the PRBSM field in the **EPHYBISCR** register (0x0016) is set to zero, the count stops at 0xFF. See the **EPHYBISCR** register for further details. |
| 7:0 | IPGLENGTH | RW | 0x7D | BIST IPG Length<br>Inter Packet Gap (IPG) Length defines the size of the gap (in bytes) between any 2 successive packets generated by the BIST. Default value is 0x7D, which is equal to 125 bytes. |

## Register 97: Ethernet PHY BIST Control and Status 2 - MR28 (EPHYBICSR2), address 0x01C

This register allows programming the length of the generated packets in bytes for the BIST mechanism

Ethernet PHY BIST Control and Status 2 - MR28 (EPHYBICSR2)

Base n/a
Address 0x01C
Type RW, reset 0x05EE



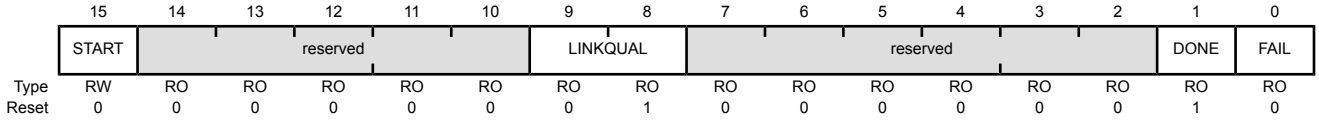

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:11 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10:0 | PKTLENGTH | RW | 0x5EE | BIST Packet Length<br>The value of this register defines the size (in bytes) of every packet that is generated by the BIST. Default value is 0x5EE, which is equal to 1518 bytes. |

## Register 98: Ethernet PHY Cable Diagnostic Control - MR30 (EPHYCDCR), address 0x01E

This register provides the ability to start cable diagnostics and monitor its status.

Ethernet PHY Cable Diagnostic Control - MR30 (EPHYCDCR)

Base n/a
Address 0x01E
Type RO, reset 0x0102



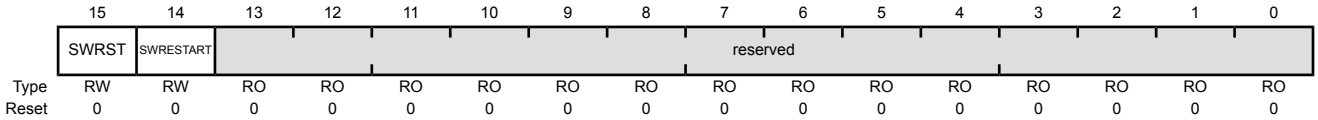

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | START | RW | 0 | Cable Diagnostic Process Start<br>Diagnostic Start bit is cleared with raise of Diagnostic Done indication.<br><br>Value Description<br>0 Cable Diagnostic is disabled<br>1 Start cable diagnostic measurement |
| 14:10 | reserved | RO | 0 | 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. |
| 9:8 | LINKQUAL | RO | 0x1 | Link Quality Indication<br>The value of these bits are valid only when link is active (when the LINK bit in the **EPHYSTS** register is a 0x1).<br><br>Value Description<br>0x0 reserved<br>0x1 Good Quality Link Indication<br>0x2 Mid- Quality Link Indication<br>0x3 Poor Quality Link Indication |
| 7:2 | reserved | RO | 0 | 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. |
| 1 | DONE | RO | 1 | Cable Diagnostic Process Done<br><br>Value Description<br>0 Diagnostic has not completed<br>1 Indicates that the cable measurement process completed |
| 0 | FAIL | RO | 0 | Cable Diagnostic Process Fail<br><br>Value Description<br>0 Diagnostic has not failed<br>1 Indicates that the cable measurement process failed |

## Register 99: Ethernet PHY Reset Control - MR31 (EPHYRCR), address 0x01F

This register allows the system to reset or restart the PHY by register access.

Ethernet PHY Reset Control - MR31 (EPHYRCR)

Base n/a
Address 0x01F
Type RW, reset 0x0000



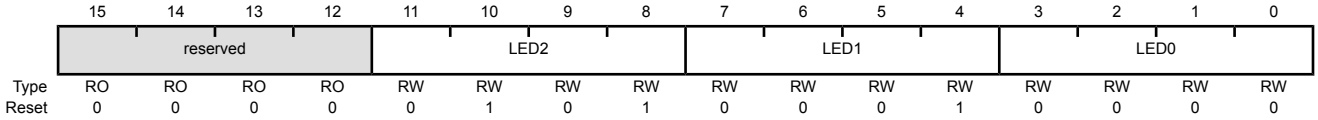

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SWRST | RW | 0 | Software Reset<br><br>Value Description<br>0 Normal Operation.<br>1 Soft reset. This mode resets the digital portion of the PHY and all of the registers. This bit self clears after completion. |
| 14 | SWRESTART | RW | 0 | Software Restart<br><br>Value Description<br>0 Normal Operation.<br>1 Restart PHY. This mode resets the digital portion of the PHY but no the registers. This bit self clears after completion of the restart. |
| 13:0 | reserved | RO | 0 | 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. |

## Register 100: Ethernet PHY LED Configuration - MR37 (EPHYLEDCFG), address 0x025

This register configures the LEDs provided in the PHY.

Ethernet PHY LED Configuration - MR37 (EPHYLEDCFG)

Base n/a
Address 0x025
Type RW, reset 0x0000.0510




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:12 | reserved | RO | 0 | 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. |
| 11:8 | LED2 | RW | 0x5 | LED2 Configuration<br>The following encodings are used to program the specific function desired for the LED.<br>Value Description<br>0x0 Link OK<br>0x1 RX/TX Activity<br>0x2 TX Activity<br>0x3 RX Activity<br>0x4 Collision<br>0x5 100-Base TX<br>0x6 10-Base TX<br>0x7 Full Duplex<br>0x8 Link OK/Blink on TX/RX Activity<br>0x9-0xF Reserved |
| 7:4 | LED1 | RW | 0x1 | LED1 Configuration<br>The following encodings are used to program the specific function desired for the LED.<br>Value Description<br>0x0 Link OK<br>0x1 RX/TX Activity<br>0x2 TX Activity<br>0x3 RX Activity<br>0x4 Collision<br>0x5 100-Base TX<br>0x6 10-Base TX<br>0x7 Full Duplex<br>0x8 Link OK/Blink on TX/RX Activity<br>0x9-0xF Reserved |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | LED0 | RW | 0x0 | LED0 Configuration<br>The following encodings are used to program the specific function desired for the LED. |

| Value | Description |
|---|---|
| 0x0 | Link OK |
| 0x1 | RX/TX Activity |
| 0x2 | TX Activity |
| 0x3 | RX Activity |
| 0x4 | Collision |
| 0x5 | 100-Base TX |
| 0x6 | 10-Base TX |
| 0x7 | Full Duplex |
| 0x8 | Link OK/Blink on TX/RX Activity |
| 0x9-0xF | Reserved |

# 25 Universal Serial Bus (USB) Controller

**Important:** The full USB chapter is under NDA. This chapter describes the module features at a high level. For a copy of the full NDA data sheet, follow the instructions in the *Non-Disclosure Agreement for the Tiva C Series TM4C129XKCZAD Microcontroller Data Sheet* (literature number SPMS425).

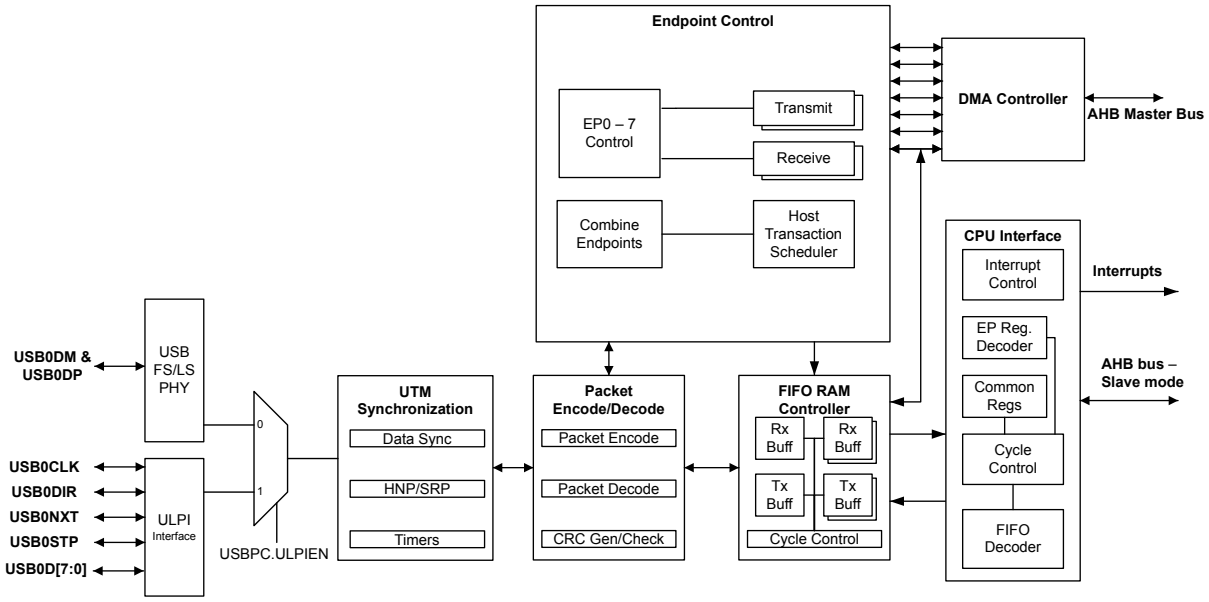
The TM4C129XKCZAD USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB Host, Device, or OTG functions. If the integrated ULPI interface is utilized, the USB can operate at high-speed. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. 16 endpoints including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus 14 endpoints defined by firmware along with a dynamic sizable FIFO support multiple packet queueing. USB DMA access to the FIFO allows minimal interference from system software. Software-controlled connect and disconnect allows flexibility during USB device start-up. The controller complies with OTG Standard's Session Request Protocol (SRP) and Host Negotiation Protocol (HNP).

The TM4C129XKCZAD USB module has the following features:

- Complies with USB-IF (Implementer's Forum) certification standards
- USB 2.0 high-speed (480 Mbps) operation with the integrated ULPI interface communicating with an external PHY
- Link Power Management support which uses link-state awareness to reduce power usage
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 16 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 7 configurable IN endpoints and 7 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop detection and interrupt
- Integrated USB DMA with bus master capability
  - Up to eight RX Endpoint channels and up to eight TX Endpoint channels are available.
  - Each channel can be separately programmed to operate in different modes
  - Incremental burst transfers of 4-, 8-, 16- or unspecified length supported

## 25.1 Block Diagram

**Figure 25-1. USB Module Block Diagram**




## 25.2 Signal Description

The following table lists the external signals of the USB controller and describes the function of each. Some USB controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these USB signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the USB function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the USB signal to the specified GPIO port pin. The USB0VBUS and USB0ID signals are configured by clearing the appropriate DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

**Note:** When used in OTG mode, USB0VBUS and USB0ID do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the DEVMOD field in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to connect the USB0VBUS and/or USB0ID inputs to fixed levels internally, freeing the PB0 and PB1 pins for GPIO use. Note that PB1 (USB0VBUS) is a 5-V tolerant signal as required. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

The termination resistors for the USB PHY have been added internally, and thus there is no need for external resistors. For a device, there is a 1.5 KOhm pull-up on the D+ and for a host there are 15 KOhm pull-downs on both D+ and D-.

**Note:** Port pins PL6 and PL7 operate as Fast GPIO pads, but have 4-mA drive capability only. GPIO register controls for drive strength, slew rate and open drain have no effect on these pins. The registers which have no effect are as follows: **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR12R**, **GPIOSLR**, and **GPIOODR**. Refer to "General-Purpose Input/Outputs (GPIOs)" on page 772 and "Recommended GPIO Operating Characteristics" on page 2097 for more information.

**Table 25-1. USB Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| USB0CLK | B17 | PB3 (14) | O | TTL | 60-MHz clock to the external PHY. |
| USB0D0 | G16 | PL0 (14) | I/O | TTL | USB data 0. |
| USB0D1 | H19 | PL1 (14) | I/O | TTL | USB data 1. |
| USB0D2 | G18 | PL2 (14) | I/O | TTL | USB data 2. |
| USB0D3 | J18 | PL3 (14) | I/O | TTL | USB data 3. |
| USB0D4 | H18 | PL4 (14) | I/O | TTL | USB data 4. |
| USB0D5 | G19 | PL5 (14) | I/O | TTL | USB data 5. |
| USB0D6 | B12 | PP5 (14) | I/O | TTL | USB data 6. |
| USB0D7 | D8 | PP4 (14) | I/O | TTL | USB data 7. |
| USB0DIR | C12 | PP3 (14) | O | TTL | Indicates that the external PHY is able to accept data from the USB controller. |
| USB0DM | B18 | PL7 | I/O | Analog | Bidirectional differential data pin (D- per USB specification) for USB0. |
| USB0DP | C18 | PL6 | I/O | Analog | Bidirectional differential data pin (D+ per USB specification) for USB0. |
| USB0EPEN | V5<br>R7<br>B3 | PA6 (5)<br>PA7 (11)<br>PD6 (5) | O | TTL | Optionally used in Host mode to control an external power source to supply power to the USB bus. |
| USB0ID | A16 | PB0 | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side). |
| USB0NXT | B13 | PP2 (14) | O | TTL | Asserted by the external PHY to throttle all data types. |
| USB0PFLT | R7<br>B2 | PA7 (5)<br>PD7 (5) | I | TTL | Optionally used in Host mode by an external power source to indicate an error state by that power source. |
| USB0STP | A17 | PB2 (14) | O | TTL | Asserted by the USB controller to signal the end of a USB transmit packet or register write operation. |
| USB0VBUS | B16 | PB1 | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |

## 25.3 Register Map

Table 25-2 on page 1834 lists the registers. All addresses given are relative to the USB base address of 0x4005.0000. Note that the USB controller clock must be enabled before the registers can be programmed (see page 410). There must be a delay of 3 system clocks after the USB module clock is enabled before any USB module registers are accessed.

**Note:** Descriptions for all these registers can be found in the NDA version of the data sheet.

**Table 25-2. List of Registers**

| Offset | Name | Description |
|---|---|---|
| 0x000 | USBFADDR | USB Device Functional Address |
| 0x001 | USBPOWER | USB Power |
| 0x002 | USBTXIS | USB Transmit Interrupt Status |
| 0x004 | USBRXIS | USB Receive Interrupt Status |
| 0x006 | USBTXIE | USB Transmit Interrupt Enable |
| 0x008 | USBRXIE | USB Receive Interrupt Enable |
| 0x00A | USBIS | USB General Interrupt Status |
| 0x00B | USBIE | USB Interrupt Enable |
| 0x00C | USBFRAME | USB Frame Value |
| 0x00E | USBEPIDX | USB Endpoint Index |
| 0x00F | USBTEST | USB Test Mode |
| 0x020 | USBFIFO0 | USB FIFO Endpoint 0 |
| 0x024 | USBFIFO1 | USB FIFO Endpoint 1 |
| 0x028 | USBFIFO2 | USB FIFO Endpoint 2 |
| 0x02C | USBFIFO3 | USB FIFO Endpoint 3 |
| 0x030 | USBFIFO4 | USB FIFO Endpoint 4 |
| 0x034 | USBFIFO5 | USB FIFO Endpoint 5 |
| 0x038 | USBFIFO6 | USB FIFO Endpoint 6 |
| 0x03C | USBFIFO7 | USB FIFO Endpoint 7 |
| 0x060 | USBDEVCTL | USB Device Control |
| 0x061 | USBCCONF | USB Common Configuration |
| 0x062 | USBTXFIFOSZ | USB Transmit Dynamic FIFO Sizing |
| 0x063 | USBRXFIFOSZ | USB Receive Dynamic FIFO Sizing |
| 0x064 | USBTXFIFOADD | USB Transmit FIFO Start Address |
| 0x066 | USBRXFIFOADD | USB Receive FIFO Start Address |
| 0x070 | ULPIVBUSCTL | USB ULPI VBUS Control |
| 0x074 | ULPIREGDATA | USB ULPI Register Data |
| 0x075 | ULPIREGADDR | USB ULPI Register Address |
| 0x076 | ULPIREGCTL | USB ULPI Register Control |
| 0x078 | USBEPINFO | USB Endpoint Information |
| 0x079 | USBRAMINFO | USB RAM Information |
| 0x07A | USBCONTIM | USB Connect Timing |
| 0x07B | USBVPLEN | USB OTG VBUS Pulse Timing |
| 0x07C | USBHSEOF | USB High-Speed Last Transaction to End of Frame Timing |
| 0x07D | USBFSEOF | USB Full-Speed Last Transaction to End of Frame Timing |
| 0x07E | USBLSEOF | USB Low-Speed Last Transaction to End of Frame Timing |
| 0x080 | USBTXFUNCADDR0 | USB Transmit Functional Address Endpoint 0 |
| 0x082 | USBTXHUBADDR0 | USB Transmit Hub Address Endpoint 0 |
| 0x083 | USBTXHUBPORT0 | USB Transmit Hub Port Endpoint 0 |
| 0x088 | USBTXFUNCADDR1 | USB Transmit Functional Address Endpoint 1 |

**Table 25-2. List of Registers *(continued)***

| Offset | Name | Description |
|---|---|---|
| 0x08A | USBTXHUBADDR1 | USB Transmit Hub Address Endpoint 1 |
| 0x08B | USBTXHUBPORT1 | USB Transmit Hub Port Endpoint 1 |
| 0x08C | USBRXFUNCADDR1 | USB Receive Functional Address Endpoint 1 |
| 0x08E | USBRXHUBADDR1 | USB Receive Hub Address Endpoint 1 |
| 0x08F | USBRXHUBPORT1 | USB Receive Hub Port Endpoint 1 |
| 0x090 | USBTXFUNCADDR2 | USB Transmit Functional Address Endpoint 2 |
| 0x092 | USBTXHUBADDR2 | USB Transmit Hub Address Endpoint 2 |
| 0x093 | USBTXHUBPORT2 | USB Transmit Hub Port Endpoint 2 |
| 0x094 | USBRXFUNCADDR2 | USB Receive Functional Address Endpoint 2 |
| 0x096 | USBRXHUBADDR2 | USB Receive Hub Address Endpoint 2 |
| 0x097 | USBRXHUBPORT2 | USB Receive Hub Port Endpoint 2 |
| 0x098 | USBTXFUNCADDR3 | USB Transmit Functional Address Endpoint 3 |
| 0x09A | USBTXHUBADDR3 | USB Transmit Hub Address Endpoint 3 |
| 0x09B | USBTXHUBPORT3 | USB Transmit Hub Port Endpoint 3 |
| 0x09C | USBRXFUNCADDR3 | USB Receive Functional Address Endpoint 3 |
| 0x09E | USBRXHUBADDR3 | USB Receive Hub Address Endpoint 3 |
| 0x09F | USBRXHUBPORT3 | USB Receive Hub Port Endpoint 3 |
| 0x0A0 | USBTXFUNCADDR4 | USB Transmit Functional Address Endpoint 4 |
| 0x0A2 | USBTXHUBADDR4 | USB Transmit Hub Address Endpoint 4 |
| 0x0A3 | USBTXHUBPORT4 | USB Transmit Hub Port Endpoint 4 |
| 0x0A4 | USBRXFUNCADDR4 | USB Receive Functional Address Endpoint 4 |
| 0x0A6 | USBRXHUBADDR4 | USB Receive Hub Address Endpoint 4 |
| 0x0A7 | USBRXHUBPORT4 | USB Receive Hub Port Endpoint 4 |
| 0x0A8 | USBTXFUNCADDR5 | USB Transmit Functional Address Endpoint 5 |
| 0x0AA | USBTXHUBADDR5 | USB Transmit Hub Address Endpoint 5 |
| 0x0AB | USBTXHUBPORT5 | USB Transmit Hub Port Endpoint 5 |
| 0x0AC | USBRXFUNCADDR5 | USB Receive Functional Address Endpoint 5 |
| 0x0AE | USBRXHUBADDR5 | USB Receive Hub Address Endpoint 5 |
| 0x0AF | USBRXHUBPORT5 | USB Receive Hub Port Endpoint 5 |
| 0x0B0 | USBTXFUNCADDR6 | USB Transmit Functional Address Endpoint 6 |
| 0x0B2 | USBTXHUBADDR6 | USB Transmit Hub Address Endpoint 6 |
| 0x0B3 | USBTXHUBPORT6 | USB Transmit Hub Port Endpoint 6 |
| 0x0B4 | USBRXFUNCADDR6 | USB Receive Functional Address Endpoint 6 |
| 0x0B6 | USBRXHUBADDR6 | USB Receive Hub Address Endpoint 6 |
| 0x0B7 | USBRXHUBPORT6 | USB Receive Hub Port Endpoint 6 |
| 0x0B8 | USBTXFUNCADDR7 | USB Transmit Functional Address Endpoint 7 |
| 0x0BA | USBTXHUBADDR7 | USB Transmit Hub Address Endpoint 7 |
| 0x0BB | USBTXHUBPORT7 | USB Transmit Hub Port Endpoint 7 |
| 0x0BC | USBRXFUNCADDR7 | USB Receive Functional Address Endpoint 7 |
| 0x0BE | USBRXHUBADDR7 | USB Receive Hub Address Endpoint 7 |
| 0x0BF | USBRXHUBPORT7 | USB Receive Hub Port Endpoint 7 |
| 0x102 | USBCSRL0 | USB Control and Status Endpoint 0 Low |

**Table 25-2. List of Registers *(continued)***

| Offset | Name | Description |
|---|---|---|
| 0x103 | USBCSRH0 | USB Control and Status Endpoint 0 High |
| 0x108 | USBCOUNT0 | USB Receive Byte Count Endpoint 0 |
| 0x10A | USBTYPE0 | USB Type Endpoint 0 |
| 0x10B | USBNAKLMT | USB NAK Limit |
| 0x110 | USBTXMAXP1 | USB Maximum Transmit Data Endpoint 1 |
| 0x112 | USBTXCSRL1 | USB Transmit Control and Status Endpoint 1 Low |
| 0x113 | USBTXCSRH1 | USB Transmit Control and Status Endpoint 1 High |
| 0x114 | USBRXMAXP1 | USB Maximum Receive Data Endpoint 1 |
| 0x116 | USBRXCSRL1 | USB Receive Control and Status Endpoint 1 Low |
| 0x117 | USBRXCSRH1 | USB Receive Control and Status Endpoint 1 High |
| 0x118 | USBRXCOUNT1 | USB Receive Byte Count Endpoint 1 |
| 0x11A | USBTXTYPE1 | USB Host Transmit Configure Type Endpoint 1 |
| 0x11B | USBTXINTERVAL1 | USB Host Transmit Interval Endpoint 1 |
| 0x11C | USBRXTYPE1 | USB Host Configure Receive Type Endpoint 1 |
| 0x11D | USBRXINTERVAL1 | USB Host Receive Polling Interval Endpoint 1 |
| 0x120 | USBTXMAXP2 | USB Maximum Transmit Data Endpoint 2 |
| 0x122 | USBTXCSRL2 | USB Transmit Control and Status Endpoint 2 Low |
| 0x123 | USBTXCSRH2 | USB Transmit Control and Status Endpoint 2 High |
| 0x124 | USBRXMAXP2 | USB Maximum Receive Data Endpoint 2 |
| 0x126 | USBRXCSRL2 | USB Receive Control and Status Endpoint 2 Low |
| 0x127 | USBRXCSRH2 | USB Receive Control and Status Endpoint 2 High |
| 0x128 | USBRXCOUNT2 | USB Receive Byte Count Endpoint 2 |
| 0x12A | USBTXTYPE2 | USB Host Transmit Configure Type Endpoint 2 |
| 0x12B | USBTXINTERVAL2 | USB Host Transmit Interval Endpoint 2 |
| 0x12C | USBRXTYPE2 | USB Host Configure Receive Type Endpoint 2 |
| 0x12D | USBRXINTERVAL2 | USB Host Receive Polling Interval Endpoint 2 |
| 0x130 | USBTXMAXP3 | USB Maximum Transmit Data Endpoint 3 |
| 0x132 | USBTXCSRL3 | USB Transmit Control and Status Endpoint 3 Low |
| 0x133 | USBTXCSRH3 | USB Transmit Control and Status Endpoint 3 High |
| 0x134 | USBRXMAXP3 | USB Maximum Receive Data Endpoint 3 |
| 0x136 | USBRXCSRL3 | USB Receive Control and Status Endpoint 3 Low |
| 0x137 | USBRXCSRH3 | USB Receive Control and Status Endpoint 3 High |
| 0x138 | USBRXCOUNT3 | USB Receive Byte Count Endpoint 3 |
| 0x13A | USBTXTYPE3 | USB Host Transmit Configure Type Endpoint 3 |
| 0x13B | USBTXINTERVAL3 | USB Host Transmit Interval Endpoint 3 |
| 0x13C | USBRXTYPE3 | USB Host Configure Receive Type Endpoint 3 |
| 0x13D | USBRXINTERVAL3 | USB Host Receive Polling Interval Endpoint 3 |
| 0x140 | USBTXMAXP4 | USB Maximum Transmit Data Endpoint 4 |
| 0x142 | USBTXCSRL4 | USB Transmit Control and Status Endpoint 4 Low |
| 0x143 | USBTXCSRH4 | USB Transmit Control and Status Endpoint 4 High |
| 0x144 | USBRXMAXP4 | USB Maximum Receive Data Endpoint 4 |
| 0x146 | USBRXCSRL4 | USB Receive Control and Status Endpoint 4 Low |

**Table 25-2. List of Registers *(continued)***

| Offset | Name | Description |
|---|---|---|
| 0x147 | USBRXCSRH4 | USB Receive Control and Status Endpoint 4 High |
| 0x148 | USBRXCOUNT4 | USB Receive Byte Count Endpoint 4 |
| 0x14A | USBTXTYPE4 | USB Host Transmit Configure Type Endpoint 4 |
| 0x14B | USBTXINTERVAL4 | USB Host Transmit Interval Endpoint 4 |
| 0x14C | USBRXTYPE4 | USB Host Configure Receive Type Endpoint 4 |
| 0x14D | USBRXINTERVAL4 | USB Host Receive Polling Interval Endpoint 4 |
| 0x150 | USBTXMAXP5 | USB Maximum Transmit Data Endpoint 5 |
| 0x152 | USBTXCSRL5 | USB Transmit Control and Status Endpoint 5 Low |
| 0x153 | USBTXCSRH5 | USB Transmit Control and Status Endpoint 5 High |
| 0x154 | USBRXMAXP5 | USB Maximum Receive Data Endpoint 5 |
| 0x156 | USBRXCSRL5 | USB Receive Control and Status Endpoint 5 Low |
| 0x157 | USBRXCSRH5 | USB Receive Control and Status Endpoint 5 High |
| 0x158 | USBRXCOUNT5 | USB Receive Byte Count Endpoint 5 |
| 0x15A | USBTXTYPE5 | USB Host Transmit Configure Type Endpoint 5 |
| 0x15B | USBTXINTERVAL5 | USB Host Transmit Interval Endpoint 5 |
| 0x15C | USBRXTYPE5 | USB Host Configure Receive Type Endpoint 5 |
| 0x15D | USBRXINTERVAL5 | USB Host Receive Polling Interval Endpoint 5 |
| 0x160 | USBTXMAXP6 | USB Maximum Transmit Data Endpoint 6 |
| 0x162 | USBTXCSRL6 | USB Transmit Control and Status Endpoint 6 Low |
| 0x163 | USBTXCSRH6 | USB Transmit Control and Status Endpoint 6 High |
| 0x164 | USBRXMAXP6 | USB Maximum Receive Data Endpoint 6 |
| 0x166 | USBRXCSRL6 | USB Receive Control and Status Endpoint 6 Low |
| 0x167 | USBRXCSRH6 | USB Receive Control and Status Endpoint 6 High |
| 0x168 | USBRXCOUNT6 | USB Receive Byte Count Endpoint 6 |
| 0x16A | USBTXTYPE6 | USB Host Transmit Configure Type Endpoint 6 |
| 0x16B | USBTXINTERVAL6 | USB Host Transmit Interval Endpoint 6 |
| 0x16C | USBRXTYPE6 | USB Host Configure Receive Type Endpoint 6 |
| 0x16D | USBRXINTERVAL6 | USB Host Receive Polling Interval Endpoint 6 |
| 0x170 | USBTXMAXP7 | USB Maximum Transmit Data Endpoint 7 |
| 0x172 | USBTXCSRL7 | USB Transmit Control and Status Endpoint 7 Low |
| 0x173 | USBTXCSRH7 | USB Transmit Control and Status Endpoint 7 High |
| 0x174 | USBRXMAXP7 | USB Maximum Receive Data Endpoint 7 |
| 0x176 | USBRXCSRL7 | USB Receive Control and Status Endpoint 7 Low |
| 0x177 | USBRXCSRH7 | USB Receive Control and Status Endpoint 7 High |
| 0x178 | USBRXCOUNT7 | USB Receive Byte Count Endpoint 7 |
| 0x17A | USBTXTYPE7 | USB Host Transmit Configure Type Endpoint 7 |
| 0x17B | USBTXINTERVAL7 | USB Host Transmit Interval Endpoint 7 |
| 0x17C | USBRXTYPE7 | USB Host Configure Receive Type Endpoint 7 |
| 0x17D | USBRXINTERVAL7 | USB Host Receive Polling Interval Endpoint 7 |
| 0x200 | USBDMAINTR | USB DMA Interrupt |
| 0x204 | USBDMACTL0 | USB DMA Control 0 |
| 0x208 | USBDMAADDR0 | USB DMA Address 0 |

**Table 25-2. List of Registers *(continued)***

| Offset | Name | Description |
|---|---|---|
| 0x20C | USBDMACOUNT0 | USB DMA Count 0 |
| 0x214 | USBDMACTL1 | USB DMA Control 1 |
| 0x218 | USBDMAADDR1 | USB DMA Address 1 |
| 0x21C | USBDMACOUNT1 | USB DMA Count 1 |
| 0x224 | USBDMACTL2 | USB DMA Control 2 |
| 0x228 | USBDMAADDR2 | USB DMA Address 2 |
| 0x22C | USBDMACOUNT2 | USB DMA Count 2 |
| 0x234 | USBDMACTL3 | USB DMA Control 3 |
| 0x238 | USBDMAADDR3 | USB DMA Address 3 |
| 0x23C | USBDMACOUNT3 | USB DMA Count 3 |
| 0x244 | USBDMACTL4 | USB DMA Control 4 |
| 0x248 | USBDMAADDR4 | USB DMA Address 4 |
| 0x24C | USBDMACOUNT4 | USB DMA Count 4 |
| 0x254 | USBDMACTL5 | USB DMA Control 5 |
| 0x258 | USBDMAADDR5 | USB DMA Address 5 |
| 0x25C | USBDMACOUNT5 | USB DMA Count 5 |
| 0x264 | USBDMACTL6 | USB DMA Control 6 |
| 0x268 | USBDMAADDR6 | USB DMA Address 6 |
| 0x26C | USBDMACOUNT6 | USB DMA Count 6 |
| 0x274 | USBDMACTL7 | USB DMA Control 7 |
| 0x278 | USBDMAADDR7 | USB DMA Address 7 |
| 0x27C | USBDMACOUNT7 | USB DMA Count 7 |
| 0x304 | USBRQPKTCOUNT1 | USB Request Packet Count in Block Transfer Endpoint 1 |
| 0x308 | USBRQPKTCOUNT2 | USB Request Packet Count in Block Transfer Endpoint 2 |
| 0x30C | USBRQPKTCOUNT3 | USB Request Packet Count in Block Transfer Endpoint 3 |
| 0x310 | USBRQPKTCOUNT4 | USB Request Packet Count in Block Transfer Endpoint 4 |
| 0x314 | USBRQPKTCOUNT5 | USB Request Packet Count in Block Transfer Endpoint 5 |
| 0x318 | USBRQPKTCOUNT6 | USB Request Packet Count in Block Transfer Endpoint 6 |
| 0x31C | USBRQPKTCOUNT7 | USB Request Packet Count in Block Transfer Endpoint 7 |
| 0x340 | USBRXDPKTBUFDIS | USB Receive Double Packet Buffer Disable |
| 0x342 | USBTXDPKTBUFDIS | USB Transmit Double Packet Buffer Disable |
| 0x344 | USBCTO | USB Chirp Timeout |
| 0x346 | USBHHSRTN | USB High Speed to UTM Operating Delay |
| 0x348 | USBHSBT | USB High Speed Time-out Adder |
| 0x360 | USBLPMATTR | USB LPM Attributes |
| 0x362 | USBLPMCNTRL | USB LPM Control |
| 0x363 | USBLPMIM | USB LPM Interrupt Mask |
| 0x364 | USBLPMRIS | USB LPM Raw Interrupt Status |
| 0x365 | USBLPMFADDR | USB LPM Function Address |
| 0x400 | USBEPC | USB External Power Control |
| 0x404 | USBEPCRIS | USB External Power Control Raw Interrupt Status |
| 0x408 | USBEPCIM | USB External Power Control Interrupt Mask |

**Table 25-2. List of Registers *(continued)***

| Offset | Name | Description |
|---|---|---|
| 0x40C | USBEPCISC | USB External Power Control Interrupt Status and Clear |
| 0x410 | USBDRRIS | USB Device RESUME Raw Interrupt Status |
| 0x414 | USBDRIM | USB Device RESUME Interrupt Mask |
| 0x418 | USBDRISC | USB Device RESUME Interrupt Status and Clear |
| 0x41C | USBGPCS | USB General-Purpose Control and Status |
| 0x430 | USBVDC | USB VBUS Droop Control |
| 0x434 | USBVDCRIS | USB VBUS Droop Control Raw Interrupt Status |
| 0x438 | USBVDCIM | USB VBUS Droop Control Interrupt Mask |
| 0x43C | USBVDCISC | USB VBUS Droop Control Interrupt Status and Clear |
| 0xFC0 | USBPP | USB Peripheral Properties |
| 0xFC4 | USBPC | USB Peripheral Configuration |
| 0xFC8 | USBCC | USB Clock Configuration |

# 26 LCD Controller

The Liquid Crystal Display (LCD) Controller provides support for a variety of LCD and OLED panels.

The TM4C129XKCZAD controller includes an LCD Controller with the following features:

- Character-based panels
  - Support for two character panels (CS0 and CS1) with independent and programmable bus timing parameters when in asynchronous Hitachi, Motorola, and Intel modes
  - Support for one character panel (CS0) with programmable bus timing parameters when in synchronous Motorola and Intel modes
  - Can be used as a generic 16-bit address/data interleaved MPU bus master with no external stall
- Passive matrix LCD panels
  - Panel types including STN, DSTN, and C-DSTN
  - AC Bias Control
- Active matrix LCD panels
  - Panel types including TN TFT
  - 1, 2, 4, or 8 bits per pixel with palette RAM and 16 or 24 bits per pixel without palette RAM
- OLED Panels
  - Passive Matrix (PM OLED) with frame buffer and controller IC inside the panel
  - Active Matrix (AM OLED)
- Bus mastering capability from either system SRAM or EPI memory.

## 26.1 Block Diagram

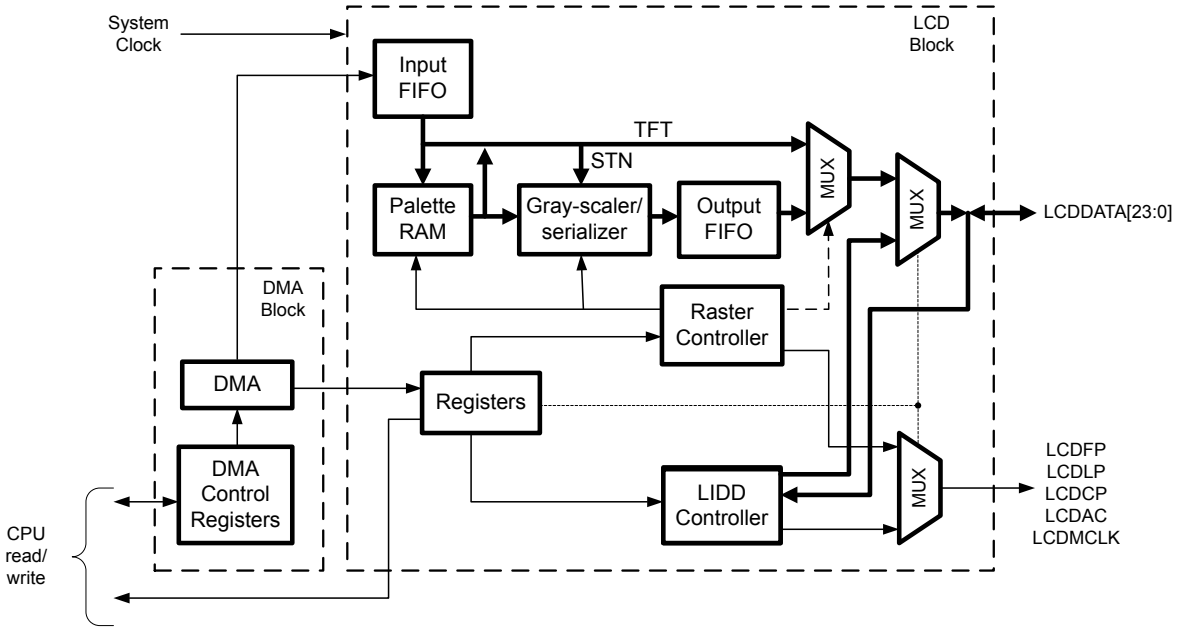
The LCD controller consists of two independent controllers, the Raster Controller and the LCD Interface Display Driver (LIDD) controller. Each controller operates independently from the other and only one of them is active at any given time.

- The Raster Controller provides a synchronous LCD interface. It provides timing and data for constant graphics refresh to a passive display. It supports a wide variety of monochrome and full-color display types and sizes by use of programmable timing controls, a built-in palette, and a gray-scale/serializer. Graphics data is processed and stored in frame buffers. A frame buffer is a contiguous memory block in the system. A built-in DMA engine supplies the graphics data to the Raster engine which, in turn, outputs to the external LCD device.
- The LIDD controller provides an asynchronous and/or synchronous interface depending on the mode the LIDD controller is operating in. It provides full-timing programmability of control signals (chip selects, read/write strobes, enable, direction) and output data.

Figure 26-1 on page 1841 shows the LCD controller details. The raster and LIDD Controllers are responsible for generating the correct external timing. The LCD DMA engine provides a constant flow of data from the frame buffer(s) to the external LCD panel via the Raster and LIDD Controllers. In addition, CPU access is provided to read and write registers.

The solid, thick lines in Figure 13-1 indicate the data path. For more information on the Raster Controller data path, see Figure 26-3 on page 1850 .

**Figure 26-1. LCD Block Diagram**




## 26.2 Signal Description

The following table lists the external signals of the LCD controller and describes the function of each.

**Table 26-1. LCD Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| LCDAC | N1 | PJ6 (15) | O | TTL | LCD AC bias or latch enable in Raster mode; Primary chip select (CS0)/Primary Enable (E0) in LIDD MPU/Hitachi mode |
| LCDCP | N5 | PR0 (15) | O | TTL | LCD Pixel Clock in Raster mode; read strobe or read/write strobe in LIDD mode |
| LCDDATA00 | P3 | PR4 (15) | I/O | TTL | LCD Data Pin 0 input/output. |
| LCDDATA01 | P2 | PR5 (15) | I/O | TTL | LCD Data Pin 1 input/output. |
| LCDDATA02 | U8 | PF7 (15) | I/O | TTL | LCD Data Pin 2 input/output. |
| LCDDATA03 | V8 | PR3 (15) | I/O | TTL | LCD Data Pin 3 input/output. |
| LCDDATA04 | W9 | PR6 (15) | I/O | TTL | LCD Data Pin 4 input/output. |
| LCDDATA05 | R10 | PR7 (15) | I/O | TTL | LCD Data Pin 5 input/output. |

**Table 26-1. LCD Signals (212BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| LCDDATA06 | V9 | PS4 (15) | I/O | TTL | LCD Data Pin 6 input/output. |
| LCDDATA07 | T13 | PS5 (15) | I/O | TTL | LCD Data Pin 7 input/output. |
| LCDDATA08 | U10 | PS6 (15) | I/O | TTL | LCD Data Pin 8 input/output. |
| LCDDATA09 | R13 | PS7 (15) | I/O | TTL | LCD Data Pin 9 input/output. |
| LCDDATA10 | W10 | PT0 (15) | I/O | TTL | LCD Data Pin 10 input/output. |
| LCDDATA11 | V10 | PT1 (15) | I/O | TTL | LCD Data Pin 11 input/output. |
| LCDDATA12 | U12 | PN7 (15) | I/O | TTL | LCD Data Pin 12 input/output. |
| LCDDATA13 | T12 | PN6 (15) | I/O | TTL | LCD Data Pin 13 input/output. |
| LCDDATA14 | H17 | PJ2 (15) | I/O | TTL | LCD Data Pin 14 input/output. |
| LCDDATA15 | F16 | PJ3 (15) | I/O | TTL | LCD Data Pin 15 input/output. |
| LCDDATA16 | F18 | PJ4 (15) | O | TTL | LCD Data Pin 16 output. |
| LCDDATA17 | E17 | PJ5 (15) | O | TTL | LCD Data Pin 17 output. |
| LCDDATA18 | E18 | PT2 (15) | O | TTL | LCD Data Pin 18 output. |
| LCDDATA19 | F17 | PT3 (15) | O | TTL | LCD Data Pin 19 output. |
| LCDDATA20 | D12 | PS0 (15) | O | TTL | LCD Data Pin 20 output. |
| LCDDATA21 | D13 | PS1 (15) | O | TTL | LCD Data Pin 21 output. |
| LCDDATA22 | B14 | PS2 (15) | O | TTL | LCD Data Pin 22 output. |
| LCDDATA23 | A14 | PS3 (15) | O | TTL | LCD Data Pin 23 output. |
| LCDFP | N4 | PR1 (15) | O | TTL | LCD Frame Clock or VSYNC in Raster mode; Address Latch Enable in LIDD mode |
| LCDLP | N2 | PR2 (15) | O | TTL | LCD Line Clock or HSYNC in Raster mode; Write Strobe or Direction bit in LIDD mode |
| LCDMCLK | T8 | PF6 (15) | O | TTL | LCD Memory Clock/Secondary chip Select (CS1)/Secondary Enable (E1) in LIDD Synchronous/Async MPU/Hitachi mode |

The following table gives a more detailed summary of how the signals are used in each mode. These features are also discussed in the functional description section.

**Table 26-2. LCD External Signal Details**

| Signal | Type | Mode | Function |
|---|---|---|---|
| LCDFP (VSYNC) | Output | Raster Mode | This signal functions as a frame clock the LCD uses to signal the start of a new frame of pixels. This signal is also used by TFT displays as the vertical synchronization signal. |
| | | LIDD Mode | This signal functions as an Address Latch Enable (ALE) |
| LCDLP (HSYNC) | Output | Raster Mode | This signal functions as a line clock the LCD uses to signal the end of a line of pixels. This signal is also used by TFT displays as the horizontal synchronization signal. |
| | | LIDD Mode | This signal functions as a write strobe or direction bit depending on the interface mode. |

**Table 26-2. LCD External Signal Details *(continued)***

| Signal | Type | Mode | Function |
|---|---|---|---|
| LCDCP (Pixel clock) | Output | Raster Mode | This signal functions as the pixel clock the LCD uses to clock the pixel data into the line shift register. In passive mode, the pixel clock transitions only when valid data is available on the data lines. In active mode, the pixel clock transitions continuously, and the LCDAC pin is used as an output enable to signal when data is available on the LCD pin. |
| | | LIDD Mode | This signal functions as a read strobe (RS) or read/write strobe (RS/WS) depending on the interface mode. |
| LCDAC | Output | Raster Mode | This signal functions as the AC-bias used to trigger the LCD to switch the polarity of the power supplies to the row and column axis of the screen to counteract DC offset. The signal is used in TFT mode as the output enable to signal when data is latched from the data pins using the pixel clock.<br>LIDD Character: Primary Enable Strobe (E0)<br>LIDD Graphics: Chip Select 0 (CS0) |
| | | LIDD Mode | This signal functions as the primary enable (E0) or Hitachi mode and the primary chip select (CS0) depending on the interface mode. |
| LCDMCLK | OUT | LIDD Mode | This signal is not used in Raster mode, but is used as a secondary Enable Strobe (E1) or Chip Select (CS1) depending on the interface mode. |
| LCDDATA | Output | Raster Mode | These signals function as the LCD data bus providing 4-, 8-, 16-, or 24-bit data path. For monochrome displays, each signal represents a pixel; for passive color displays, groupings of three signals represent one pixel (red, green, and blue). LCDDATA[3:0] are used for monochrome displays of 2, 4, and 8 BPP; LCDDATA[7:0] are used for color STN displays and LCDDATA[15:0] or LCDDATA[23:0] are used for active (TFT) mode. |
| | Output/Input | LIDD Mode | These signals function as the read and write path for the command and data registers. |

Table 26-3 on page 1843 details MODESEL programming for the alternate pin functions in LIDD mode. For further programming information refer to the **LCDLIDDCTL** register.

**Table 26-3. LCD Alternate Signal Functions in LIDD Mode**

| PIN | MODE = 0x0 | MODE = 0x1 | MODE = 0x2 | MODE = 0x3 | MODE = 0x4 |
|---|---|---|---|---|---|
| LCDCP | RS/WS | RS/WS | RS | RS | N/A |
| LCDLP | DIR | DIR | WS | WS | DIR |
| LCDFP | ALE | ALE | ALE | ALE | ALE |
| LCDAC | CS0 | CS0 | CS0 | CS0 | E0 |
| LCDMCLK | MCLK | CS1 | MCLK | CS1 | E1 |

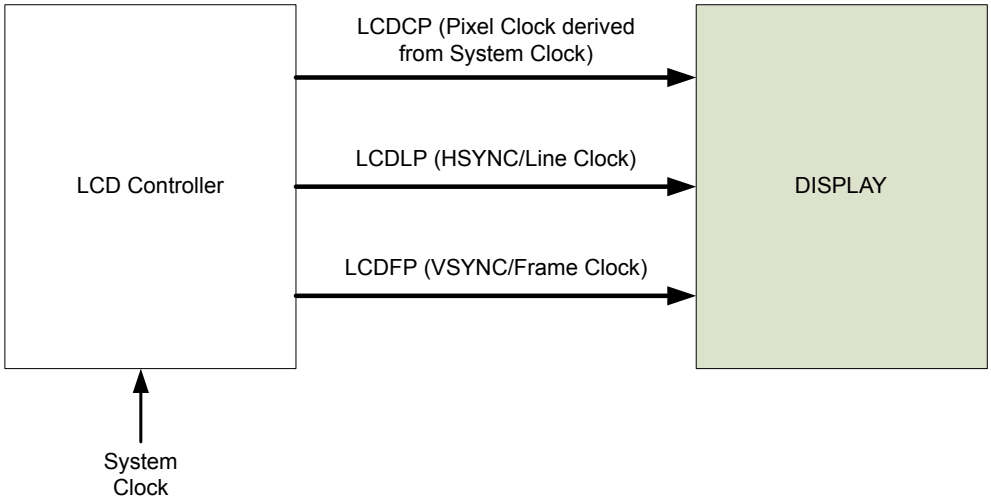
## 26.3 Functional Description

The following sections describe the functional capability of the LCD Controller

### 26.3.1 Clocking

This section details the various clocks and signals. Figure 26-2 on page 1844 shows the input and output LCD controller clocks.

**Figure 26-2. Input and Output Clocks**




### 26.3.1.1 Pixel Clock (LCDCP signal)

The pixel clock (LCDCP) frequency is derived from the system clock, which is the internal reference clock (MCLK) to the LCD module. The pixel clock is used by the LCD display to clock the pixel data into the line shift register. The formula used for the pixel clock is:

LCDCP = System Clock / CLKDIV

where CLKDIV is a field in the **LCD Control (LCDCTL)** register. The value of CLKDIV should not be 0 or 1 in Raster Mode. The pixel clock operates differently depending on the mode of the LCD controller:

- Passive (STN) mode: LCDCP only transitions when valid data is available for output. It does not transition when the horizontal clock (HSYNC) is asserted or during wait state insertion.
- Active (TFT) mode: LCDCP continuously toggles as long as the Raster Controller is enabled.

### 26.3.1.2 HSYNC Horizontal Clock (LCDLP signal)

HSYNC (LCDLP signal) toggles after all pixels in a horizontal line have been transmitted to the LCD and a programmable number of pixel clock wait states have elapsed both at the beginning and end of each line. The **LCD Raster Timing 0 (LCDRASTRTIM0)** register, offset 0x02C, fully defines the behavior of this signal.

HSYNC can be programmed to be synchronized with the rising or falling edge of the pixel clock (LCDCP). The PXLCLKCTL and PSYNCRF bits in the **LCD Raster Timing 2 (LCDRASTRTIM2)** register, offset 0x034, are used to configure the synchronization.

- Active (TFT) mode: The horizontal clock or the line clock is also used by TFT displays as the horizontal synchronization signal (HSYNC).

  The timings of the horizontal clock (line clock) pins are programmable to support:

  – Delay insertion both at the beginning and end of each line

– Line clock polarity

– Line clock pulse width, driven on rising or falling edge of pixel clock

#### 26.3.1.3 VSYNC Vertical Clock (LCDFP signal)

VSYNC (LCDFP signal) toggles after all lines in a frame have been transmitted to the LCD and a programmable number of line clock cycles has elapsed both at the beginning and end of each frame. The **LCD Raster Timing 1 (LCDRASTRTIM1)** register, offset 0x030, fully defines the behavior of this signal.

VSYNC can be programmed to be synchronized with the rising or falling edge of LCDCP pixel clock. The PXLCLKCTL and PSYNCRF bits in the **LCD Raster Timing 2 (LCDRASTRTIM2)** register, offset 0x034, are used to configure the synchronization.

- Passive (STN) mode: The vertical, or frame, clock toggles during the first line of the screen.
- Active (TFT) mode: The vertical, or frame, clock is also used by TFT displays as the vertical synchronization signal (VSYNC). The timings of the vertical clock pins are programmable to support:

  – Delay insertion both at the beginning and end of each frame

  – Frame clock polarity

#### 26.3.1.4 LCD AC Bias Enable (LCDAC)

- Passive (STN) mode: To prevent a DC charge within the screen pixels, the power and ground supplies of the display are periodically switched. The Raster Controller signals the LCD to switch the polarity by toggling this pin.
- Active (TFT) mode: This signal acts as an output enable (OE) signal. It is used to signal the external LCD that the data is valid on the data bus.

### 26.3.2 LCD DMA Engine

The LCD DMA engine provides the capability to output graphics data to constantly refresh the external LCD display, without burdening the CPU, via interrupts or a firmware timer. It operates on one or two frame buffers, which are set up during initialization. Using two frame buffers (ping-pong buffers) enables the simultaneous operation of outputting the current video frame to the external display and updating the next video frame. The ping-pong buffering approach is preferred in most applications.

When the Raster Controller is used, the LCD DMA engine reads data from a frame buffer and writes it to the input FIFO. The Raster Controller requests data from the FIFO for frame refresh, so the DMA is used to keep the FIFO filled.

When the LIDD controller is used, the LCD DMA engine accesses the LIDD controller's address and/or data registers. The following steps are needed to configure the DMA engine:

1. Configure the data format in the **LCD DMA Control (LCDDMACTL)** register, offset 0x040.
2. Configure frame buffer 0 by programming the **LCD DMA Frame Buffer 0 Base Address (LCDDMABAFB0)** register, offset 0x044, and the **LCD DMA Frame Buffer 0 Ceiling Address (LCDDMACAFB0)** register, offset 0x048

3. Configure frame buffer 1 by programming the **LCD DMA Frame Buffer 1 Base Address (LCDDMABAFB1)** register, offset 0x04C, and the **LCD DMA Frame Buffer 1 Ceiling Address (LCDDMACAFB1)** register, offset 0x050.

In addition, depending on whether the application is operating in LIDD mode or Raster Mode, the **LCDLIDDCTL** or the **LCDRASTRCTL** register should be configured appropriately, along with all of the timing registers. To enable DMA transfers, the DMAEN bit in the **LCDLIDDCTL** or the LCDEN bit in the **LCDRASTRCTL** register should be written with a 1.

#### 26.3.2.1 Interrupts

Interrupts in this LCD module are related to DMA engine operation. Five registers are used to control and monitor the interrupts:

- **LCDLIDDCTL** and **LCDRASTRCTL** registers enable and disable individual features that can generate interrupts.
- The **LCD Interrupt Raw Status and Set Register (LCDRISSET)** register, offset 0x058, collects all of the interrupt status information.
- The **LCD Interrupt Status and Clear (LCDMISCLR)** register, offset 0x05C, is used to read and clear the status of the masked interrupt triggers.
- The **LCD Interrupt Mask (LCDIM)** register, offset 0x060, is used to control whether an interrupt source is masked or not.

***LIDD Mode***

When operating in LIDD mode, the DMA engine generates one interrupt signal every time the specified frame buffer has been transferred completely. The EOF1, EOF0 and DONE bits of the **LCDRISSET** register reflect the interrupt signal, regardless of whether or not the interrupt is enabled.

***Raster Mode***

When operating in Raster mode, the DMA engine can generate the interrupts in the following scenarios:

- Output FIFO under-run: This interrupt occurs when the DMA engine cannot keep up with the data rate consumed by the LCD (which is determined by the LCDCP.) This is likely due to a system memory throughput issue or an incorrect LCDCP setting. The FIFOU bit in **LCD Interrupt Raw Status and Set Register (LCDRISSET)** register is set when this error occurs. This bit is cleared by disabling the Raster Controller, that is clearing the LCDEN bit in **LCD Raster Control (LCDRASTRCTL)** register.
- Frame Synchronization lost: This error happens when the DMA engine attempts to read what it believes to be the first word of the video buffer but the data cannot be recognized as such. This situation can be caused by an invalid frame buffer address or an invalid BPP value. The SYNC bit in the IRQSTATUS_RAW register is set when such an error is detected. This field is cleared by disabling the Raster Controller (clearing the LCDEN bit in **LCD Raster Control (LCDRASTRCTL)** register).
- Palette loaded: This interrupt can be generated when the palette is loaded into the memory by the DMA engine. At the same time, the PALLOAD bit in the **LCD Interrupt Raw Status and Set Register (LCDRISSET)** register is set. In data-only (PALMODE = 0x2) and palette-plus-data (PALMODE = 0x0) modes, writing 0 to this bit clears the interrupt. In the palette-only (PALMODE

= 0x1) mode, this bit is cleared by disabling the Raster Controller (clearing the LCDEN bit in the **LCD Raster Control (LCDRASTRCTL)** register).

- AC Bias transition: If the ACBI field in the **LCD Raster Timing 2 (LCDRASTRTIM2)** register is programmed with a nonzero value, an internal counter is loaded with this value and starts to decrement each time LCDAC (AC-bias signal) switches its state. When the counter reaches zero, the ACBS bit in the **LCD Interrupt Raw Status and Set Register (LCDRISSET)** register is set, which delivers an interrupt signal to the system interrupt controller (if the interrupt is enabled.) The counter reloads the value in ACBI field, but does not start to decrement until the ABC bit is cleared by writing 0 to this bit.

- Frame transfer completed: When one frame of data is transferred completely, the DONE bit in the **LCD Interrupt Raw Status and Set Register (LCDRISSET)** register is set. This bit is cleared by disabling the Raster Controller (clearing the LCDEN bit in **LCD Raster Control (LCDRASTRCTL)** register). The EOF0 and EOF1 bits in **LCD Interrupt Raw Status and Set Register (LCDRISSET)** register are set accordingly.

Note that the function enable bits are in the in **LCD Raster Control (LCDRASTRCTL)** register and must be set in order to generate an interrupt to the CPU.

### 26.3.3 LIDD Bus Operation

The integrated LCD Interface Display Drive (LIDD) Controller has programmable timing parameters that support a wide variety of character-based LCD panels. Alternatively, the DMA module can be used to mimic the CPU and perform a sequence of write-only data bus transactions to the character-based LCD panel.

LIDD mode is enabled by clearing the LCDMODE bit in the **LCD Control (LCDCTL)** register.

LIDD Controller operation is summarized as follows:

- During initialization, the **LCD LIDD CS0 Configuration (LIDDCS0CFG)** and **LCD LIDD CS1 Configuration (LIDDCS1CFG)** registers are configured to match the requirements of the LCD panel being used.

- During normal operation, the CPU writes display data to the **LIDD CS0 Data Read/Write Initiation (LIDDCS0DATA)** and **LIDD CS1 Data Read/Write Initiation (LIDDCS1DATA)** registers. The LIDD interface converts the CPU write into the proper signal transition sequence for the display, as programmed earlier. Note that the first CPU write should send the beginning address of the update to the LCD panel and the subsequent writes update data at display locations starting from the first address and continuing sequentially. Note that DMA may be used instead of CPU.

- The LIDD Controller is also capable of reading back status or data from the LCD panel, if the latter has this capability. This is set up and activated in a similar manner to the write function described above.

Table 26-4 on page 1848 describes how the signals are used to interface to external LCD modules, which are configured by the **LCDLIDDCTL** register.

**Table 26-4. LIDD I/O Name Map**

| Display Type | Interface Type | Data Bits | LCDLIDDCTL[2:0] | Signal Name | LCD Display I/O Name | LCD Display I/O Description |
|---|---|---|---|---|---|---|
| Character Display | HD44780 Type | 4 | 0x4 | LCDDATA[7:4] | DATA[7:4] | LCD Data Bus (length defined by instruction) |
| | | | | LCDAC | E (or E0) | Enable Strobe (first display) |
| | | | | LCDLP | R/W | Read/Write |
| | | | | LCDFP | RS | Register Select (Data/not Instruction |
| | | | | LCDMCLK | E1 | Enable Strobe (second display optional) |
| Character Display | HD44780 Type | 8 | 0x4 | LCDDATA[7:0] | DATA[7:0] | LCD Data Bus (length defined by instruction) |
| | | | | LCDAC | E (or E0) | Enable Strobe (first display) |
| | | | | LCDLP | R/W | Read/Write |
| | | | | LCDFP | RS | Register Select (Data/not Instruction |
| | | | | LCDMCLK | E1 | Enable Strobe (second display optional) |
| Micro Interface Graphic Display | 6800 Family | Up to 16 | 0x1 | LCDDATA[15:0] | DATA[15:0] | LCD Data Bus (16 bits always available) |
| | | | | LCDCP | E | Enable Clock |
| | | | | LCDLP | R/W | Read/Write |
| | | | | LCDFP | A0 | Address/Data Select |
| | | | | LCDAC | CS (or CS0) | Chip Select (first display) |
| | | | | LCDMCLK | CS1 | Chip Select (second display optional) |
| | | | 0x0 | LCDMCLK | None | Synchronous Clock (optional) |
| Micro Interface Graphic Display | 8080 Family | Up to 16 | 0x3 | LCDDATA[15:0] | DATA[15:0] | LCD Data Bus (16 bits always available) |
| | | | | LCDCP | RD | Read Strobe |
| | | | | LCDLP | WR | Write Strobe |
| | | | | LCDFP | A0 | Address/Data Select |
| | | | | LCDAC | CS (or CS0) | Chip Select (first display) |
| | | | | LCDMCLK | CS1 | Chip Select (second display optional) |
| | | | 0x2 | LCDMCLK | None | Synchronous Clock (optional) |

## 26.3.4 Raster Control

The Raster mode is enabled by setting the MODESEL bit in the **LCD Control (LCDCTL)** register. The following table shows the active external signals when this mode is active:

**Table 26-5. Operation Modes Supported by Raster Controller**

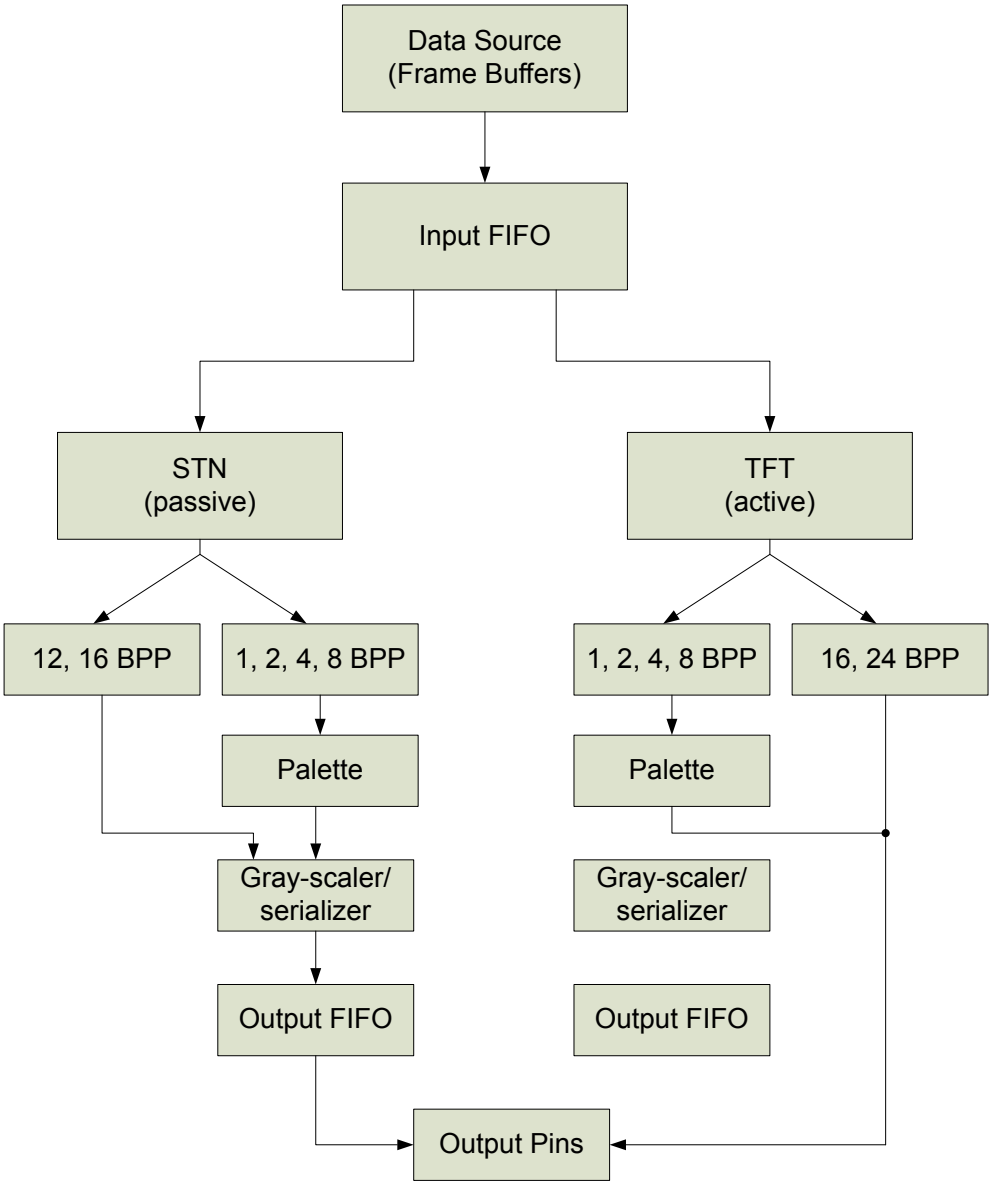
| Interface | Data Bus Width | RASTERCTRL [9, 7, 1] | Signal Name | Description |
|---|---|---|---|---|
| Passive (STN) Mono 4-bit | 4 | 001 | LCDDATA[3:0] | Data Bus |
| | | | LCDCP | Pixel Clock |
| | | | LCDLP | Horizontal clock (line clock) |
| | | | LCDFP | Vertical clock (frame clock) |
| | | | LCDAC | AC Bias |
| | | | LCDMCLK | Not used |
| Passive (STN) Mono 8-bit | 8 | 101 | LCDDATA[7:0] | Data Bus |
| | | | LCDCP | Pixel Clock |
| | | | LCDLP | Horizontal clock (line clock) |
| | | | LCDFP | Vertical clock (frame clock) |
| | | | LCDAC | AC Bias |
| | | | LCDMCLK | Not used |
| Passive (STN) Color | 8 | 100 | LCDDATA[7:0] | Data Bus |
| | | | LCDCP | Pixel Clock |
| | | | LCDLP | Horizontal clock (line clock) |
| | | | LCDFP | Vertical clock (frame clock) |
| | | | LCDAC | AC Bias |
| | | | LCDMCLK | Not used |
| Active (TFT) Color | 16 | x10 | LCDDATA[15:0] | Data Bus |
| | | | LCDCP | Pixel Clock |
| | | | LCDLP | Horizontal clock (line clock) |
| | | | LCDFP | Vertical clock (frame clock) |
| | | | LCDAC | AC Bias |
| | | | LCDMCLK | Not used |

#### 26.3.4.1 Logical Data Path

The block diagram of the Raster controller is shown in Figure 26-1 on page 1841. Figure 26-3 on page 1850 illustrates its logical data path for various operation modes (passive (STN) versus active (TFT) and various BPP size). Figure 26-3 on page 1850 shows that:

- The gray-scaler/serializer and output FIFO blocks are bypassed in active (TFT) modes.
- The palette is bypassed in both 12- and 16-BPP modes.

**Figure 26-3. LCD Raster Data Path**




In summary:

- The display image is stored in frame buffers.
- The built-in DMA engine constantly transfers the data stored in the frame buffers to the input FIFO.
- The Raster Controller relays data to the external pins according to the specified format.

### 26.3.5 LCD Frame Buffer

The LCD controller has the option to use the internal SRAM memory or external memory through the EPI interface to hold the frame buffer. The frame buffer is a contiguous memory block, storing enough data to fill a full LCD screen. The frame buffer contains the palette look-up table (palette RAM) and the frame data for a given source image. When moving data into the LCD, the PALMODE bit in the **LCD Raster Control (LCDRASTRCTL)** register can be configured for palette and data loading, palette loading only or data loading only.

**Note:** When using the LCD with EPI to interface to external memory, the external code address space 0x1000.0000 must be selected by programing the ECADR field to 0x1 in the **EPI Address Map (EPIADDRMAP)** register at EPI offset 0x01C.

### 26.3.6 Palette RAM

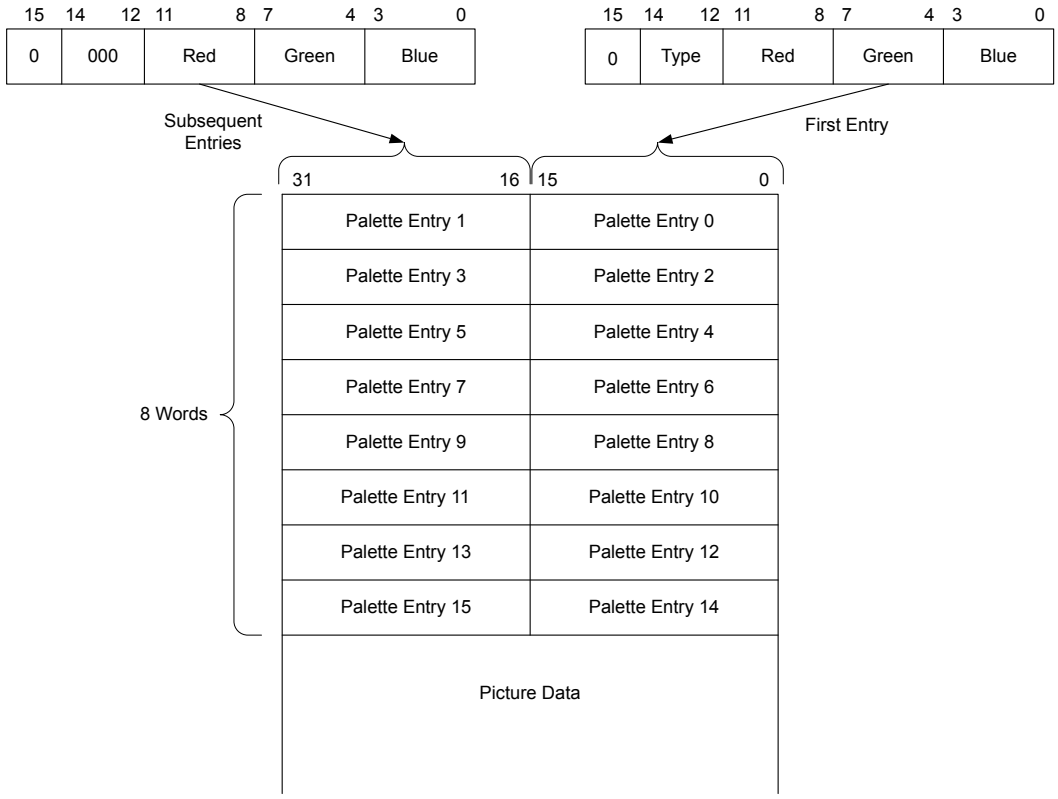
The palette RAM is a programmable cross reference table that maps a small set of input codes into a larger set of output codes. For active matrix displays, the palette RAM programmed in the LCD is used only for color source images. For passive matrix displays, the palette RAM is used for both grayscale and color source images. For color components, the passive matrix mode only supports, 4 bits per color component. 1, 2, 4, and 8 bit per pixel (bpp) source images always use the palette. 12, 16, and 24 bits per pixel do not use the palette RAM, since raw data transfers occur.

The LCD palette RAM uses a 12-bit output code such that there is a possibility of 4096 color options to choose from. Depending on the bpp frame storage of palette RAM, two, four, 16, or 256 palette colors can be stored in the palette RAM.

#### 26.3.6.1 Palette RAM Structure for 1, 2, and 4 Bits Per Pixel

When palette RAM is enabled for 1, 2, and 4 bpp frame storage, the palette buffer is 16 word entries, where the upper half of the word is unused. The first entry of the palette RAM contains the encoding for the palette type (1, 2, 3, 6, or 12/16/24) of the stored picture on bits [14:12] , as well as the first input code for one palette entry (bits [11:0]). Subsequent entries only contain the input codes for next palette entries. Figure 26-4 on page 1852 shows the frame buffer structure.

**Figure 26-4. Palette RAM Structure for 1, 2, and 4 Bits Per Pixel**




Note that for 1 bpp source pictures, the first two palette entries are valid; for 2 bpp sources, the first four palette entry sources are valid; for 4 bpp, all 16 palette entries are valid. Even if the palette entries are not valid for a particular type, there is a fixed number of 8 words reserved for the palette RAM.

The first palette entry represents the RGB value for the source pixel encoded as 0x0. The second palette entry represents the RBG value for the source pixel encoded as 0x1, and so on, such that the sixteenth palette entry is used to represent the RGB value for a pixel encoded as 0xF.

Source data encoded as 12, 16, or 24 bpp do not use the palette RAM since the encoded data already represents the actual RGB values. However, 16 palette RAM entries still precede the picture data in this case and the first palette entry must still define the bpp in bits [14:12]. The type field encoding is defined below in Table 26-6 on page 1852

**Table 26-6. Type Encoding in Palette Entry 0 Buffer**

| First Palette RAM Entry [14:12] | Source Image Type |
|---|---|
| 0x0 | 1 bpp |
| 0x1 | 2 bpp |
| 0x2 | 4 bpp |
| 0x3 | 8 bpp |
| 0x4 | 12, 16, or 24 bpp |

### 26.3.6.2 Palette RAM Structure for 8 Bits Per Pixel

For image sources that are 8 bpp, 256 palette entries ($2^8$) are required. This structure of the palette RAM is similar to the one described for 1, 2, 4, or 4 bpp except that the palette requires 256 entries (128 words), instead of 16. The first palette entry again contains the "type" definition of the picture along with the RGB values in bits [11:0].

#### *LCD RAM Configuration*

The LCD RAM can be configured to hold palette only, picture data only or both palette and picture data. Depending on the mode, the RAM is used differently:

- Palette Only: The PALMODE field in the **LCD Raster Control (LCDRASTRCTL)** register is programmed to 0x01. If the type field in the first palette entry is for 1, 2, 4, or 12/16/24 bpp, an 8 word palette buffer will be read from DDR by the DMA. If the bpp field in the first entry is 8 bpp, a 128 word palette will be loaded by the DMA. Note that the **DMA Frame Buffer n Base Address (LCDDMABAn)** and **LCD DMA Frame Buffer n Ceiling Address (LCDMACAn)** registers must encompass the palette information size. Note that although an 8 word buffer is created for 12/16/24 bpp, it is ignored during data transfers.

- Data Only: When PALMODE is programmed to 0x10, the bpp field from the last palette read is used. Thus, a single palette in DDR can be associated with multiple stored pictures in DDR, enabling the palette loading to occur once (PALMODE = 0x01) and subsequent data-only frame buffers to be transferred by the DMA into the RAM (PALMODE = 0x10). For 12/16/24 bpp frames the bpp field in the previously read palette RAM buffer is irrelevant.

- Palette and Data: When the PALMODE encoding is 0x00, the frame buffer is comprised of palette data followed immediately by pixel information. For 12, 14, and 24 bit (packed or unpacked) raw formats, an 8 word palette is prepended to the pixel data. Thus, palette information will be read from DDR even though it is not applicable data and the configuration will look similar to Figure 26-4 on page 1852.

#### *Raw Data and Color Component Formats*

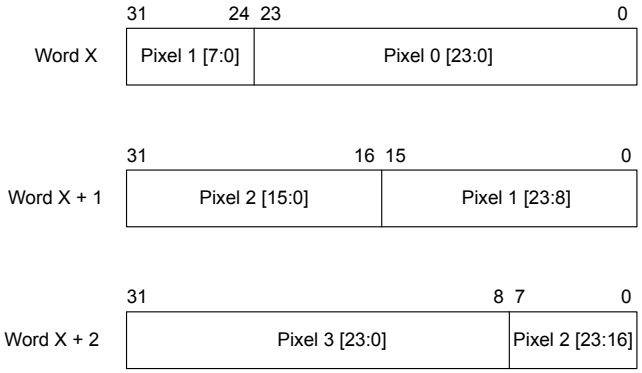
Depending on the value programmed in the raster control register you can have different color formats. Raw data formats are used for active matrix panels. The LCD controller supports, 24 bpp, 16 bpp and 12 bpp raw formats.

**Note:** The TYPE code in the first palette entry must be set to "1XX" for 12/16/24 bpp raw data formats.

#### *Data Format for 24 bpp*

For raw formats used in active matrix panels, the palette RAM provides a 12 bpp output, or a color depth of 12 bits (4096 colors per pixel). Data formats of 24 bpp can be stored in packed or unpacked format. In packed format, four 24 bit pixels are stored in 3 contiguous 32-bit words (also called a quad-pixel triplet) in DDR, as show in Figure 26-5 on page 1854.

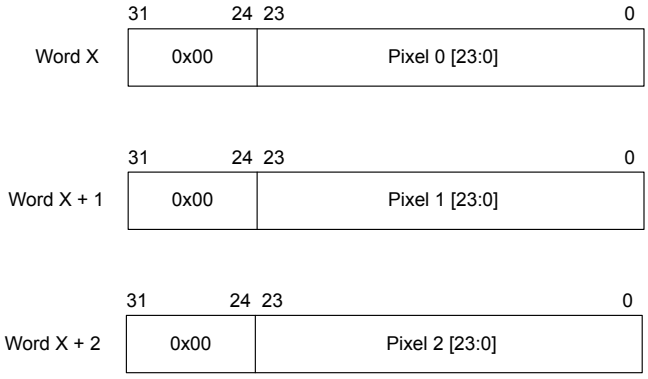
**Figure 26-5. 24 bpp Packed Data Format**




When using a 24 bit packed format, four consecutive pixels provide three word-aligned accesses for the DMA. If the MSBPPL and PPL bits in the **LCDRASTRTIM0** register are programmed to a multiple of 16, then the MSBLPP and LPP bits of the **LCDRASTRTIM1**and **LCDRASTRTIM2** register must follow a multiple of 16 so that the DMA can access the data using word-aligned accesses.

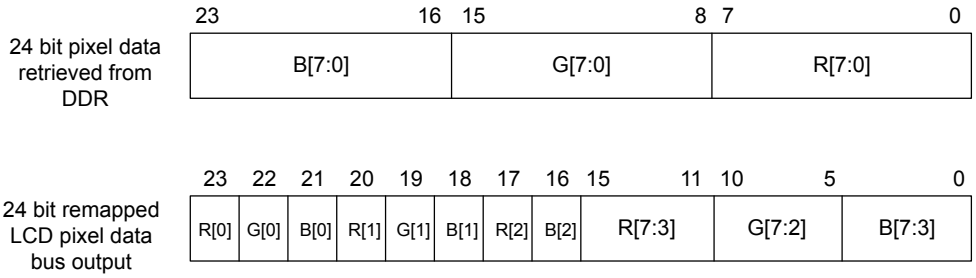
When LCDTFT = 0x1, TFT24UPCK = 0x1 and TFT24 = 0x1 in the **LCDRASTRCTL** register, 24 bit unpacked format is used. In this mode, each 24 bpp data is stored in a 32 bit word, with the upper 8 bits unused. Figure 26-6 on page 1854 shows the unpacked implementation.

**Figure 26-6. 24 bpp Unpacked Data Format**




For 24 bpp color components, data is stored in the DDR in B-G-R order with each color representing 8 bits. When the data is transferred to the LCD pixel data bus (LCDDATA[23:0]), the output is transposed such that a portion of the color components are now displayed on LCDDATA [15:0] in RGB565 format. The three least significant bits of the RGB 24-bit data are prepended to RGB565 data on LCDDATA[23:15]. The transposition of data is shown in Figure 26-7 on page 1855.

**Figure 26-7. 24 bpp Color RGB Remapping on LCDDATA[23:0]**




### *Data format for 16 bpp*

The LCD controller is configured to support 16 bpp raw data format when TFT24 = 0x0, LCDTFT = 0x1, TFTMAP = 0x0 in the **LCDRASTRCTL** register. Two 16 bpp raw data pixels can be stored in one word in DDR. The configuration is shown in Figure 26-8 on page 1855. For color component ordering, pixel data is in a RGB565 configuration and is output on the LCD pixel data bus LCDDATA [15:0] in the same order. LCDDATA bits [23:16] are undefined in 16 bpp mode. Figure 26-9 on page 1855 shows the configuration of the pixel data retrieved from memory and LCD bus output.

**Figure 26-8. 16 bpp Data Format**

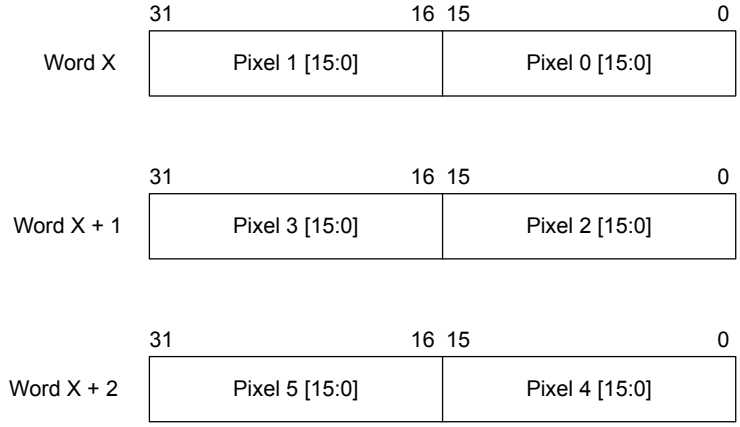
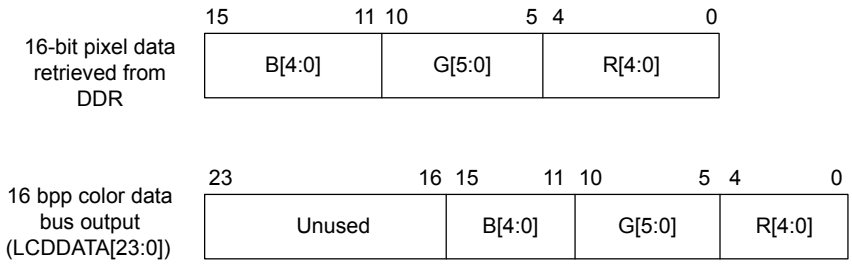


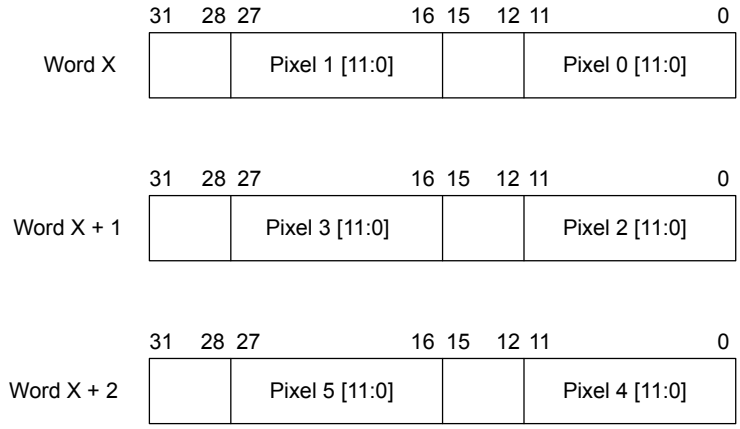

**Figure 26-9. 16 bpp Color Component Ordering**



### *Data Format for 12 bpp*

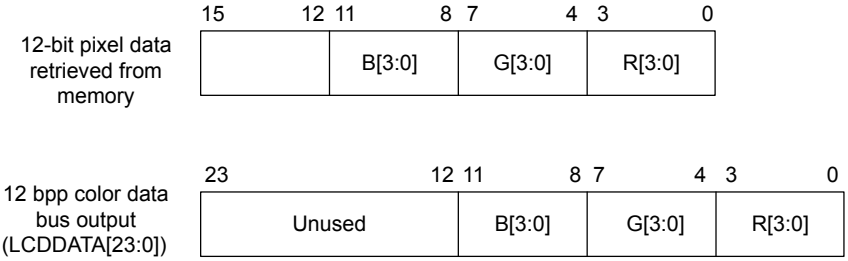
Two pixel data are stored in each 32-bit word in memory. The upper four bits in each 16-bit halfword are unused. Figure 26-10 on page 1856 shows the 12bpp data format.

**Figure 26-10. 12 bpp Data Format**




12 bpp color format has 4 pixels per Red, Green and Blue color component. The BGR configuration is output on LCDDATA [11:0] as shown in Figure 26-11 on page 1856. LCDDATA [23:12] are unused.

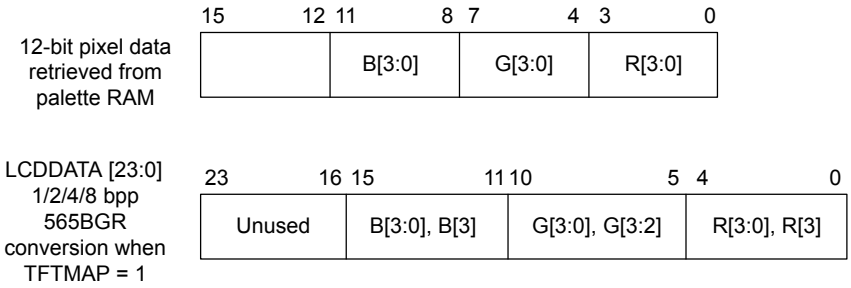
**Figure 26-11. 12 bpp Color Component Ordering**




### *1/2/4/8 bpp Data Format*

The Palette frame buffer for 1/2/4/8 bpp data supports passive and active matrix displays. Active matrix is enabled by setting the TFT bit in the **LCD Raster Control (LCDRASTRCTL)** register. When the TFTMAP is set to 0, the output from the palette look-up is a 12 bpp BGR pixel as shown in Figure 26-4 on page 1852, where each color is a 4 bit value. This output is sent to LCDDATA [11:0]. By setting TFTMAP to 1, the pixel data output is converted to a 16-bit BGR565 format. The extra least significant are dithered by replication of the component values' most significant bits. This conversion is shown in Figure 26-12 on page 1857.

**Figure 26-12. 1/2/4/8 bpp Dither Output**




## 26.3.7 Palette

As explained in the previous section, the pixel data is an index of palette entry (when palette is used). The number of colors supported is given by 2number of BPP. However, due to a limitation of the grayscaler/ serializer block, fewer grayscales or colors may be supported.

The PALMODE field in the **LCD Raster Control (LCDRASTRCTL)** register affects the palette loading:

- If PALMODE is 0x0 (palette-plus-data mode) or 0x1 (palette-only mode), the palette is loaded by the DMA engine at the very beginning, which is followed by the loading of pixel data.
- If PALMODE is 0x2 (data-only mode), the palette is not loaded. Instead, the DMA engine loads the pixel data immediately.

## 26.3.8 Gray-Scaler/Serializer - Passive (STN) Mode

Once a palette entry is selected from the look-up palette by the pixel data, its content is sent to the grayscaler/ serializer. If it is monochrome data, it is encoded as 4 bits. If it is color data, it is encoded as 4 bits (Red), 4 bits (Green), and 4 bits (Blue). These 4-bit values are used to select one of the 16 intensity levels, as shown in Table 13-10. A patented algorithm is used during this processing to provide an optimized intensity value that matches the eye's visual perception of color/gray gradations.

## 26.3.9 Gray-Scaler/Serializer - Active (TFT) Mode

The gray-scaler/serializer is bypassed in this mode.

## 26.3.10 Color/Grayscale Intensities and Modulation Rates

The following table details the intensities and modulation rates for varying dither values.

**Table 26-7. Intensities and Modulation Rates**

| Dither Value (4-Bit Value from Palette) | Intensity (0% is White) | Modulation Rate (Ratio of ON to ON+OFF Pixels) |
|---|---|---|
| 0x0 | 0.0% | 0 |
| 0x1 | 14.3% | 1/7 |
| 0x2 | 20.0% | 1/5 |
| 0x3 | 25% | 1/4 |
| 0x4 | 33.3% | 3/9 |
| 0x5 | 40.0% | 2/5 |
| 0x6 | 44.4% | 4/9 |

**Table 26-7. Intensities and Modulation Rates *(continued)***

| Dither Value (4-Bit Value from Palette) | Intensity (0% is White) | Modulation Rate (Ratio of ON to ON+OFF Pixels) |
|---|---|---|
| 0x7 | 50.0% | 1/2 |
| 0x8 | 55.6% | 5/9 |
| 0x9 | 60.0% | 3/5 |
| 0xA | 66.6% | 6/9 |
| 0xB | 75% | 3/4 |
| 0xC | 80.0% | 4/5 |
| 0xD | 85.7% | 6/7 |
| 0xE | 93.3% | 14/15 |
| 0xF | 100.0% | 1 |

## 26.3.11 Summary of Color Depth

summarizes the color depths depending on the number of BPP.

**Table 26-8. Number of Colors/Shades of Gray Available on Screen**

| Number of BPP | Passive Mode (LCDTFT = 0) | | Active Mode (LCDTFT = 1) |
|---|---|---|---|
| | Monochrome (LCDBW = 1) | Color (LCDBW = 0) | Color Only (LCDBW = 0) |
| 1 | 2 palette entries to select within 15 grayscales | 2 palette entries to select within 3375 possible colors | 2 palette entries to select within 4096 possible colors |
| 2 | 4 palette entries to select within 15 grayscales | 4 palette entries to select within 3375 possible colors | 4 palette entries to select within 4096 possible colors |
| 4 | 16 palette entries to select within 15 grayscales | 16 palette entries to select within 3375 possible colors | 16 palette entries to select within 4096 possible colors |
| 8 | Not relevant because it would consist in 256 palette entries to select within 15 grayscales, but exists anyways | 256 palette entries to select within 3375 possible colors | 256 palette entries to select within 4096 possible colors |
| 12 | X | 3375 possible colors | 4096 possible colors |
| 16 | X | 3375 possible colors (STN_565 = 1) | Up to 65536 possible colors |
| 24 | X | X | Up to 16.7 million colors |

## 26.3.12 Output Format

The following section describes the output format for both passive and active LCD modes.

### 26.3.12.1 Passive (STN) Mode

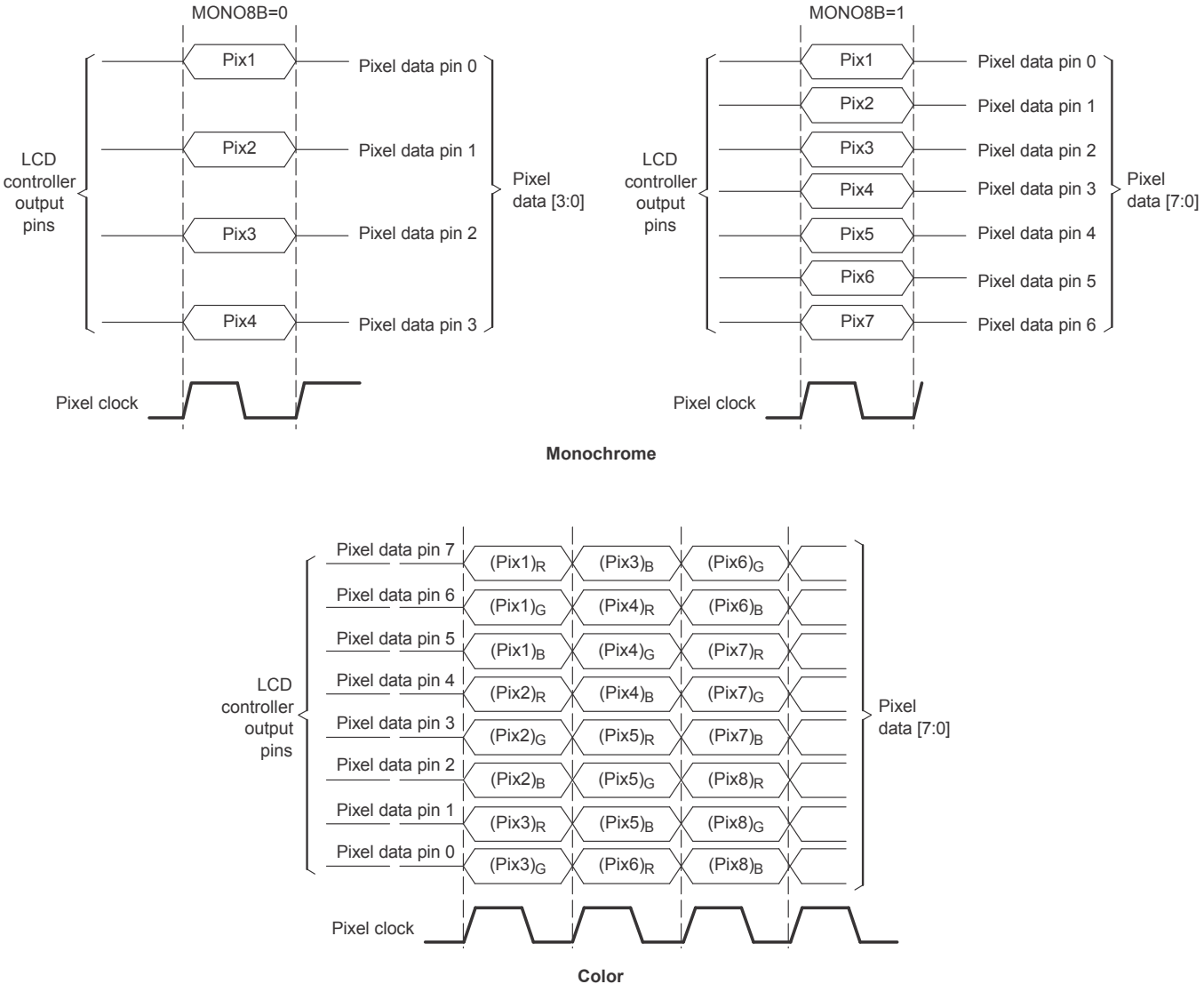
As shown inFigure 26-13 on page 1859, the pixel data stored in frame buffers go through palette (if applicable) and grayscaler/ serializer before reaching the Output FIFO. As a result, it is likely that the data fed to the output FIFO is numerically different from the data in the frame buffers. (However, they represent the same color or grayscale.)

The output FIFO formats the received data according to display modes (see Table 13-7). FigureFigure 26-13 on page 1859, shows the actual data output on the external pins.

### 26.3.12.2 Active (TFT) Mode

As shown inFigure 26-13 on page 1859, the gray-scaler/serializer and output FIFO are bypassed in active (TFT) mode. Namely, at each pixel clock, one pixel data (16 bits) is output to the external LCD.

**Figure 26-13. Monchrome and Color Output**




## 26.3.13 Subpicture Feature

A feature exists in the LCD to cover either the top or lower portion of the display with a default color. This feature is called a subpicture and is illustrated in Figure 26-14 on page 1860. Subpictures are only allowed for Active Matrix Mode (when LCDTFT = 0x1 in the **LCD Raster Control (LCDRASTRCTL)** register).

Subpicture reduce the bandwidth since lines containing default pixel data are not read from memory. For example, suppose the panel has 100 lines of which 50 are default pixel data lines. Then, only 50 lines of data are DMAed from DDR for this subpicture setup. That is, the **LCD DMA Frame Buffer**

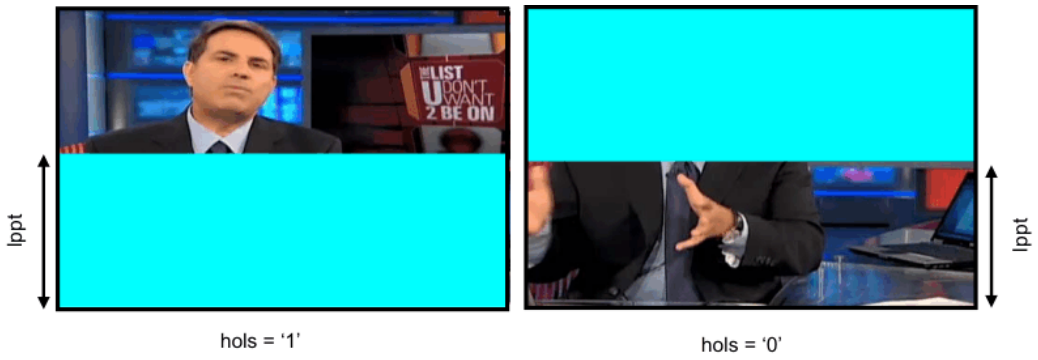
**n Base Address (LCDDMABAFBn)** and **LCD DMA Frame Buffer n Base Address (LCDDMABAFBn)** registers only encompass 50 lines of data instead of 100.

**Figure 26-14. Example of Subpicture**




The subpicture feature is enabled when the SPEN control bit in the **LCD Raster Subpanel Display n (LCDRASTRSUBPn)** register is set to 1. The HOLS bit, when programmed to 0, puts the Default Pixel Data lines at the top of the screen and the active video lines at the bottom of the screen. When the HOLS bit is set to 1, Active video lines are at the top of the screen and Default Pixel Data lines are at the bottom of the screen. The HOLS bit behavior is shown in Figure 26-15 on page 1860.

**Figure 26-15. Subpicture Output with HOLS Bit Variations**




The LPPT (lines per panel threshold) bitfield defines the number of lines at the bottom of the picture for both HOLS = 1 or 0. LPPT is an encoded value in the range {0:2047} used to represent the number of lines in the range {1:2048}.

## 26.4 Interrupts

The application can program the LCD controller to trigger an interrupt for the following events:

- DMA end-of-frame 1: The DMA end-of-frame 1 (EOF1) interrupt is triggered when the DMA module has completed transferring the contents of a bounded frame buffer 1.
- DMA end-of-frame 0: The DMA end-of-frame 0 (EOF0) interrupt is triggered when the DMA module has completed transferring the contents of a bounded frame buffer 0.

- DMA palette loaded: When PALMODE is set to Palette-only or Palette+data, the palette loaded interrupt is triggered when the palette portion of the DMA transfer has been stored in the Palette RAM.
- DMA FIFO Underflow: The FIFO Underflow (FIFOU) interrupt is triggered when the real-time output needs to send a value for pixel data but one cannot be found in the FIFO.
- AC Bias Count Decremented to Zero: For Passive Matrix displays, a count can be kept of the number of times the AC Bias line toggles. Once the specified number of transitions has been seen, the AC Bias Count (ACBS) interrupt is triggered. The module does not post any further interrupts or keep counting AC Bias transitions until the interrupt has been cleared.
- Frame synchronization lost: When the DMA module reads a frame buffer and stores it in the FIFO, it sets a start frame and an end frame indicator embedded with the data. On retrieving the data from the FIFO, the Sync Lost (SYNCS) interrupt is triggered if the start indicator is not found at the first pixel of a new frame.
- Recurrent Raster mode frame done: In raster mode, the Recurrent Frame Done (RRASTRDONE) interrupt is triggered each time a complete frame has been sent to the interface pins.
- Raster or LIDD frame done (shared interrupt): In LIDD DMA mode, a frame buffer of data is sent. When the frame buffer has completed, the LIDD Frame Done (DONE) interrupt is triggered. In order to do another LIDD DMA, the DMA engine must be disabled and then re-enabled.

  In Raster mode, the interrupt is triggered after LCDEN bit is programmed to 0 in the **LCD Raster Control (LCDRASTRCTL)** register and after the last frame is sent to the pins. After the Raster mode DMA is running, the interrupt occurs only once after the module is disabled.

The interrupts are enabled in the **LCD Interrupt Mask register (LCDIM)** register. All pending interrupts in the LCD module must be serviced by the Host's Interrupt Service Routine before it exits.

## 26.5 Bus Transaction Modes

The LCD Controller supports character-based LCD panels by ensuring the bus cycles for reading/writing commands and data are met. The LIDD (LCD Interface Display Driver) provides this support. The LIDD interfaces with an ARM CPU in the SOC using a VBUS slave port. Abundant programmable timing parameters can be configured such that the off-chip interface can support a wide variety of character based LCD panels. Alternatively, the DMA module can be used to mimic the CPU and perform a sequence of write-only data bus transactions to the character based LCD panel. The Motorola and Intel interfaces support asynchronous and synchronous modes. The only difference between the two is that, for synchronous mode, the internal bus clock is output on a pin that replaces CS1. In other words, only one character panel can be supported under synchronous mode while two panels can be supported for asynchronous mode.

## 26.6 Initialization and Configuration

Basic configuration of the LCD controller is as follows:

1. Enable the LCD Controller in the System Control module using the **RCGCLCD** register. See page 419.
2. Enable the DMA, core and LIDD controller (if required) in the **LCDCLKEN** register.

3. Select the LCD mode (LIDD or Raster) by programming the LCDMODE bit in the **LCDCTL** register. In addition, configure the LCDCP or LCDMCLK frequency through the CLKDIV field.

4. When operating in LIDD mode:

   a. Program the **LCDLIDDCTL** register along with the chip select configuration register, **LIDDCSnCFG**.

   b. Select the DMA parameters in the **LCDDMACTL** register. Program the frame buffer boundaries in the **LCDDMABAFB0** and **LCDDMACAFB0** registers.

   c. Enable any required interrupts in the **LCDIM** register.

   d. Initiate LIDD mode transactions by setting the DMAEN bit in the **LCDLIDDCTL** register.

5. When operating in Raster Mode:

   a. Program the **LCDRASTRCTL** register along with the raster timings in the **LCDRASTRTIMn** registers.

   b. Configure the subpanel display in the **LCDRASTRSUBPn** registers.

   c. Select the DMA parameters such as burst size, and memory layout in the **LCDDMACTL** register. Program the frame buffer boundaries in the **LCDDMABAFBn** and **LCDDMACAFBn** registers.

   d. Enable any required interrupts in the **LCDIM** register.

   e. Initiate Raster mode transactions by setting the LCDEN bit in the **LCDRASTRCTL** register.

The LCD DMA may interface to internal SRAM or the external memory through EPI for transfer of data. To configure the EPI for use with LCD, refer to “External Peripheral Interface (EPI)” on page 848.

**Note:** When using the LCD with EPI to interface to external memory, the external code address space 0x1000.0000 must be selected by programing the ECADR field to 0x1 in the **EPI Address Map (EPIADDRMAP)** register at EPI offset 0x01C.

## 26.7 Register Map

Table 26-9 on page 1862 lists the LCD registers. The offset listed is a hexadecimal increment to the register's address, relative to the LCD module's base address of 0x4405.0000.

**Table 26-9. LCD Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | LCDPID | RO | 0x4F20.1000 | LCD PID Register Format | 1864 |
| 0x004 | LCDCTL | RW | 0x0000.0000 | LCD Control | 1865 |
| 0x00C | LCDLIDDCTL | RW | 0x0000.0000 | LCD LIDD Control | 1867 |
| 0x010 | LIDDCS0CFG | RW | 0x0044.0044 | LCD LIDD CS0 Configuration | 1870 |
| 0x014 | LIDDCS0ADDR | RW | 0x0000.0000 | LIDD CS0 Read/Write Address | 1871 |
| 0x018 | LIDDCS0DATA | RW | 0x0000.0000 | LIDD CS0 Data Read/Write Initiation | 1872 |

**Table 26-9. LCD Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x01C | LIDDCS1CFG | RW | 0x0044.0044 | LIDD CS1 Configuration | 1873 |
| 0x020 | LIDDCS1ADDR | RW | 0x0000.0000 | LIDD CS1 Address Read/Write Initiation | 1874 |
| 0x024 | LIDDCS1DATA | RW | 0x0000.0000 | LIDD CS1 Data Read/Write Initiation | 1875 |
| 0x028 | LCDRASTRCTL | RW | 0x0000.0000 | LCD Raster Control | 1876 |
| 0x02C | LCDRASTRTIM0 | RW | 0x0000.0000 | LCD Raster Timing 0 | 1880 |
| 0x030 | LCDRASTRTIM1 | RW | 0x0000.0000 | LCD Raster Timing 1 | 1881 |
| 0x034 | LCDRASTRTIM2 | RW | 0x0000.0000 | LCD Raster Timing 2 | 1882 |
| 0x038 | LCDRASTRSUBP1 | RW | 0x0000.0000 | LCD Raster Subpanel Display 1 | 1885 |
| 0x03C | LCDRASTRSUBP2 | RW | 0x0000.0000 | LCD Raster Subpanel Display 2 | 1886 |
| 0x040 | LCDDMACTL | RW | 0x0000.0000 | LCD DMA Control | 1887 |
| 0x044 | LCDDMABAFB0 | RW | 0x0000.0000 | LCD DMA Frame Buffer 0 Base Address | 1889 |
| 0x048 | LCDDMACAFB0 | RW | 0x0000.0000 | LCD DMA Frame Buffer 0 Ceiling Address | 1890 |
| 0x04C | LCDDMABAFB1 | RW | 0x0000.0000 | LCD DMA Frame Buffer 1 Base Address | 1891 |
| 0x050 | LCDDMACAFB1 | RW | 0x0000.0000 | LCD DMA Frame Buffer 1 Ceiling Address | 1892 |
| 0x054 | LCDSYSCFG | RW | 0x0000.0028 | LCD System Configuration Register | 1893 |
| 0x058 | LCDRISSET | RW | 0x0000.0000 | LCD Interrupt Raw Status and Set Register | 1895 |
| 0x05C | LCDMISCLR | RW | 0x0000.0000 | LCD Interrupt Status and Clear | 1898 |
| 0x060 | LCDIM | RW | 0x0000.0000 | LCD Interrupt Mask | 1901 |
| 0x064 | LCDIENC | RW | 0x0000.0000 | LCD Interrupt Enable Clear | 1904 |
| 0x06C | LCDCLKEN | RW | 0x0000.0000 | LCD Clock Enable | 1907 |
| 0x070 | LCDCLKRESET | RW | 0x0000.0000 | LCD Clock Resets | 1908 |

## 26.8 Register Descriptions

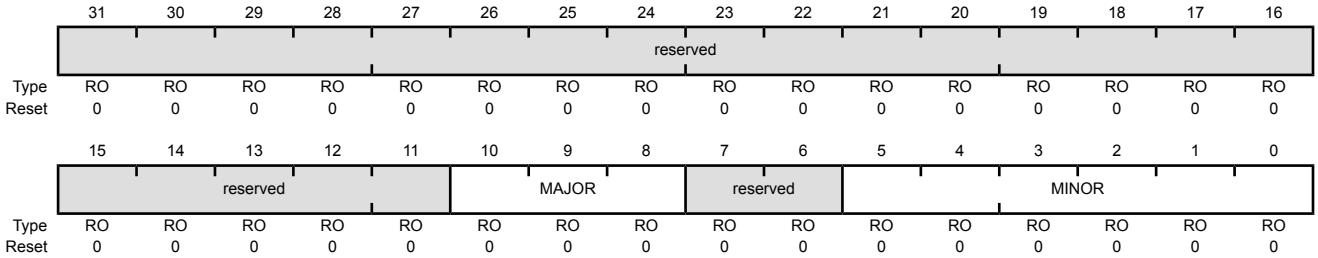
Following is a description of the individual registers of the LCD controller.

## Register 1: LCD PID Register Format (LCDPID), offset 0x000

This register contains information regarding the Peripheral ID (PID) of the LCD module.

LCD PID Register Format (LCDPID)

Base 0x4405.0000
Offset 0x000
Type RO, reset 0x4F20.1000

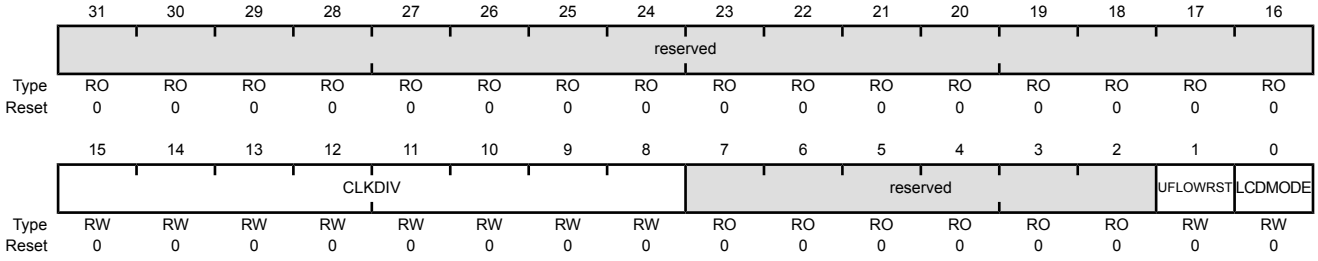



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:11 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10:8 | MAJOR | RO | 0x0 | Major Release Number<br>This field contains the major release number for the IP. |
| 7:6 | reserved | RO | 0 | 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. |
| 5:0 | MINOR | RO | 0x00 | Minor Release Number<br>This field contains the minor release number for the IP. |

## Register 2: LCD Control (LCDCTL), offset 0x004

The **LCD Control (LCDCTL)** register configures the mode, clock frequencies and restart behavior of the LCD Controller.

LCD Control (LCDCTL)
Base 0x4405.0000
Offset 0x004
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:8 | CLKDIV | RW | 0x00 | Clock Divisor<br>This field contains a value (from 0-255) used to specify the frequency of the pixel clock, LCDCP (in Raster Mode) or MCLK ( in LIDD mode).<br>The equation for the two clock outputs is: SYSCLK/CLKDIV, where:<br>■ LCDCP can range from SYSCLK/2 to SYSCLK/255. CLKDIV = 0x0 or CLKDIV = 0x1 are not allowed.<br>■ MCLK can vary from SYSCLK to SYSCLK/255 (using CLKDIV = 0x0 or CLKDIV = 0x1 sets MCLK = SYSCLK.<br>See Table 26-10 on page 1866 for $f_{SYSCLK}$ to $f_{LCDCP}$ frequency conversions based on CLKDIV value. |
| 7:2 | reserved | RO | 0x00 | 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. |
| 1 | UFLOWRST | RW | 0 | Underflow Restart<br>Value Description<br>0 On an underflow, software must restart the module.<br>1 On an underflow, the hardware restarts the module on the next frame. |
| 0 | LCDMODE | RW | 0 | LCD Mode Select<br>Value Description<br>0 LCD Controller is operating in LIDD Mode<br>1 LCD Controller is operating in Raster Mode |

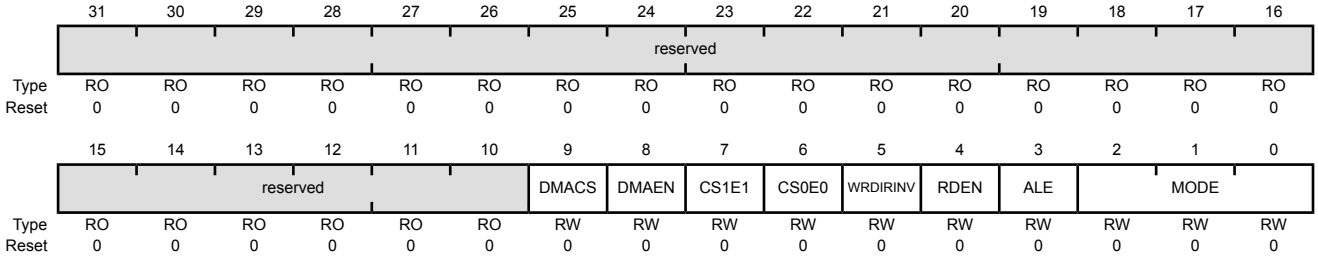
**Table 26-10. SYSCLK to Pixel Clock (LCDCP) Frequency Conversion Table**

| $f_{SYSCLK}$ (MHz) | CLKDIV Value | $f_{LCDCP}$ (MHz) |
|---|---|---|
| 120 | 2 | 60 |
| | 3 | 40 |
| | 4 | 30 |
| | 5 | 24 |
| | 6 | 20 |
| | 7 | 17.1428 |
| | 8 | 15 |
| | ... | ... |
| | N | $f_{SYSCLK}/N$ |
| 80 | 2 | 40 |
| | 3 | 26.666 |
| | 4 | 20 |
| | 5 | 16 |
| | 6 | 13.333 |
| | 7 | 11.428 |
| | ... | ... |
| | N | $f_{SYSCLK}/N$ |
| 60 | 2 | 30 |
| | 3 | 20 |
| | 4 | 15 |
| | 5 | 12 |
| | 6 | 10 |
| | N | $f_{SYSCLK}/N$ |

## Register 3: LCD LIDD Control (LCDLIDDCTL), offset 0x00C

This register configures the functionality of the LCD controller interface when it is programmed to function in LIDD mode.

LCD LIDD Control (LCDLIDDCTL)
Base 0x4405.0000
Offset 0x00C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9 | DMACS | RW | 0 | CS0/CS1 Select for LIDD DMA Writes<br><br>Value Description<br>0 DMA writes to LIDD CS0 (LCDAC).<br>1 DMA writes to LIDD CS1 (LCDMCLK).<br><br>**Note:** In Motorola 6800 or Intel 8080 synchronous modes, CS1 is not used and thus, DMACS should not be set to 0x1. |
| 8 | DMAEN | RW | 0 | LIDD DMA Enable<br><br>Value Description<br>0 Deactivate DMA control of LIDD interface.<br>DMA control is released upon completion of transfer of the current frame of data (LIDD Frame Done) when this bit is clear.<br>The microcontroller has direct read/write access to the panel in this mode.<br>1 Activate DMA to drive LIDD interface to support streaming data to smart panels.<br>The microcontroller cannot access the panel directly in this mode. |
| 7 | CS1E1 | RW | 0 | Chip Select 1 (CS1)/Enable 1(E1) Polarity Control<br>CS1 is active low by default. EN1 is active high by default.<br><br>Value Description<br>0 CS1/E1 (LCDMCLK) are not inverted. CS1 remains active low and E1 remains active high.<br>1 Invert CS1/E1 (LCDMCLK). CS1 is active high and E1 is active low. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | CS0E0 | RW | 0 | Chip Select 0 (CS0)/Enable 0 (E0) Polarity Control<br>CS0 is active low by default. E0 is active high by default.<br><br>Value Description<br>0 CS0/E0 (LCDAC) are not inverted. CS0 remains active low and E0 remains active high.<br>1 Invert CS0/E0. CS0 is active high and E0 is active low. |
| 5 | WRDIRINV | RW | 0 | Write Strobe (WR) /Direction (DIR) Polarity Control<br>WR active low by default. DIR is write low/read high by default.<br><br>Value Description<br>0 WR/DIR (LCDLP) are not inverted. WR remains active low and DIR is write low/read high.<br>1 Invert WR/DIR. WR is active high and DIR is write high/read low. |
| 4 | RDEN | RW | 0 | Read Strobe (RD) /Direct Enable (EN) Polarity Control<br>RD is active low by default. EN is active high by default.<br><br>Value Description<br>0 RD/EN (LCDCP) are not inverted. RD is active low and EN is active high.<br>1 Invert RD/EN (LCDCP). RD is active high and EN is active low. |
| 3 | ALE | RW | 0 | Address Latch Enable (ALE) Polarity Control<br>ALE is active low by default.<br><br>Value Description<br>0 ALE (LCDFP) is not inverted.<br>1 Invert ALE (LCDFP) . ALE is active high. |
| 2:0 | MODE | RW | 0x0 | LIDD Mode Select.<br>Selects type of LCD display interface for the LIDD to drive. See Table 26-3 on page 1843 for external pin assignments based on MODE encoding.<br><br>Value Description<br>0x0 Synchronous Motorola 6800 Mode<br>0x1 Asynchronous Motorola 6800 Mode<br>0x2 Synchronous Intel 8080 mode<br>0x3 Asynchronous Intel 8080 mode<br>0x4 Asynchronous Hitachi mode<br>0x5-0x7 reserved |

The MODE bit field described above defines the function of the LCD external pins as follows:

**Table 26-11. LCD Alternate Signal Functions in LIDD Mode**

| PIN | MODE = 0x0 | MODE = 0x1 | MODE = 0x2 | MODE = 0x3 | MODE = 0x4 |
|---|---|---|---|---|---|
| LCDCP | RS/WS | RS/WS | RS | RS | N/A |
| LCDLP | DIR | DIR | WS | WS | DIR |
| LCDFP | ALE | ALE | ALE | ALE | ALE |
| LCDAC | CS0 | CS0 | CS0 | CS0 | E0 |
| LCDMCLK | MCLK | CS1 | MCLK | CS1 | E1 |

## Register 4: LCD LIDD CS0 Configuration (LIDDCS0CFG), offset 0x010

The **LIDD CS0 Configuration (LIDDCS0CFG)** register defines the timings for the read and write strobes with respect to CS0.

LCD LIDD CS0 Configuration (LIDDCS0CFG)

Base 0x4405.0000
Offset 0x010
Type RW, reset 0x0044.0044

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRSU | | | | | WRDUR | | | | | | WRHOLD | | | | RDSU |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RDSU | | | | RDDUR | | | | | | RDHOLD | | | | GAP | |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

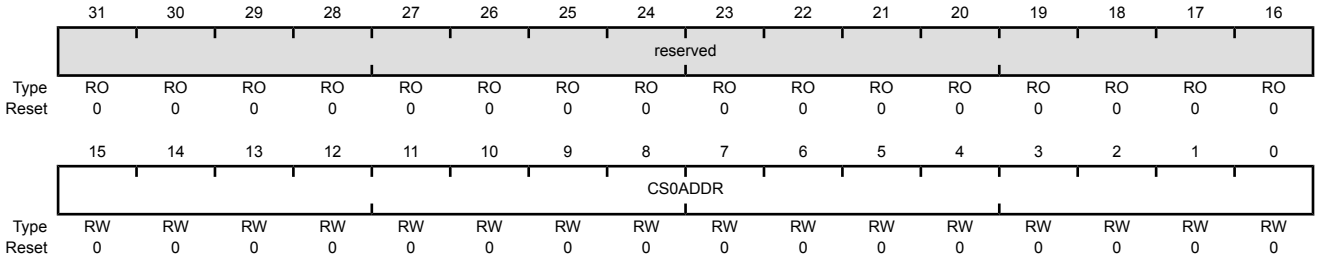
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:27 | WRSU | RW | 0x0 | Write Strobe (WR) Set-Up Cycles<br>When performing a write access, this field defines the number of MCLK cycles that the LCDDATA bus, output enable, ADE, DIR and CS0 signals have to be ready before the write strobe. The minimum value is 0x0. |
| 26:21 | WRDUR | RW | 0x2 | Write Strobe (WR) Duration Cycles<br>This field value defines the number of MCLK cycles for which the write strobe is held active when performing a write access. The minimum value is 0x1. |
| 20:17 | WRHOLD | RW | 0x2 | Write Strobe (WR) Hold cycles.<br>This field value defines the number of MCLK cycles for which the LCDDATA bus, output enable, ALE, DIR and CS0 signals are held after the write strobe is deasserted when performing a write access. The minimum value is 0x1. |
| 16:12 | RDSU | RW | 0x0 | Read Strobe (RD) Set-Up cycles.<br>When performing a read access, this field defines the number of MCLK cycles that the LCDDATA bus, output enable, ALE, DIR and CS0 signals have to be ready before the read strobe is asserted. |
| 11:6 | RDDUR | RW | 0x1 | Read Strobe (RD) Duration cycles.<br>This field defines the number of MCLK cycles for which the read strobe is held active when performing a read access. The minimum value is 0x1. |
| 5:2 | RDHOLD | RW | 0x1 | Read Strobe (RD) Hold cycles.<br>This field defines of MCLK cycles for which the LCDDATA bus, output enable, ALE, DIR and CS0 signals are held after the read strobe is deasserted when performing a read access. The minimum value is 0x1. |
| 1:0 | GAP | RW | 0x0 | Field value defines the number of LCDMCLK cycles (GAP +1) between the end of one CS0 (LCDAC) device access and the start of another CS0 (LCDAC) device access unless the two accesses are both reads. In this case, this delay is not incurred. The minimum value is 0x0. |

## Register 5: LIDD CS0 Read/Write Address (LIDDCS0ADDR), offset 0x014

This register contains the read and write address of the current access enabled by CS0 (LCDAC).

LIDD CS0 Read/Write Address (LIDDCS0ADDR)

Base 0x4405.0000
Offset 0x014
Type RW, reset 0x0000.0000



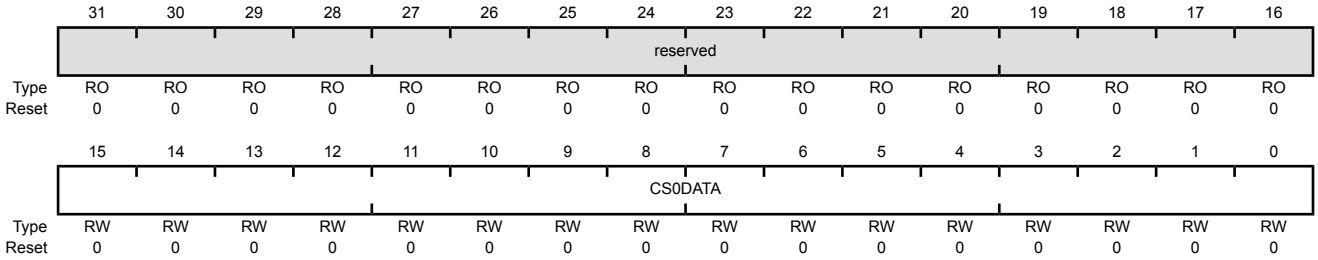

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | CS0ADDR | RW | 0x0000 | LCD Address<br>The LCD Controller supports a shared Address/Data output bus. A write to this register initiates a bus write transaction. A read from this register initiates a bus read transaction.<br>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the **LIDDCTRL** register). If the LIDD is being used as a generic bus interface, writing to this register can store CS0ADDR to an external transparent latch holding a 16-bit address.<br>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the command instruction area of the panel. |

## Register 6: LIDD CS0 Data Read/Write Initiation (LIDDCS0DATA), offset 0x018

This register contains the read and write data of the current access enabled by CS0 (LCDAC).

LIDD CS0 Data Read/Write Initiation (LIDDCS0DATA)

Base 0x4405.0000
Offset 0x018
Type RW, reset 0x0000.0000

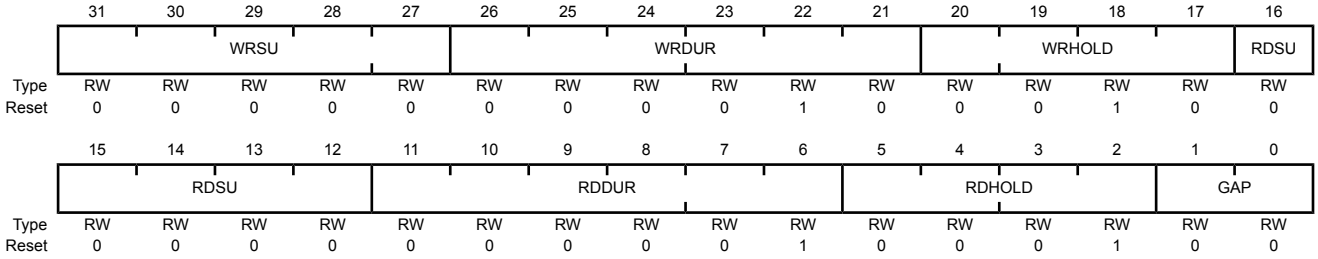



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | CS0DATA | RW | 0x0000 | LCD Data Read/Write<br>The LCD Controller supports a shared Address/Data output bus. A write to this register initiates a bus write transaction. A read from this register initiates a bus read transaction.<br>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the **LIDDCTRL** register).<br>If the LIDD is being used as a generic bus interface, writing to this register can store CS0ADDR to an external transparent latch holding a 16-bit address.<br>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the data area of the panel. |

## Register 7: LIDD CS1 Configuration (LIDDCS1CFG), offset 0x01C

The **LIDD CS1 Configuration (LIDDCS1CFG)** register defines the timings for the read and write strobes with respect to CS1.

LIDD CS1 Configuration (LIDDCS1CFG)
Base 0x4405.0000
Offset 0x01C
Type RW, reset 0x0044.0044



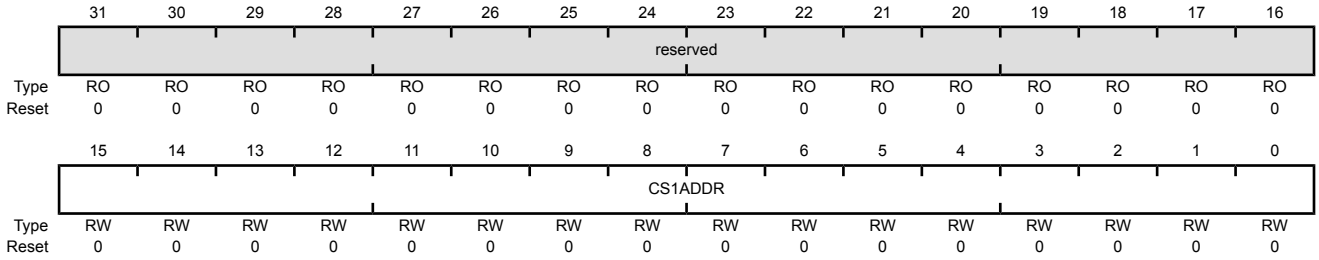

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:27 | WRSU | RW | 0x0 | Write Strobe (WR) Set-Up Cycles<br>When performing a write access, this field defines the number of LCDMCLK cycles that Data Bus/Pad Output Enable, ALE, DIR, and CS1 have to be ready before WR (LCDLP) is asserted. The minimum value is 0x0. |
| 26:21 | WRDUR | RW | 0x2 | Write Strobe (WR) Duration Cycles<br>Field value defines the number of LCDMCLK cycles for which WR (LCDLP) is held active when performing a write access. The minimum value is 0x1. |
| 20:17 | WRHOLD | RW | 0x2 | Write Strobe (WR) Hold cycles.<br>Field value defines the number of LCDMCLK cycles for which Data Bus/Pad Output Enable, ALE, the DIR, and CS1 signals are held after WR (LCDLP) is deasserted when performing a write access. The minimum value is 0x1. |
| 16:12 | RDSU | RW | 0x0 | Read Strobe (RD) Set-Up cycles.<br>When performing a read access, this field defines the number of LCDMCLK cycles that Data Bus/Pad Output Enable, ALE, the DIR, and CS1 signals have to be ready before RD (LCDCP) is asserted. |
| 11:6 | RDDUR | RW | 0x1 | Read Strobe (RD) Duration cycles.<br>Field value defines the number of LCDMCLK cycles for which RD (LCDCP) is held active when performing a read access. The minimum value is 0x1. |
| 5:2 | RDHOLD | RW | 0x1 | Read Strobe (RD) Hold cycles.<br>Field value defines the number of LCDMCLK cycles for which Data Bus/Pad Output Enable, ALE, DIR, and CS1 signals are held after RD (LCDCP) is deasserted when performing a read access. The minimum value is 0x1. |
| 1:0 | GAP | RW | 0x0 | Field value defines the number of LCDMCLK cycles (GAP + 1) between the end of one CS1 (LCDAC) device access and the start of another CS0 (LCDAC) device access unless the two accesses are both reads. In this case, this delay is not incurred. The minimum value is 0x0. |

## Register 8: LIDD CS1 Address Read/Write Initiation (LIDDCS1ADDR), offset 0x020

This register contains the read and write address of the current access enabled by CS1 (LCDAC).

LIDD CS1 Address Read/Write Initiation (LIDDCS1ADDR)

Base 0x4405.0000
Offset 0x020
Type RW, reset 0x0000.0000



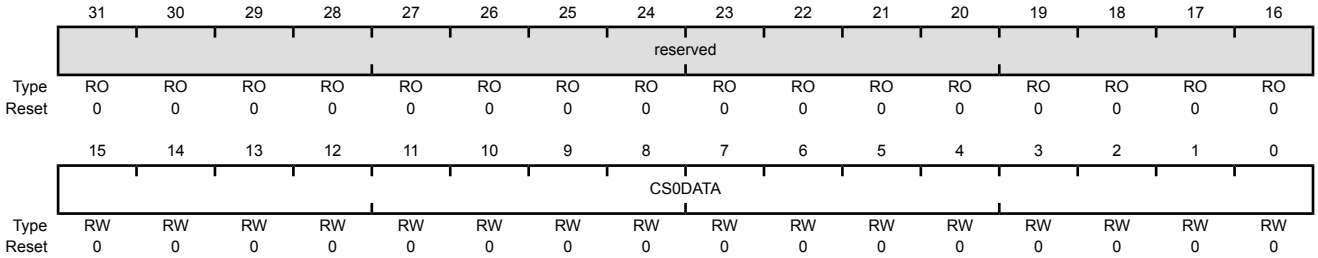

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | CS1ADDR | RW | 0x0000 | LCD Address Bus<br>The LCD Controller supports a shared Address/Data output bus. A write to this register would initiate a bus write transaction. A read from this register would initiate a bus read transaction.<br>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the **LIDDCTRL** register). If the LIDD is being used as a generic bus interface, writing to this register can store CS1ADDR to an external transparent latch holding a 16-bit address.<br>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the command instruction area of the panel. |

## Register 9: LIDD CS1 Data Read/Write Initiation (LIDDCS1DATA), offset 0x024

What is the difference between this register an the above register

LIDD CS1 Data Read/Write Initiation (LIDDCS1DATA)

Base 0x4405.0000
Offset 0x024
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | CS0DATA | RW | 0x0000 | LCD Data Read/Write Initiation<br>The LCD Controller supports a shared Address/Data output bus. A write to this register would initiate a bus write transaction. A read from this register would initiate a bus read transaction.<br>CPU reads and writes to this register are not permitted if the LIDD module is in DMA mode (DMAEN bit set in the **LIDDCTRL** register). If the LIDD is being used as a generic bus interface, writing to this register can store CS1ADDR to an external transparent latch holding a 16-bit address.<br>If the LIDD is being used to interface with a character based LCD panel in Hitachi configuration mode, reading and writing to this register can be used to access the data area of the panel. |

## Register 10: LCD Raster Control (LCDRASTRCTL), offset 0x028

The **LCD Raster Control (LCDRASTRCTL)** register is used to configure the features of Raster Mode.

LCD Raster Control (LCDRASTRCTL)

Base 0x4405.0000
Offset 0x028
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | TFT24UPCK | TFT24 | FRMBUFSZ | TFTMAP | NIBMODE | PALMODE | | REQDLY | | | |
| Type | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REQDLY | | | | reserved | | MONO8B | RDORDER | LCDTFT | reserved | | | | | LCDBW | LCDEN |
| Type | RW | RW | RW | RW | RO | RO | RW | RW | RW | RO | RO | RO | RO | RO | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:27 | reserved | RO | 0x0 | 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. |
| 26 | TFT24UPCK | RW | 0 | 24-bit TFT Mode Packing<br>This bit is only used when TFT24 and LCDTFT are both set to 1.<br>If TFT24UPCK is clear, 24-bit pixels are packed in 32-bit boundaries which means four pixels are saved in every three words, as shown below.<br>Word0: pix1[7:0],pix0[23:0]<br>Word1: pix2[15:0],pix1[23:8]<br>Word2: pix3[23:0],pix2[23:16]<br>Refer to Figure 26-5 on page 1854 for information on how the pixels are packed.<br>If this bit is set to 1, then 24-bit pixels are stored unpacked in DDR with the uppermost byte unused, as shown below:<br>Word0: Unused[7:0],pix0[23:0]<br>Word1: Unused[7:0],pix1[23:0]<br>Word2: Unused[7:0],pix2[23:0]<br>Word3: Unused[7:0], pix3[23:0]<br><br>Value Description<br>0 24-bit pixels are packed into 32 bit boundaries, which means 4 pixels are saved in every three words<br>1 24-bit pixels are stored unpacked in DDR with the uppermost byte unused |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25 | TFT24 | RW | 0 | 24-Bit TFT Mode<br><br>Value Description<br>0 24-Bit TFT Mode disabled. Palette RAM lookup is used for output pixel data.<br>1 24-Bit TFT Mode enabled. 24-bit data in TFT Active mode. The format of the framebuffer data depends on TFT24UPCK. |
| 24 | FRMBUFSZ | RW | 0 | Frame Buffer Select<br><br>This mode is valid when LCDTFT is 0 and there are 16 bpp raw data frame buffers (bpp=00)<br><br>Only for this case, this bit selects whether the frame buffer format is 16 bpp 565 or 12bpp.<br><br>The Grayscaler can only take 12 bits per pixel. The frame buffer data is 16 bits per pixel 565 when FRMBUFSZ is set to 1 and only the 4 most significant bits of each color component are sent to the Grayscaler input.<br><br>Value Description<br>0 Framebuffer is 12 bpp packed in bits [11:0]<br>1 Framebuffer is 16 bpp 565 |
| 23 | TFTMAP | RW | 0 | TFT Mode Alternate Signal Mapping for Palettized Framebuffer<br><br>This bit be 0 for all 12/16/24-bpp raw data formats. This bit can only be 1 for 1/2/4/8-bpp Palette Lookup data. Valid only in Active Matrix mode when LCDTFT=1.<br><br>Value Description<br>0 4 bits per component output data for 1, 2, 4, and 8 bpp modes will be right aligned on LCDDATA[11:0]<br>1 4 bits per component output data for 1, 2, 4, and 8 bpp will be converted to 5,6,5, format and use LCDDATA[15:0]={R3 R2 R1 R0 R3 G3 G2 G1 G0 G3 G2 B3 B2 B1 B0 B3} |
| 22 | NIBMODE | RW | 0 | Nibble Mode<br><br>This bit is used to determine palette indexing and is used in conjunction with RDORDER.<br><br>Value Description<br>0 Nibble mode is disabled<br>1 Nibble mode is enabled |
| 21:20 | PALMODE | RW | 0x0 | Pallette Loading Mode<br><br>For Raw Data (12/16/24 bpp) frame buffers, no Palette lookup is employed. Thus, these frame buffers use the Data-Only loading mode.<br><br>Value Description<br>0x0 Palette and data loading, reset value<br>0x1 Palette loading only<br>0x2 Data loading only. |

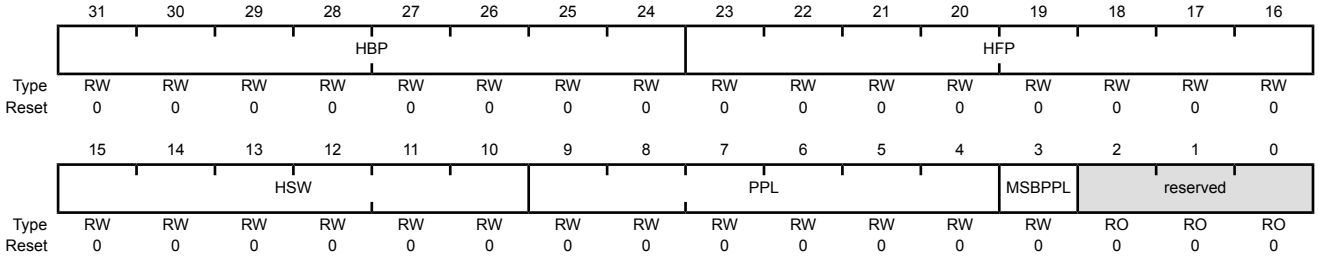
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 19:12 | REQDLY | RW | 0x00 | Palette Loading Delay<br>This 8-bit parameter pauses reading of the Palette data from the asynchronous FIFO between each burst of 16 words. The delay is in terms of system clock (SYSCLK) cycles. The value (0-255) used to specify the number of system clock cycles that should be paused between bursts of 16 word reads from the asynchronous FIFO while loading the Palette SRAM. Programming REQDLY=0x00 disables this pause when loading the Palette table.<br>When loading the Palette from system memory, palette data is burst into the internal Palette SRAM from the Asynchronous FIFO. 1, 2, and 4-bit per pixel frame buffer encodings use a fixed 16-word entry palette residing above the video data.<br>The 8-bit per pixel frame buffer encoding uses a 256-word entry palette residing above the video data. Likewise, 12, 16, and 24-bit per pixel frame buffer encodings also define a 256-word entry palette even though these encodings will not do a full bit-depth palette lookup. However, the 256-word palette entry must still be read from DDR as a frame buffer is fetched.<br>Bursting in 256 words in sequential SYSCLK cycles may cause the asynchronous FIFO to underflow depending on the DDR burst bandwidth. |
| 11:10 | reserved | RO | 0x0 | 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. |
| 9 | MONO8B | RW | 0 | Mono 8-Bit<br>This bit is ignored in all other modes.<br>Value Description<br>0 lcd_pixel_o[3:0] is used to output four pixel values to the panel each pixel clock transition<br>1 lcd_pixel_o[7:0] is used to output eight pixel values to the panel each pixel clock transition. |
| 8 | RDORDER | RW | 0 | Raster Data Order Select<br>For 1, 2, 4, and 8 BPP framebuffers.<br>This bit has no effect on raw data framebuffers (12/16/24 bpp). This bit is used to determine palette indexing and is used in conjunction with NIBMODE.<br>Value Description<br>0 The framebuffer parsing for Palette Data lookup is from Bit 0 to bit 31 of the input word from the DMA output.<br>1 The famebuffer parsing for Palette Data lookup is from Bit 31 to Bit 0 of the input word from the DMA output. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | LCDTFT | RW | 0 | LCD TFT<br>Value Description<br>0 Passive or STN display operation enabled; dither logic is enabled.<br>1 Active or TFT display operation enabled, external palette and DAC required, dither logic bypassed, pin timing changes to support continuous pixel clock, output enable, vsync, and hsync. |
| 6:2 | reserved | RO | 0x0 | 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. |
| 1 | LCDBW | RW | 0 | LCD Monochrome<br>Only applies for passive matrix panels.<br>Value Description<br>0 Color operation enabled<br>1 Monochrome operation enabled |
| 0 | LCDEN | RW | 0 | LCD Controller Enable for Raster Operations<br>This bit does not affect LIDD mode behavior.<br>Value Description<br>0 LCD controller disabled<br>1 LCD controller enabled |

## Register 11: LCD Raster Timing 0 (LCDRASTRTIM0), offset 0x02C

LCD Raster Timing 0 (LCDRASTRTIM0)

Base 0x4405.0000
Offset 0x02C
Type RW, reset 0x0000.0000

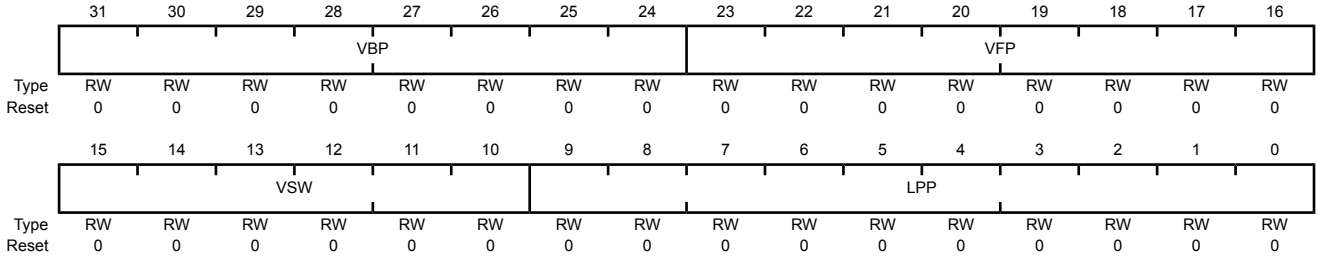



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | HBP | RW | 0x00 | Horizontal Back Porch Lowbits<br>Bits 7:0 of the horizontal back porch field.<br>Encoded value (from 1-1024) used to specify the number of pixel clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display (programmed value plus 1). Note that pixel clock is held in its inactive state during the beginning of the line wait period in passive display mode, and is permitted to transition in active display mode. |
| 23:16 | HFP | RW | 0x00 | Horizontal Front Porch Lowbits<br>Encoded value (from 1-1024) used to specify the number of pixel clock periods to add to the end of a line transmission before line clock is asserted (programmed value plus 1). Note that pixel clock is held in its inactive state during the end of line wait period in passive display mode, and is permitted to transition in active display mode. |
| 15:10 | HSW | RW | 0x00 | Horizontal Sync Pulse Width Lowbits<br>Bits 5:0 of the horizontal sync pulse width field.<br>Encoded value (from 1-1024) used to specify the number of pixel clock periods to pulse the line clock at the end of each line (programmed value plus 1). Note that pixel clock is held in its inactive state during the generation of line clock in passive display mode, and is permitted to transition in active display mode. |
| 9:4 | PPL | RW | 0x00 | Pixels-per-line LSB[9:4]<br>The PPL field is set to the (number of horizontal pixels -1) divided by 16. |
| 3 | MSBPPL | RW | 0 | Pixels-per-line MSB[10]<br>Needed in order to support up to 2048 ppl. |
| 2:0 | reserved | RO | 0x0 | 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. |

## Register 12: LCD Raster Timing 1 (LCDRASTRTIM1), offset 0x030

LCD Raster Timing 1 (LCDRASTRTIM1)

Base 0x4405.0000
Offset 0x030
Type RW, reset 0x0000.0000

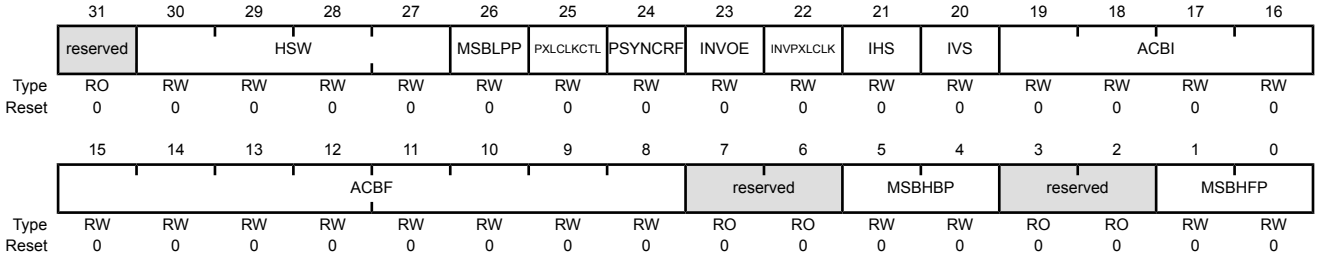



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | VBP | RW | 0x00 | Vertical Back Porch<br>Value (from 0-255) use to specify the number of line clock periods to add to the beginning of a frame before the first set of pixels is output to the display. Note that line clock transitions during the insertion of the extra line clock periods. |
| 23:16 | VFP | RW | 0x00 | Vertical Front Porch<br>Value (from 0-255) used to specify the number of line clock periods to add to the end of each frame. Note that the line clock transitions during the insertion of the extra line clock periods. |
| 15:10 | VSW | RW | 0x00 | Vertical Sync Width Pulse<br>In active mode (LCDTFT=1), encoded value (from 1-64) used to specify the number of line clock periods to set the lcd_fp pin active at the end of each frame after the (vfp) period elapses. The number of clock cycles is programmed value plus one. The frame clock is used as the VSYNC signal in active mode.<br>In passive mode (LCDTFT=0), encoded value (from 1-64) used to specify the number of extra line clock periods to insert after the vertical front porch (vfp) period has elapsed. Note that the width of lcd_fp is not affected by vsw in passive mode and line clock transitions during the insertion of the extra line clock periods (programmed value plus one). |
| 9:0 | LPP | RW | 0x00 | Lines Per Panel<br>Encoded value (programmed value range of {0:2047} represents an actual range of {1:2048}) used to specify the number of lines per panel. It represents the total number of lines on the LCD (programmed value plus one). Bit 10 of this field is in **LCDRASTRTIM2**. |

## Register 13: LCD Raster Timing 2 (LCDRASTRTIM2), offset 0x034

LCD Raster Timing 2 (LCDRASTRTIM2)

Base 0x4405.0000
Offset 0x034
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | reserved | RO | 0 | 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. |
| 30:27 | HSW | RW | 0x0 | Bits 9:6 of the horizontal sync width field |
| 26 | MSBLPP | RW | 0 | MSB of Lines Per Panel<br>Bit 10 of the LPP field in **LCDRASTRTIM1**. |
| 25 | PXLCLKCTL | RW | 0 | Hsync/Vsync Pixel Clock Control On/Off<br>Note that this bit MUST be programmed to 0 for Passive Matrix displays. The edge timing is fixed.<br>Value Description<br>0 LCDLP and LCDFP are driven on opposite edges of pixel clock than the LCD pixel output.<br>1 LCDLP and LCDFP are driven according to bit 24, PSYNCRF |
| 24 | PSYNCRF | RW | 0 | Program HSYNC/VSYNC Rise or Fall<br>Value Description<br>0 LCDLP and LCDFP are driven on the falling edge of pixel clock (PXLCLKCTL must be set to 1).<br>1 LCDLP and LCDFP are driven on the rising edge of pixel clock (PXLCLKCTL must be set to 1). |
| 23 | INVOE | RW | 0 | Invert Output Enable<br>Value Description<br>0 LCDAC pin is active high in active display mode<br>1 LCDAC pin is active low in active display mode<br>Active display mode: data driven out of the LCD's data lines on programmed pixel clock edge where AC-bias is active. Note that INVOE is ignored in passive display mode. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 22 | INVPXLCLK | RW | 0 | Invert Pixel Clock<br><br>Value Description<br>0 Data is driven on the LCD's data lines on the rising edge of LCDCP.<br>1 Data is driven on the LCD's data lines in the falling edge of LCDCP.<br><br>For Active Matrix output (LCDTFT=1), the Output Pixel Clock is a free running clock in that it transitions in horizontal blanking (including horizontal front porch, horizontal back porch) areas and all vertical blanking times.<br>For Passive Matrix output (LCDTFT=0), the Output Pixel Clock on occurs when an output data value is written. It is in a return-to-zero state when INVPXLCLK=0 and a return-to-one state when INVPXLCLK=1. |
| 21 | IHS | RW | 0 | Invert Hysync<br><br>Value Description<br>0 LCDLP pin is active high and inactive low<br>1 LCDLP pin is active low and inactive high<br><br>Active and passive mode: horizontal sync pulse/line clock active between lines, after the end of line wait period |
| 20 | IVS | RW | 0 | Invert Vsync<br><br>Value Description<br>0 LCDFP pin is active high and inactive low<br>1 LCDFP pin is active low and inactive high<br><br>Active mode: vertical sync pulse active between frames, after end of frame wait period<br>Passive mode: frame clock active during first line of each frame |
| 19:16 | ACBI | RW | 0x0 | AC Bias Pins Transitions per Interrupt<br>Value (from 0x0 to 0xF) used to specify the number of AC Bias pin transitions to count before setting the line count status (ACBS) bit, signaling an interrupt request. Counter frozen when ACBS is set, and is restarted when ACBS is cleared by software. This function is disabled when ACBI = 0x0000. |
| 15:8 | ACBF | RW | 0x00 | AC Bias Pin Frequency<br>Value (from 0x0 to 0xFF) used to specify the number of line clocks to count before transitioning the AC Bias pin.<br>This pin is used to periodically invert the polarity of the power supply to prevent DC charge build-up within the display. ACBF = Number of line clocks/toggle of the LCDAC pin. |
| 7:6 | reserved | RO | 0x0 | 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 | MSBHBP | RW | 0x0 | Bits 9:8 of the horizontal back porch field. |

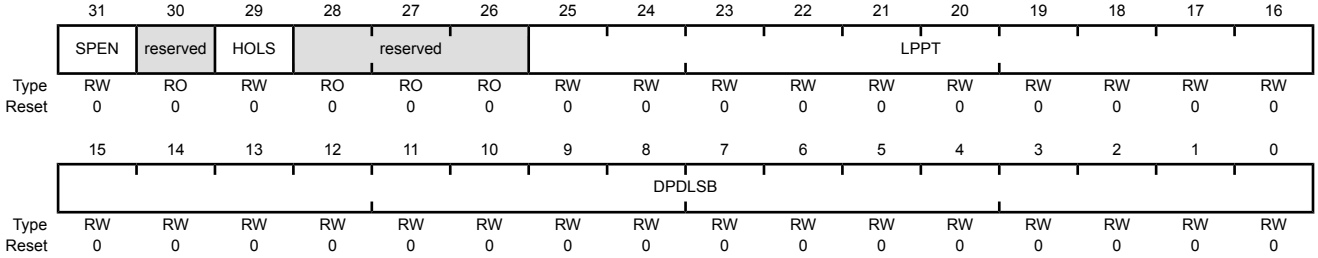
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1:0 | MSBHFP | RW | 0x0 | Bits 9:8 of the horizontal front porch field. |

## Register 14: LCD Raster Subpanel Display 1 (LCDRASTRSUBP1), offset 0x038

Note that subpictures are only allowed for Active Matrix mode (LCDTFT=1) in **LCDRASTRCTL**

LCD Raster Subpanel Display 1 (LCDRASTRSUBP1)

Base 0x4405.0000
Offset 0x038
Type RW, reset 0x0000.0000



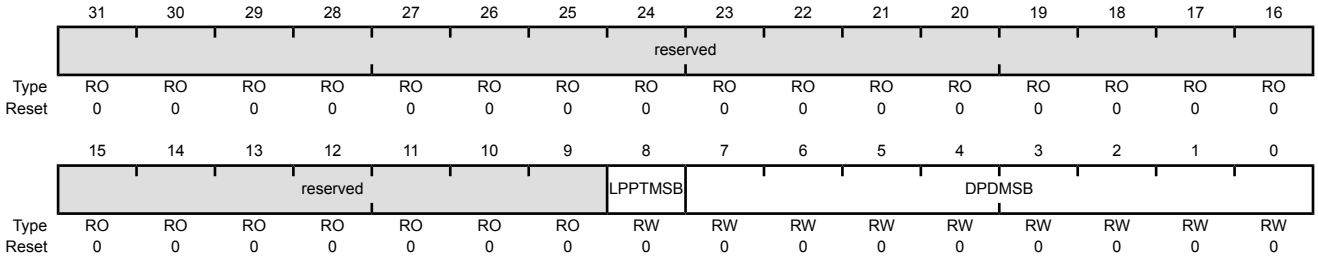

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SPEN | RW | 0 | Sub Panel Enable<br><br>Value Description<br>0 Function disabled<br>1 Sub-panel function mode enabled |
| 30 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 29 | HOLS | RW | 0 | High or Low Signal<br>This field indicates the position of the sub-panel based on the LPPT value.<br><br>Value Description<br>0 Default Pixel Data lines are at the top of the screen and the active video lines are at the bottom of the screen.<br>1 Active video lines are at the top of the screen and Default Pixel Data lines are at the bottom of the screen. |
| 28:26 | reserved | RO | 0x0 | 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. |
| 25:16 | LPPT | RW | 0x000 | Line Per Panel Threshold<br>Encoded value (programmed value range of {0:2047} represents an actual range of {1:2048}) used to specify the number of lines on the bottom part of the panel. Bit 10 of this field is in **LCDRASTRSUBP2**.<br>HOLS determines whether Default Pixel Data is on the top (HOLS=0) or on the bottom (HOLS=1). LPPT defines the number of lines on the bottom part of the output. |
| 15:0 | DPDLSB | RW | 0x0000 | Default Pixel Data LSB[15:0]<br>DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reach or after passing LPPT. DPDMSB is defined in bit field [7:0] in **LCDRASTRSUBP2**. |

## Register 15: LCD Raster Subpanel Display 2 (LCDRASTRSUBP2), offset 0x03C

Note that subpictures are only allowed for Active Matrix mode (LCDTFT=1) in **LCDRASTRCTL**

LCD Raster Subpanel Display 2 (LCDRASTRSUBP2)

Base 0x4405.0000
Offset 0x03C
Type RW, reset 0x0000.0000

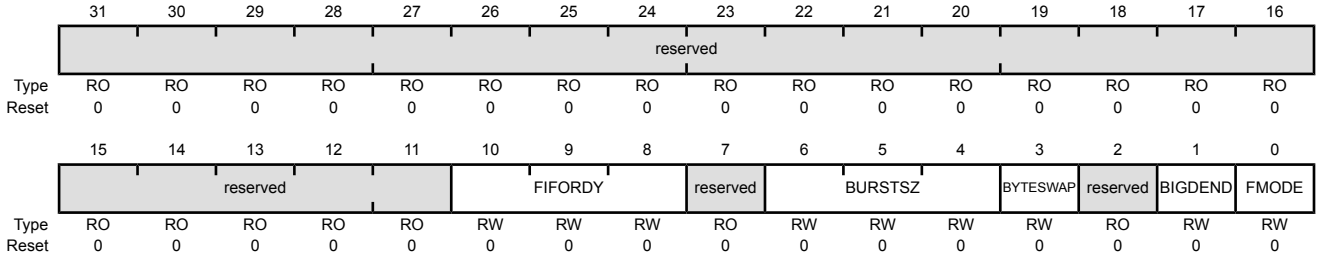



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | LPPTMSB | RW | 0 | Lines Per Panel Threshold Bit 10<br>This register is Bit 10 of the LPPT field in **LCDRASTRSUBP1**. |
| 7:0 | DPDMSB | RW | 0x00 | Default Pixel Data MSB [23:16]<br>DPD defines the default value of the pixel data sent to the panel for the lines until LPPT is reached or after passing the LPPT. |

## Register 16: LCD DMA Control (LCDDMACTL), offset 0x040

LCD DMA Control (LCDDMACTL)

Base 0x4405.0000
Offset 0x040
Type RW, reset 0x0000.0000



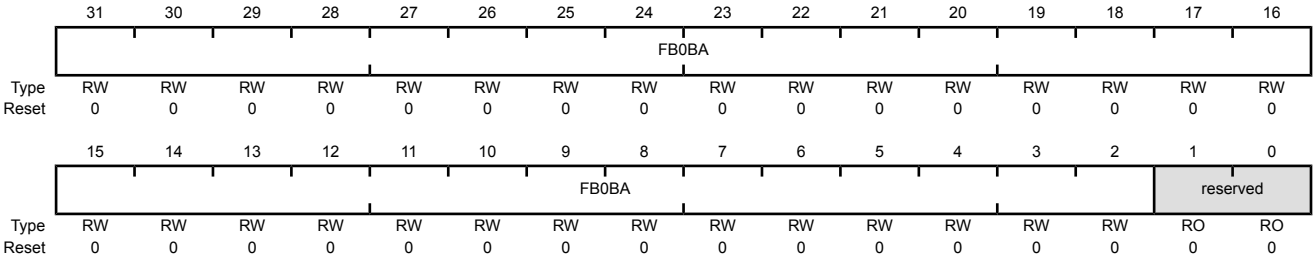

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:11 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10:8 | FIFORDY | RW | 0x0 | DMA FIFO threshold.<br>The DMA FIFO becomes ready when the number of words specified by this register from the frame buffer have been loaded.<br><br>Value Description<br>0x0 8 words<br>0x1 16 words<br>0x2 32 words<br>0x3 64 words<br>0x4 128 words<br>0x5 256 words<br>0x6 512 words<br>0x7 reserved |
| 7 | reserved | RO | 0 | 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. |
| 6:4 | BURSTSZ | RW | 0x0 | Burst Size setting for DMA transfers (all DMA transfers are 32 bits wide):<br><br>Value Description<br>0x0-0x1 reserved<br>0x2 burst size of 4<br>0x3 burst size of 8<br>0x4 burst size of 16<br>0x5-0x7 reserved<br><br>**Note:** For Raster Mode, configuring BURSTSZ should be done only after an LCD peripheral reset using the **SRLCD** register in System Control. The BURSTSZ field should not be changed once the LCD DMA is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | BYTESWAP | RW | 0 | This bit controls the bytelane ordering of the data on the output of the DMA module. It works in conjunction with the big-endian bit. See the big-endian description for configuration guidelines. |
| 2 | reserved | RO | 0 | 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. |
| 1 | BIGDEND | RW | 0 | Big Endian Enable<br>Use this bit when the processor is operating in Big Endian mode and writes to the frame buffers are less than 32-bits wide. Only in this scenario do we need to change the byte alignment for data coming into the FIFO from the frame buffer.<br>Value Description<br>0 Big Endian reordering disabled.<br>1 Big Endian reordering enabled.<br>The BIGEND and BYTESWAP bits control the byte lane ordering of the data on the output of the DMA module. |
| 0 | FMODE | RW | 0 | Frame Mode<br>Value Description<br>0 One frame buffer (FB0 only) used<br>1 Two frame buffers used,. DMA ping-pongs between FB0 and FB1 in this mode |

## Register 17: LCD DMA Frame Buffer 0 Base Address (LCDDMABAFB0), offset 0x044

LCD DMA Frame Buffer 0 Base Address (LCDDMABAFB0)

Base 0x4405.0000
Offset 0x044
Type RW, reset 0x0000.0000

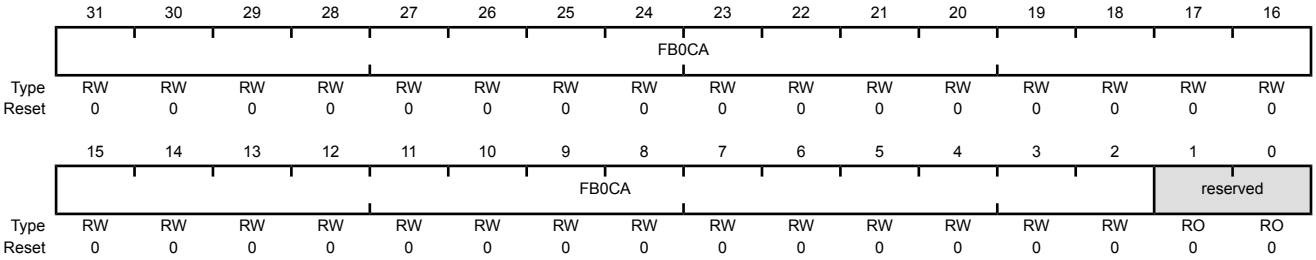



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | FB0BA | RW | 0x0000.000 | Frame Buffer 0 Base Address pointer |
| 1:0 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

## Register 18: LCD DMA Frame Buffer 0 Ceiling Address (LCDDMACAFB0), offset 0x048

LCD DMA Frame Buffer 0 Ceiling Address (LCDDMACAFB0)

Base 0x4405.0000
Offset 0x048
Type RW, reset 0x0000.0000

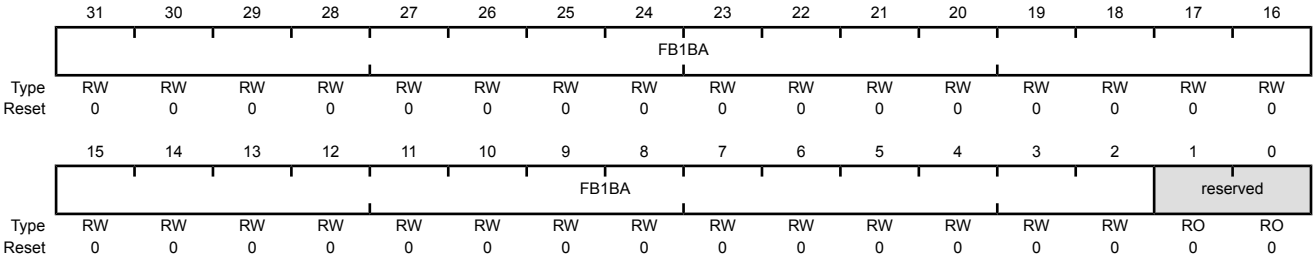



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | FB0CA | RW | 0x0000.000 | Frame Buffer 0 Ceiling Address pointer.<br>For raster modes (MODESEL = 1), this register cannot be the same value as FB0BA. |
| 1:0 | reserved | RO | 0x0 | Software should not rely on the value 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. |

## Register 19: LCD DMA Frame Buffer 1 Base Address (LCDDMABAFB1), offset 0x04C

LCD DMA Frame Buffer 1 Base Address (LCDDMABAFB1)

Base 0x4405.0000
Offset 0x04C
Type RW, reset 0x0000.0000

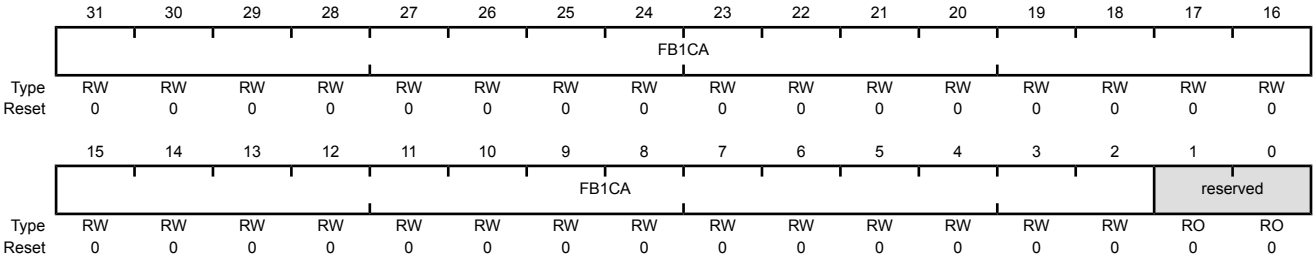



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | FB1BA | RW | 0x0000.000 | Frame Buffer 1 Base Address pointer |
| 1:0 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

## Register 20: LCD DMA Frame Buffer 1 Ceiling Address (LCDDMACAFB1), offset 0x050

LCD DMA Frame Buffer 1 Ceiling Address (LCDDMACAFB1)

Base 0x4405.0000
Offset 0x050
Type RW, reset 0x0000.0000

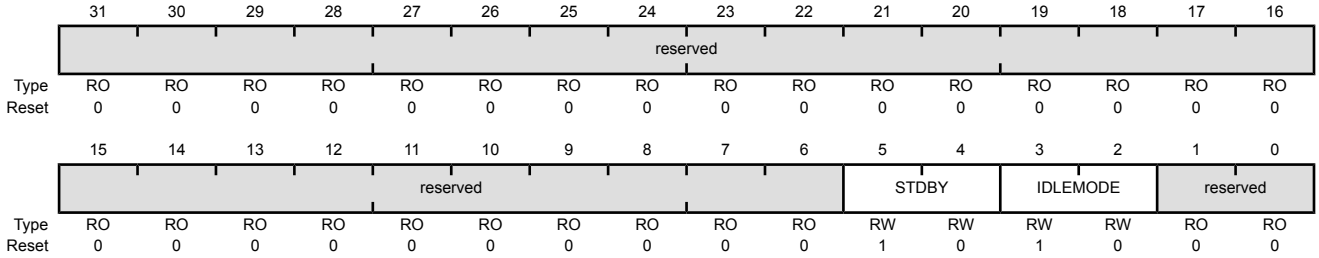



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | FB1CA | RW | 0x0000.000 | Frame Buffer 1 Ceiling Address pointer.<br>For raster modes (MODESEL = 1), this register cannot be the same value as FB1BA. |
| 1:0 | reserved | RO | 0x0 | Software should not rely on the value 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. |

## Register 21: LCD System Configuration Register (LCDSYSCFG), offset 0x054

LCD System Configuration Register (LCDSYSCFG)

Base 0x4405.0000
Offset 0x054
Type RW, reset 0x0000.0028




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.00 | 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 | STDBY | RW | 0x2 | Standby Mode<br>Configuration of the local initiator state management mode.<br>By definition, initiator may generate read/write transaction as long as it is out of STANDBY state<br><br>Value Description<br>0x0 Force-standby mode: local initiator is unconditionally placed in standby state. Backup mode, for debug only<br>0x1 No-standby mode: local initiator is unconditionally placed out of standby state. Backup mode, for debug only<br>0x2 Smart-standby mode: local initiator standby status depends on local conditions, that is the module's functional requirement from the initiator. IP module shall not generate (initiator-related) wakeup events<br>0x3 Reserved<br><br>IP module may generate (master-related) wakeup events when in standby state. |

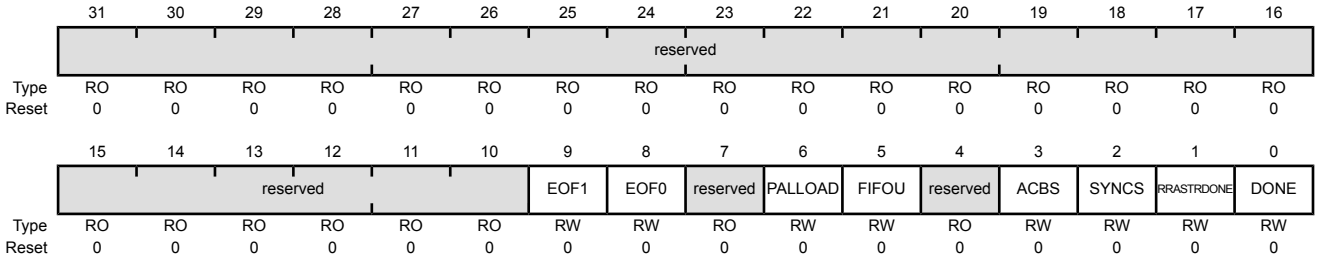
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | IDLEMODE | RW | 0x2 | Idle Mode<br>Configuration of the local target state management mode.<br>By definition, target can handle read/write transaction as long as it is out of IDLE state<br><br>Value Description<br>0x0 Force-idle mode: local target's idle state follows (acknowledges) the system's idle requests unconditionally, that regardless of the IP module's internal requirements. Backup mode, for debug only<br>0x1 No-idle mode: local target never enters idle state. Backup mode, for debug only<br>0x2 Smart-idle mode: local target's idle state eventually follows (acknowledges) the system's idle requests, depending on the IP module's internal requirements. IP module shall not generate (IRQ- or DMA-requestrelated) wakeup events<br>0x3 Reserved |
| 1:0 | reserved | RO | 0x0 | 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. |

## Register 22: LCD Interrupt Raw Status and Set Register (LCDRISSET), offset 0x058

This register contains the raw interrupt status. In addition to providing the Raw Interrupt Status on a read, a 1 to a bit will set the associated interrupt.

LCD Interrupt Raw Status and Set Register (LCDRISSET)

Base 0x4405.0000
Offset 0x058
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9 | EOF1 | RW | 0 | DMA End-of-Frame 1 Raw Interrupt Status and Set<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status<br><br>Value Description<br>0 Inactive<br>1 Active |
| 8 | EOF0 | RW | 0 | DMA End-of-Frame 0 Raw Interrupt Status and Set<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status<br><br>Value Description<br>0 Inactive<br>1 Active |
| 7 | reserved | RO | 0 | 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. |
| 6 | PALLOAD | RW | 0 | DMA Palette Loaded Raw Interrupt Status and Set<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status<br><br>Value Description<br>0 Inactive<br>1 Active |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | FIFOU | RW | 0 | DMA FIFO Underflow Raw Interrupt Status and Set<br>Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO.<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status<br>Value Description<br>0 Inactive<br>1 Active |
| 4 | reserved | RO | 0 | 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. |
| 3 | ACBS | RW | 0 | AC Bias Count Raw Interrupt Status and Set<br>For Passive Matrix Panels Only<br>AC bias transition counter has decremented to zero, indicating that the lcd_ac_o line has transitioned the number of times which is specified by the ACBI control bit-field in the **LCDRASTRTIMn** register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS.<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status<br>Value Description<br>0 Inactive<br>1 Active |
| 2 | SYNCS | RW | 0 | Frame Synchronization Lost Raw Interrupt Status and Set<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status.<br>Value Description<br>0 Inactive<br>1 Active |
| 1 | RRASTRDONE | RW | 0 | Raster Mode Frame Done interrupt.<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status.<br>Value Description<br>0 Inactive<br>1 Active |

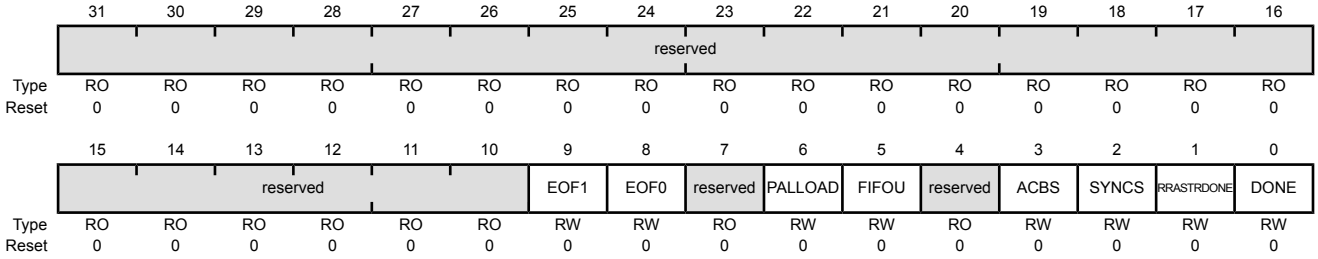
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | DONE | RW | 0 | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Raw Interrupt Status and Set<br>Writing 1 will set status. Writing 0 has no effect. Read indicates raw status.<br><br>Value Description<br>0 Inactive<br>1 Active |

## Register 23: LCD Interrupt Status and Clear (LCDMISCLR), offset 0x05C

This register contains the Interrupt Enable Status. This register returns Masked (Enabled) status interrupts on a Read. Writing a 1 to a bit will clear the associated interrupt.

LCD Interrupt Status and Clear (LCDMISCLR)

Base 0x4405.0000
Offset 0x05C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9 | EOF1 | RW | 0 | DMA End-of-Frame 1 Enabled Interrupt and Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect Read indicates enabled (masked) status.<br><br>Value Description<br>0 Inactive<br>1 Active |
| 8 | EOF0 | RW | 0 | DMA End-of-Frame 0 Raw Interrupt and Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br><br>Value Description<br>0 Inactive<br>1 Active |
| 7 | reserved | RO | 0 | 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. |
| 6 | PALLOAD | RW | 0 | DMA Palette Loaded Enabled Interrupt and Clear<br>Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status.<br><br>Value Description<br>0 Inactive<br>1 Active |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | FIFOU | RW | 0 | DMA FIFO Underflow Enabled Interrupt and Clear<br>Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO.<br>Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Inactive<br>1 Active |
| 4 | reserved | RO | 0 | 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. |
| 3 | ACBS | RW | 0 | AC Bias Count Enabled Interrupt and Clear<br>For Passive Matrix Panels Only<br>AC bias transition counter has decremented to zero, indicating that the LCDAC line has transitioned the number of times which is specified by the ACBI control bit-field in the **LCDRASTRTIMn** register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS.<br>Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Inactive<br>1 Active |
| 2 | SYNCS | RW | 0 | Frame Synchronization Lost Enabled Interrupt and Clear<br>Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Inactive<br>1 Active |
| 1 | RRASTRDONE | RW | 0 | Raster Mode Frame Done interrupt.<br>Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Inactive<br>1 Active |

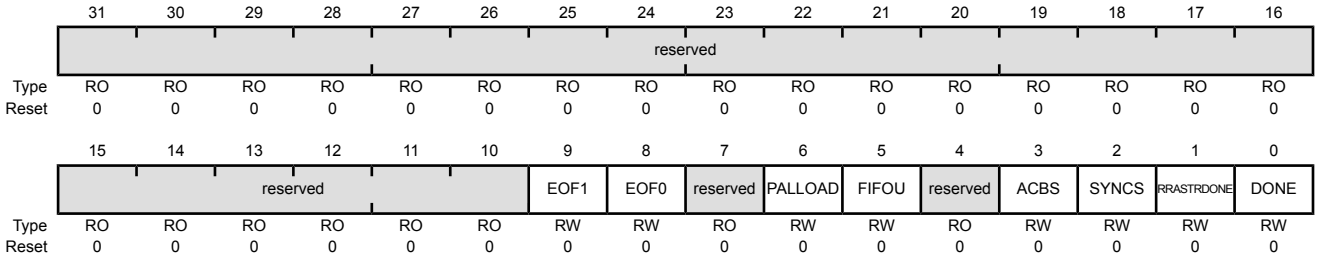
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | DONE | RW | 0 | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Enabled Interrupt and Clear<br>Writing 1 will set status. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Inactive<br>1 Active |

## Register 24: LCD Interrupt Mask (LCDIM), offset 0x060

This register provides for the setting of interrupt enables (Mask bits).

LCD Interrupt Mask (LCDIM)

Base 0x4405.0000
Offset 0x060
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9 | EOF1 | RW | 0 | DMA End-of-Frame 1 Interrupt Enable Set<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (mask) status.<br><br>Value Description<br>0 Disabled<br>1 Enabled |
| 8 | EOF0 | RW | 0 | DMA End-of-Frame 0 Interrupt Enable Set<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br><br>Value Description<br>0 Disabled<br>1 Enabled |
| 7 | reserved | RO | 0 | 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. |
| 6 | PALLOAD | RW | 0 | DMA Palette Loaded Interrupt Enable Set<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br><br>Value Description<br>0 Disabled<br>1 Enabled |

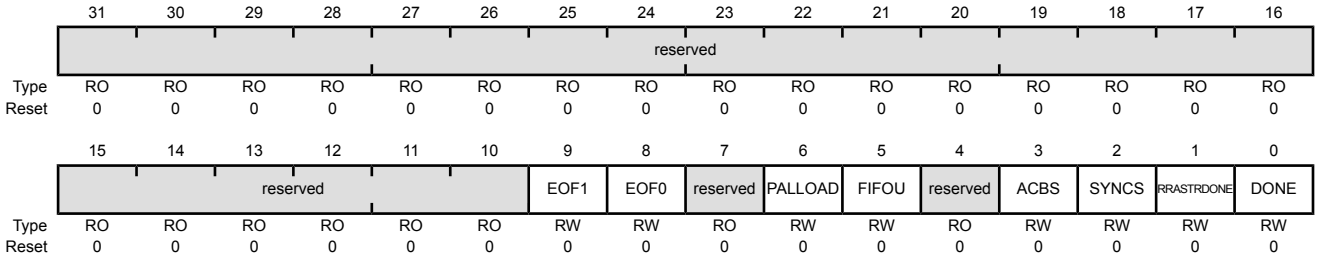
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | FIFOU | RW | 0 | DMA FIFO Underflow Interrupt Enable Set<br>Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO.<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Disabled<br>1 Enabled |
| 4 | reserved | RO | 0 | 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. |
| 3 | ACBS | RW | 0 | AC Bias Count Interrupt Enable Set<br>For Passive Matrix Panels Only<br>AC bias transition counter has decremented to zero, indicating that the LCDAC line has transitioned the number of times which is specified by the ACBI control bit-field in the **LCDRASTRTIMn** register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS.<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Disabled<br>1 Enabled |
| 2 | SYNCS | RW | 0 | Frame Synchronization Lost Interrupt Enable Set<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Disabled<br>1 Enabled |
| 1 | RRASTRDONE | RW | 0 | Raster Mode Frame Done Interrupt Enable Set<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br>Value Description<br>0 Disabled<br>1 Enabled |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | DONE | RW | 0 | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Interrupt Enable Set<br>Writing 1 will set interrupt enable. Writing 0 has no effect. Read indicates enabled (masked) status.<br><br>Value Description<br>0 Disabled<br>1 Enabled |

## Register 25: LCD Interrupt Enable Clear (LCDIENC), offset 0x064

This register provides for the clearing of interrupt enables (Mask bits).

LCD Interrupt Enable Clear (LCDIENC)
Base 0x4405.0000
Offset 0x064
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.00 | 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. |
| 9 | EOF1 | RW | 0 | DMA End-of-Frame 1 Interrupt Enable Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br><br>Value Description<br>0 Disabled<br>1 Enabled |
| 8 | EOF0 | RW | 0 | DMA End-of-Frame 0 Interrupt Enable Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br><br>Value Description<br>0 Disabled<br>1 Enabled |
| 7 | reserved | RO | 0 | 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. |
| 6 | PALLOAD | RW | 0 | DMA Palette Loaded Interrupt Enable Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br><br>Value Description<br>0 Disabled<br>1 Enabled |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | FIFOU | RW | 0 | DMA FIFO Underflow Interrupt Enable Clear<br>Indicates if LCD dithering logic is not supplying data to the FIFO at a sufficient rate. FIFO has completely emptied and data pin driver logic has attempted to take added data from FIFO.<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br>Value Description<br>0 Disabled<br>1 Enabled |
| 4 | reserved | RO | 0 | 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. |
| 3 | ACBS | RW | 0 | AC Bias Count Interrupt Enable Clear<br>For Passive Matrix Panels Only<br>AC bias transition counter has decremented to zero, indicating that the LCDAC line has transitioned the number of times which is specified by the ACBI control bit-field in the **LCDRASTRTIMn** register. The counter is reloaded with the value in ACBI but it is disabled until the user clears ABCS.<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br>Value Description<br>0 Disabled<br>1 Enabled |
| 2 | SYNCS | RW | 0 | Frame Synchronization Lost Interrupt Enable Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br>Value Description<br>0 Disabled<br>1 Enabled |
| 1 | RRASTRDONE | RW | 0 | Raster Mode Frame Done Interrupt Enable Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br>Value Description<br>0 Disabled<br>1 Enabled |

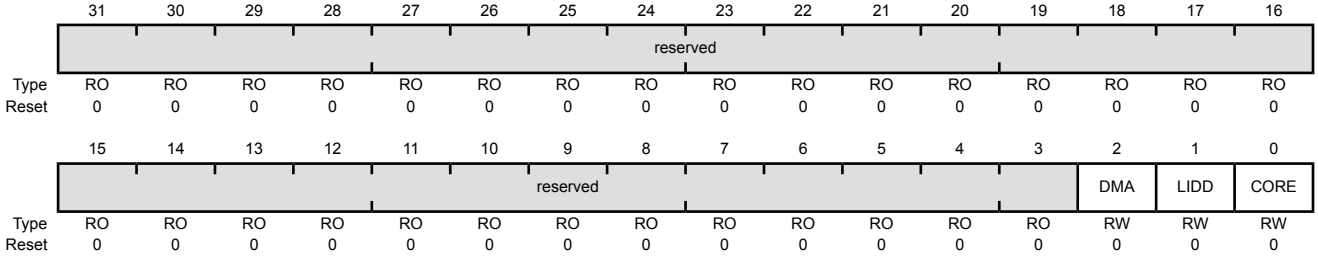
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | DONE | RW | 0 | Raster or LIDD Frame Done (shared, depends on whether Raster or LIDD mode enabled) Interrupt Enable Clear<br>Writing 1 will clear interrupt enable. Writing 0 has no effect. Read indicates enabled status.<br>Value Description<br>0 Disabled<br>1 Enabled |

## Register 26: LCD Clock Enable (LCDCLKEN), offset 0x06C

This register contains the Clock enables for each major domain within the LCD.

LCD Clock Enable (LCDCLKEN)

Base 0x4405.0000
Offset 0x06C
Type RW, reset 0x0000.0000



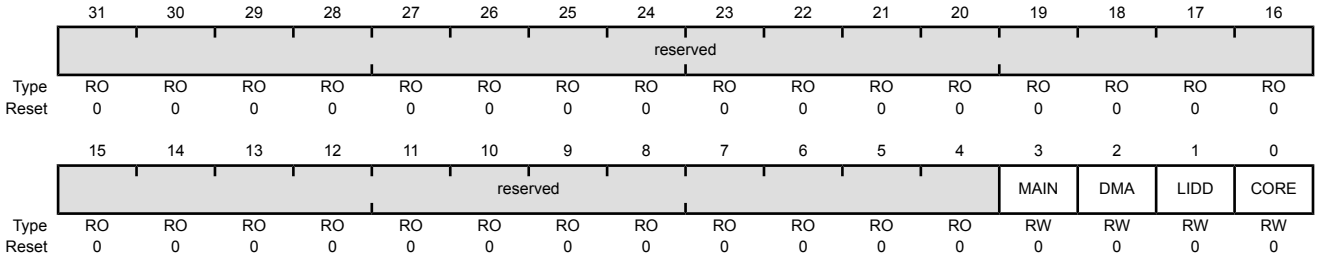

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DMA | RW | 0 | DMA Clock Enable<br>Software Clock Enable for the LCD DMA<br><br>Value Description<br>0 Clock Disabled<br>1 Clock Enabled |
| 1 | LIDD | RW | 0 | LIDD Submodule Clock Enable<br>Software Clock Enable for the LIDD submodule (character displays).<br>The LIDD submodule runs on the System Clock domain<br><br>Value Description<br>0 Clock Disabled<br>1 Clock Enabled |
| 0 | CORE | RW | 0 | LCD Core Clock Enable<br>Software Clock Enable for the LCD core, which encompasses the Raster Active Matrix and Passive Matrix logic.<br>The Core runs on the System Clock domain.<br><br>Value Description<br>0 Clock Disabled<br>1 Clock Enabled |

## Register 27: LCD Clock Resets (LCDCLKRESET), offset 0x070

This register contains the Software Resets for each major domain within the LCD.

LCD Clock Resets (LCDCLKRESET)

Base 0x4405.0000
Offset 0x070
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | MAIN | RW | 0 | Software Reset for the entire LCD module. This reset affects the L3 clk and lcd_clk domain.<br><br>Value Description<br>0 Reset Disabled<br>1 Reset Enabled |
| 2 | DMA | RW | 0 | Software Reset for the DMA submodule. This reset affects the L3 clk domain.<br><br>Value Description<br>0 Reset Disabled<br>1 Reset Enabled |
| 1 | LIDD | RW | 0 | Software Reset for the LIDD submodule (character displays). This reset affects the LIDD logic in the lcd_clk domain.<br><br>Value Description<br>0 Reset Disabled<br>1 Reset Enabled |
| 0 | CORE | RW | 0 | Software Reset for the Core, which encompasses the Raster Active Matrix and Passive Matrix logic. This reset affects the Core (active matrix and passive matrix raster mode) logic in the lcd_clk domain.<br><br>Value Description<br>0 Reset Disabled<br>1 Reset Enabled |

# 27 Analog Comparators

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result.

**Note:** Not all comparators have the option to drive an output pin. See “Signal Description” on page 1910 for more information.

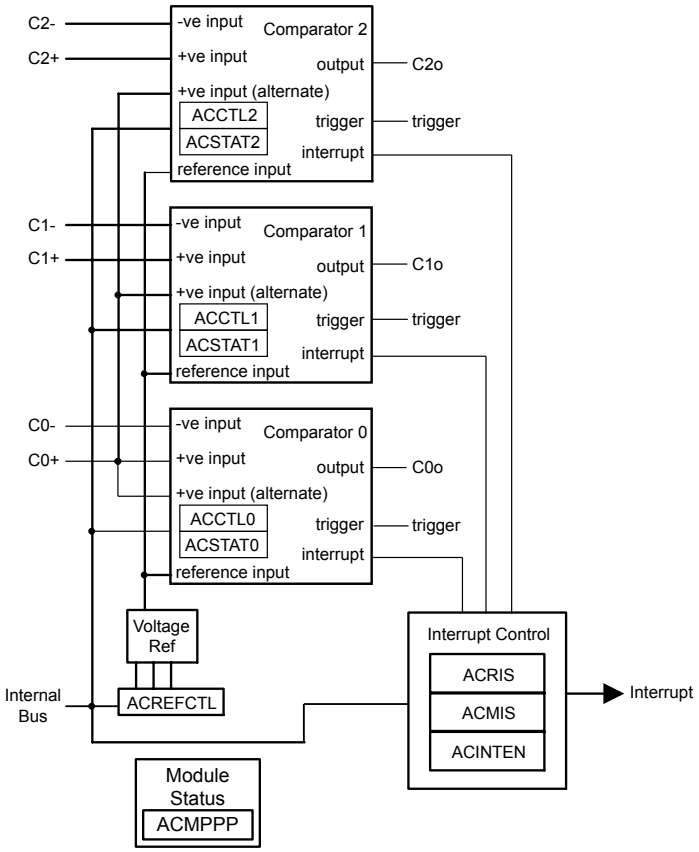
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board. In addition, the comparator can signal the application via interrupts or trigger the start of a sample sequence in the ADC. The interrupt generation and ADC triggering logic is separate and independent. This flexibility means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The TM4C129XKCZAD microcontroller provides three independent integrated analog comparators with the following functions:

- Compare external pin input to external pin input or to internal programmable voltage reference
- Compare a test voltage against any one of the following voltages:
  - An individual external reference voltage
  - A shared single external reference voltage
  - A shared internal reference voltage

## 27.1 Block Diagram

**Figure 27-1. Analog Comparator Module Block Diagram**




**Note:** This block diagram depicts the maximum number of analog comparators and comparator outputs for the family of microcontrollers; the number for this specific device may vary. See page 1923 for what is included on this device.

## 27.2 Signal Description

The following table lists the external signals of the Analog Comparators and describes the function of each. The Analog Comparator output signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the Analog Comparator function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the Analog Comparator signal to the specified GPIO port pin. The positive and negative input signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 27-1. Analog Comparators Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| C0+ | L2 | PC6 | I | Analog | Analog comparator 0 positive input. |

**Table 27-1. Analog Comparators Signals (212BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| C0- | K3 | PC7 | I | Analog | Analog comparator 0 negative input. |
| C0o | C2<br>G18 | PD0 (5)<br>PL2 (5) | O | TTL | Analog comparator 0 output. |
| C1+ | M1 | PC5 | I | Analog | Analog comparator 1 positive input. |
| C1- | M2 | PC4 | I | Analog | Analog comparator 1 negative input. |
| C1o | C1<br>J18 | PD1 (5)<br>PL3 (5) | O | TTL | Analog comparator 1 output. |
| C2+ | D6 | PP0 | I | Analog | Analog comparator 2 positive input. |
| C2- | D7 | PP1 | I | Analog | Analog comparator 2 negative input. |
| C2o | D2 | PD2 (5) | O | TTL | Analog comparator 2 output. |

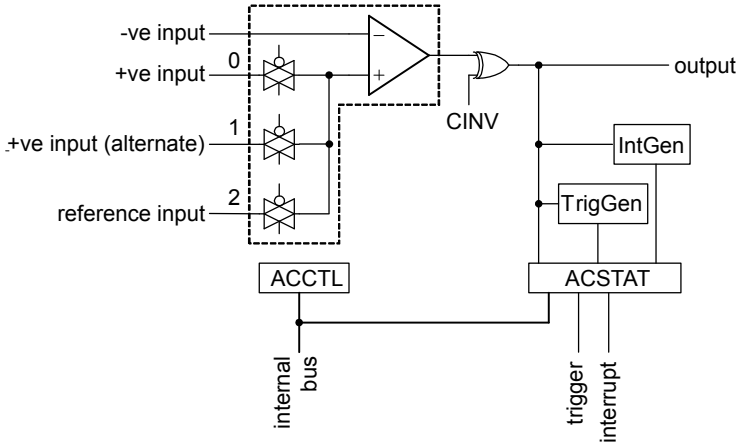
## 27.3 Functional Description

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

```
VIN- < VIN+, VOUT = 1
VIN- > VIN+, VOUT = 0
```

As shown in Figure 27-2 on page 1911, the input source for VIN- is an external input, Cn-, where n is the analog comparator number. In addition to an external input, Cn+, input sources for VIN+ can be the C0+ or an internal reference, $V_{IREF}$.

**Figure 27-2. Structure of Comparator Unit**




A comparator is configured through two status/control registers, **Analog Comparator Control (ACCTL)** and **Analog Comparator Status (ACSTAT)**. The internal reference is configured through one control register, **Analog Comparator Reference Voltage Control (ACREFCTL)**. Interrupt status and control are configured through three registers, **Analog Comparator Masked Interrupt Status (ACMIS)**, **Analog Comparator Raw Interrupt Status (ACRIS)**, and **Analog Comparator Interrupt Enable (ACINTEN)**.

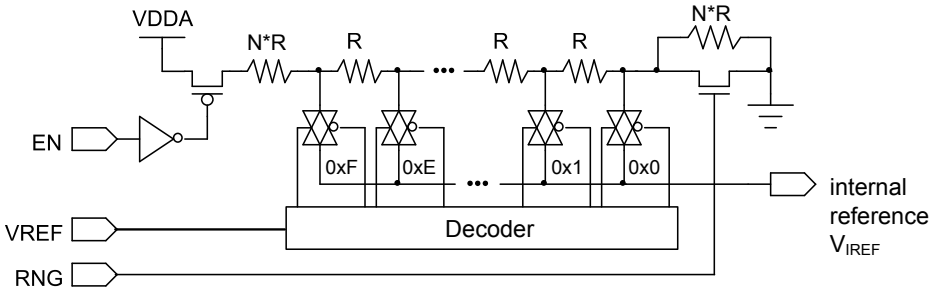
Typically, the comparator output is used internally to generate an interrupt as controlled by the ISEN bit in the **ACCTL** register. The output may also be used to drive one of the external pins (Cno), or generate an analog-to-digital converter (ADC) trigger.

**Important:** The ASRCP bits in the **ACCTL** register must be set before using the analog comparators.

### 27.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 27-3 on page 1912. The internal reference is controlled by a single configuration register (**ACREFCTL**).

**Figure 27-3. Comparator Internal Reference Structure**




**Note:** In the figure above, N*R represents a multiple of the R value that produces the results specified in Table 27-2 on page 1912.

The internal reference can be programmed in one of two modes (low range or high range) depending on the RNG bit in the **ACREFCTL** register. When RNG is clear, the internal reference is in high-range mode, and when RNG is set the internal reference is in low-range mode.

In each range, the internal reference, $V_{IREF}$, has 16 preprogrammed thresholds or step values. The threshold to be used to compare the external input voltage against is selected using the VREF field in the **ACREFCTL** register.

In the high-range mode, the $V_{IREF}$ threshold voltages start at the ideal high-range starting voltage of $V_{DDA}/4.2$ and increase in ideal constant voltage steps of $V_{DDA}/29.4$.

In the low-range mode, the $V_{IREF}$ threshold voltages start at 0 V and increase in ideal constant voltage steps of $V_{DDA}/22.12$. The ideal $V_{IREF}$ step voltages for each mode and their dependence on the RNG and VREF fields are summarized in Table 27-2.

**Table 27-2. Internal Reference Voltage and ACREFCTL Field Values**

| ACREFCTL Register | | Output Reference Voltage Based on VREF Field Value |
|---|---|---|
| EN Bit Value | RNG Bit Value | |
| EN=0 | RNG=X | 0 V (GND) for any value of VREF. It is recommended that RNG=1 and VREF=0 to minimize noise on the reference ground. |

**Table 27-2. Internal Reference Voltage and ACREFCTL Field Values *(continued)***

| ACREFCTL Register | | Output Reference Voltage Based on VREF Field Value |
|---|---|---|
| EN Bit Value | RNG Bit Value | |
| EN=1 | RNG=0 | $V_{IREF}$ High Range: 16 voltage threshold values indexed by VREF = 0x0 .. 0xF<br>Ideal starting voltage (VREF=0): $V_{DDA}$ / 4.2<br>Ideal step size: $V_{DDA}$/ 29.4<br>Ideal $V_{IREF}$ threshold values: $V_{IREF}$ (VREF) = $V_{DDA}$ / 4.2 + VREF * ($V_{DDA}$/ 29.4), for VREF = 0x0 .. 0xF<br>For minimum and maximum $V_{IREF}$ threshold values, see Table 27-3 on page 1913. |
| | RNG=1 | $V_{IREF}$ Low Range: 16 voltage threshold values indexed by VREF = 0x0 .. 0xF<br>Ideal starting voltage (VREF=0): 0 V<br>Ideal step size: $V_{DDA}$/ 22.12<br>Ideal $V_{IREF}$ threshold values: $V_{IREF}$ (VREF) = VREF * ($V_{DDA}$/ 22.12), for VREF = 0x0 .. 0xF<br>For minimum and maximum $V_{IREF}$ threshold values, see Table 27-4 on page 1914. |

Note that the values shown in Table 27-2 are the ideal values of the $V_{IREF}$ thresholds. These values actually vary between minimum and maximum values for each threshold step, depending on process and temperature. The minimum and maximum values for each step are given by:

- $V_{IREF}$(VREF) [Min] = Ideal $V_{IREF}$(VREF) – (Ideal Step size – 2 mV) / 2
- $V_{IREF}$(VREF) [Max] = Ideal $V_{IREF}$(VREF) + (Ideal Step size – 2 mV) / 2

Examples of minimum and maximum $V_{IREF}$ values for $V_{DDA}$ = 3.3V for high and low ranges, are shown inTable 27-3 and Table 27-4. Note that these examples are only valid for $V_{DDA}$ = 3.3V; values scale up and down with $V_{DDA}$.

**Table 27-3. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 0**

| VREF Value | $V_{IREF}$ Min | Ideal $V_{IREF}$ | $V_{IREF}$ Max | Unit |
|---|---|---|---|---|
| 0x0 | 0.731 | 0.786 | 0.841 | V |
| 0x1 | 0.843 | 0.898 | 0.953 | V |
| 0x2 | 0.955 | 1.010 | 1.065 | V |
| 0x3 | 1.067 | 1.122 | 1.178 | V |
| 0x4 | 1.180 | 1.235 | 1.290 | V |
| 0x5 | 1.292 | 1.347 | 1.402 | V |
| 0x6 | 1.404 | 1.459 | 1.514 | V |
| 0x7 | 1.516 | 1.571 | 1.627 | V |
| 0x8 | 1.629 | 1.684 | 1.739 | V |
| 0x9 | 1.741 | 1.796 | 1.851 | V |
| 0xA | 1.853 | 1.908 | 1.963 | V |
| 0xB | 1.965 | 2.020 | 2.076 | V |
| 0xC | 2.078 | 2.133 | 2.188 | V |
| 0xD | 2.190 | 2.245 | 2.300 | V |
| 0xE | 2.302 | 2.357 | 2.412 | V |
| 0xF | 2.414 | 2.469 | 2.525 | V |

**Table 27-4. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 1**

| VREF Value | $V_{IREF}$ Min | Ideal $V_{IREF}$ | $V_{IREF}$ Max | Unit |
|---|---|---|---|---|
| 0x0 | 0.000 | 0.000 | 0.074 | V |
| 0x1 | 0.076 | 0.149 | 0.223 | V |
| 0x2 | 0.225 | 0.298 | 0.372 | V |
| 0x3 | 0.374 | 0.448 | 0.521 | V |
| 0x4 | 0.523 | 0.597 | 0.670 | V |
| 0x5 | 0.672 | 0.746 | 0.820 | V |
| 0x6 | 0.822 | 0.895 | 0.969 | V |
| 0x7 | 0.971 | 1.044 | 1.118 | V |
| 0x8 | 1.120 | 1.193 | 1.267 | V |
| 0x9 | 1.269 | 1.343 | 1.416 | V |
| 0xA | 1.418 | 1.492 | 1.565 | V |
| 0xB | 1.567 | 1.641 | 1.715 | V |
| 0xC | 1.717 | 1.790 | 1.864 | V |
| 0xD | 1.866 | 1.939 | 2.013 | V |
| 0xE | 2.015 | 2.089 | 2.162 | V |
| 0xF | 2.164 | 2.238 | 2.311 | V |

## 27.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator clock by writing a value of 0x0000.0001 to the **RCGCACMP** register in the System Control module (see page 414).
2. Enable the clock to the appropriate GPIO modules via the **RCGCGPIO** register (see page 399). To find out which GPIO ports to enable, refer to Table 31-5 on page 2081.
3. In the GPIO module, enable the GPIO port/pin associated with the input signals as GPIO inputs. To determine which GPIO to configure, see Table 31-4 on page 2064.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the analog comparator output signals to the appropriate pins (see page 820 and Table 31-5 on page 2081).
5. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
6. Configure the comparator to use the internal voltage reference and to *not* invert the output by writing the **ACCTLn** register with the value of 0x0000.040C.
7. Delay for 10 µs.
8. Read the comparator output value by reading the **ACSTATn** register's OVAL value.

Change the level of the comparator negative input signal C- to see the OVAL value change.

## 27.5 Register Map

Table 27-5 on page 1915 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000. Note that the analog comparator clock must be enabled before the registers can be programmed (see page 414). There must be a delay of 3 system clocks after the analog comparator module clock is enabled before any analog comparator module registers are accessed.

**Table 27-5. Analog Comparators Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | ACMIS | RW1C | 0x0000.0000 | Analog Comparator Masked Interrupt Status | 1916 |
| 0x004 | ACRIS | RO | 0x0000.0000 | Analog Comparator Raw Interrupt Status | 1917 |
| 0x008 | ACINTEN | RW | 0x0000.0000 | Analog Comparator Interrupt Enable | 1918 |
| 0x010 | ACREFCTL | RW | 0x0000.0000 | Analog Comparator Reference Voltage Control | 1919 |
| 0x020 | ACSTAT0 | RO | 0x0000.0000 | Analog Comparator Status 0 | 1920 |
| 0x024 | ACCTL0 | RW | 0x0000.0000 | Analog Comparator Control 0 | 1921 |
| 0x040 | ACSTAT1 | RO | 0x0000.0000 | Analog Comparator Status 1 | 1920 |
| 0x044 | ACCTL1 | RW | 0x0000.0000 | Analog Comparator Control 1 | 1921 |
| 0x060 | ACSTAT2 | RO | 0x0000.0000 | Analog Comparator Status 2 | 1920 |
| 0x064 | ACCTL2 | RW | 0x0000.0000 | Analog Comparator Control 2 | 1921 |
| 0xFC0 | ACMPPP | RO | 0x0007.0007 | Analog Comparator Peripheral Properties | 1923 |

## 27.6 Register Descriptions

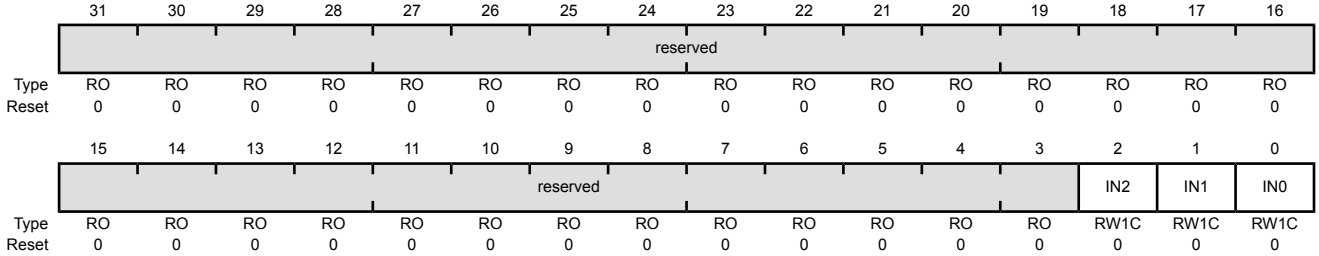
The remainder of this section lists and describes the Analog Comparator registers, in numerical order by address offset.

## Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparators.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x000
Type RW1C, reset 0x0000.0000



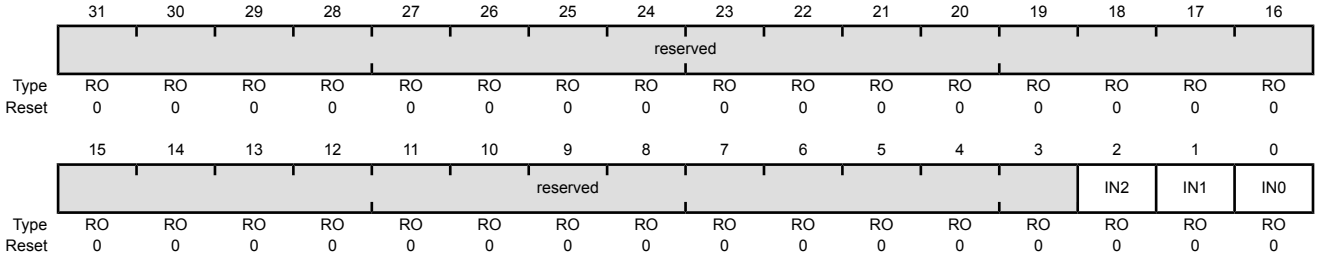

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | IN2 | RW1C | 0 | Comparator 2 Masked Interrupt Status<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The IN2 bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the IN2 bit in the **ACRIS** register. |
| 1 | IN1 | RW1C | 0 | Comparator 1 Masked Interrupt Status<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The IN1 bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the IN1 bit in the **ACRIS** register. |
| 0 | IN0 | RW1C | 0 | Comparator 0 Masked Interrupt Status<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The IN0 bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the IN0 bit in the **ACRIS** register. |

## Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparators. The bits in this register must be enabled to generate interrupts using the **ACINTEN** register.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x004
Type RO, reset 0x0000.0000



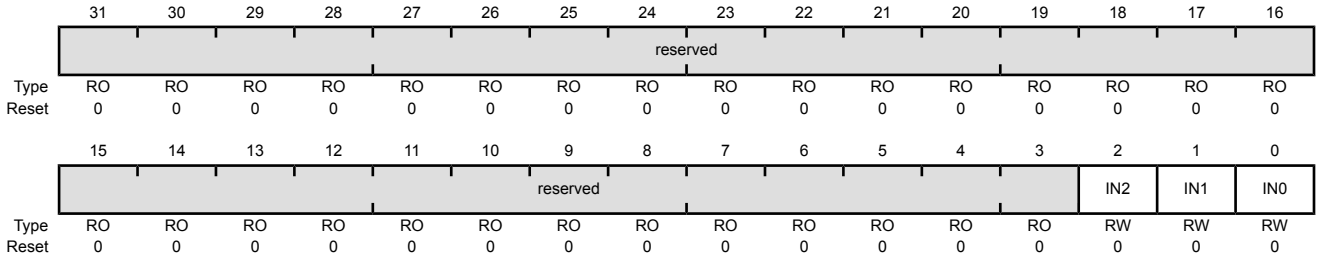

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | IN2 | RO | 0 | Comparator 2 Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 Comparator 2 has generated an interrupt for an event as configured by the ISEN bit in the **ACCTL2** register.<br><br>This bit is cleared by writing a 1 to the IN2 bit in the **ACMIS** register. |
| 1 | IN1 | RO | 0 | Comparator 1 Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 Comparator 1 has generated an interruptfor an event as configured by the ISEN bit in the **ACCTL1** register.<br><br>This bit is cleared by writing a 1 to the IN1 bit in the **ACMIS** register. |
| 0 | IN0 | RO | 0 | Comparator 0 Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 Comparator 0 has generated an interrupt for an event as configured by the ISEN bit in the **ACCTL0** register.<br><br>This bit is cleared by writing a 1 to the IN0 bit in the **ACMIS** register. |

## Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparators.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
Offset 0x008
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x00 | 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. |
| 2 | IN2 | RW | 0 | Comparator 2 Interrupt Enable<br><br>Value Description<br>0 A comparator 2 interrupt does not affect the interrupt status.<br>1 The raw interrupt signal comparator 2 is sent to the interrupt controller. |
| 1 | IN1 | RW | 0 | Comparator 1 Interrupt Enable<br><br>Value Description<br>0 A comparator 1 interrupt does not affect the interrupt status.<br>1 The raw interrupt signal comparator 1 is sent to the interrupt controller. |
| 0 | IN0 | RW | 0 | Comparator 0 Interrupt Enable<br><br>Value Description<br>0 A comparator 0 interrupt does not affect the interrupt status.<br>1 The raw interrupt signal comparator 0 is sent to the interrupt controller. |

## Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x010
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | EN | RNG | reserved | | | | VREF | | | |
| Type | RO | RO | RO | RO | RO | RO | RW | RW | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | reserved | RO | 0x0000.0 | 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. |
| 9 | EN | RW | 0 | Resistor Ladder Enable<br><br>Value Description<br>0 The resistor ladder is unpowered.<br>1 Powers on the resistor ladder. The resistor ladder is connected to $V_{DDA}$.<br><br>This bit is cleared at reset so that the internal reference consumes the least amount of power if it is not used. |
| 8 | RNG | RW | 0 | Resistor Ladder Range<br><br>Value Description<br>0 The ideal step size for the internal reference is VDDA / 29.4.<br>1 The ideal step size for the internal reference is VDDA / 22.12. |
| 7:4 | reserved | RO | 0x0 | 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. |
| 3:0 | VREF | RW | 0x0 | Resistor Ladder Voltage Ref<br><br>The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 27-2 on page 1912 for some output reference voltage examples. |

## Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020

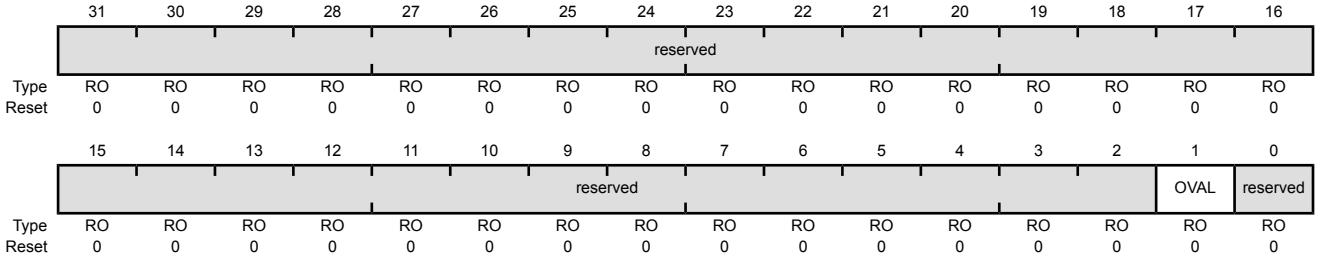
## Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040

## Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x060

These registers specify the current output value of the comparator.

Analog Comparator Status n (ACSTATn)

Base 0x4003.C000
Offset 0x020
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | 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. |
| 1 | OVAL | RO | 0 | Comparator Output Value<br><br>Value Description<br>0 VIN- > VIN+<br>1 VIN- < VIN+<br><br>VIN - is the voltage on the Cn - pin. VIN+ is the voltage on the Cn+ pin, the C0+ pin, or the internal voltage reference ($V_{IREF}$) as defined by the ASRCP bit in the **ACCTL** register. |
| 0 | reserved | RO | 0 | 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. |

## Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x024

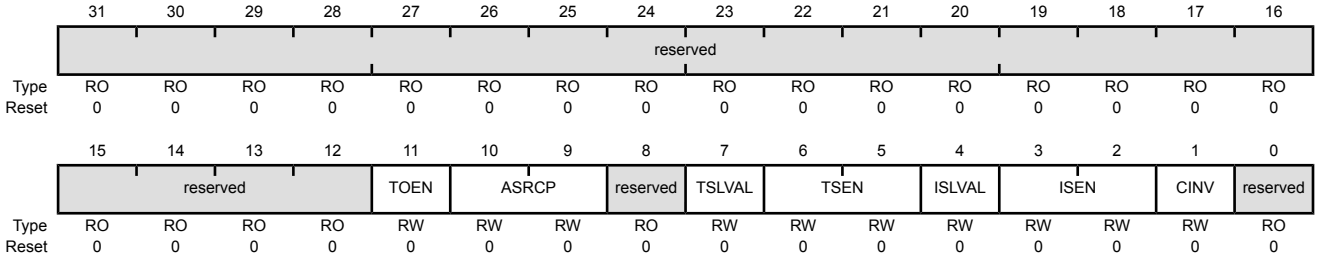
## Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044

## Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064

These registers configure the comparator's input and output.

Analog Comparator Control n (ACCTLn)

Base 0x4003.C000
Offset 0x024
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |
| 11 | TOEN | RW | 0 | Trigger Output Enable<br><br>Value Description<br>0 ADC events are suppressed and not sent to the ADC.<br>1 ADC events are sent to the ADC. |
| 10:9 | ASRCP | RW | 0x0 | Analog Source Positive<br>The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows:<br><br>Value Description<br>0x0 Pin value of Cn+<br>0x1 Pin value of C0+<br>0x2 Internal voltage reference ($V_{IREF}$)<br>0x3 Reserved |
| 8 | reserved | RO | 0 | 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. |
| 7 | TSLVAL | RW | 0 | Trigger Sense Level Value<br><br>Value Description<br>0 An ADC event is generated if the comparator output is Low.<br>1 An ADC event is generated if the comparator output is High. |

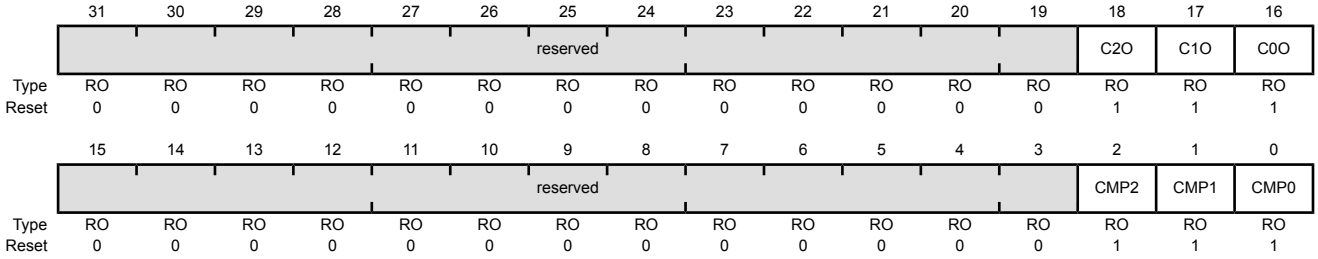
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6:5 | TSEN | RW | 0x0 | Trigger Sense<br>The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows:<br><br>Value Description<br>0x0 Level sense, see TSLVAL<br>0x1 Falling edge<br>0x2 Rising edge<br>0x3 Either edge |
| 4 | ISLVAL | RW | 0 | Interrupt Sense Level Value<br><br>Value Description<br>0 An interrupt is generated if the comparator output is Low.<br>1 An interrupt is generated if the comparator output is High. |
| 3:2 | ISEN | RW | 0x0 | Interrupt Sense<br>The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows:<br><br>Value Description<br>0x0 Level sense, see ISLVAL<br>0x1 Falling edge<br>0x2 Rising edge<br>0x3 Either edge |
| 1 | CINV | RW | 0 | Comparator Output Invert<br><br>Value Description<br>0 The output of the comparator is unchanged.<br>1 The output of the comparator is inverted prior to being processed by hardware. |
| 0 | reserved | RO | 0 | 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. |

## Register 11: Analog Comparator Peripheral Properties (ACMPPP), offset 0xFC0

The **ACMPPP** register provides information regarding the properties of the analog comparator module.

Analog Comparator Peripheral Properties (ACMPPP)

Base 0x4003.C000
Offset 0xFC0
Type RO, reset 0x0007.0007




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x0 | 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. |
| 18 | C2O | RO | 0x1 | Comparator Output 2 Present<br>Value Description<br>0 Comparator output 2 is not present.<br>1 Comparator output 2 is present. |
| 17 | C1O | RO | 0x1 | Comparator Output 1 Present<br>Value Description<br>0 Comparator output 1 is not present.<br>1 Comparator output 1 is present. |
| 16 | C0O | RO | 0x1 | Comparator Output 0 Present<br>Value Description<br>0 Comparator output 0 is not present.<br>1 Comparator output 0 is present. |
| 15:3 | reserved | RO | 0x0 | 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. |
| 2 | CMP2 | RO | 0x1 | Comparator 2 Present<br>Value Description<br>0 Comparator 2 is not present.<br>1 Comparator 2 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CMP1 | RO | 0x1 | Comparator 1 Present<br><br>Value Description<br>0 Comparator 1 is not present.<br>1 Comparator 1 is present. |
| 0 | CMP0 | RO | 0x1 | Comparator 0 Present<br><br>Value Description<br>0 Comparator 0 is not present.<br>1 Comparator 0 is present. |

# 28 Pulse Width Modulator (PWM)

Pulse width modulation (PWM) is a powerful technique for digitally encoding analog signal levels. High-resolution counters are used to generate a square wave, and the duty cycle of the square wave is modulated to encode an analog signal. Typical applications include switching power supplies and motor control.

The TM4C129XKCZAD microcontroller contains one PWM module, with four PWM generator blocks and a control block, for a total of 8 PWM outputs. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that share the same timer and frequency and can either be programmed with independent actions or as a single pair of complementary signals with dead-band delays inserted. The output signals, pwmA' and pwmB', of the PWM generation blocks are managed by the output control block before being passed to the device pins as `MnPWM0` and `MnPWM1` or `MnPWM2` and `MnPWM3`, and so on.

The TM4C129XKCZAD PWM module provides a great deal of flexibility and can generate simple PWM signals, such as those required by a simple charge pump as well as paired PWM signals with dead-band delays, such as those required by a half-H bridge driver. Three generator blocks can also generate the full six channels of gate controls required by a 3-phase inverter bridge.

Each PWM generator block has the following features:

- Four fault-condition handling inputs to quickly provide low-latency shutdown and prevent damage to the motor being controlled
- One 16-bit counter
  - Runs in Down or Up/Down mode
  - Output frequency controlled by a 16-bit load value
  - Load value updates can be synchronized
  - Produces output signals at zero and load value
- Two PWM comparators
  - Comparator value updates can be synchronized
  - Produces output signals on match
- PWM signal generator
  - Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals
  - Produces two independent PWM signals
- Dead-band generator
  - Produces two PWM signals with programmable dead-band delays suitable for driving a half-H bridge
  - Can be bypassed, leaving input PWM signals unmodified

- Can initiate an ADC sample sequence

The control block determines the polarity of the PWM signals and which signals are passed through to the pins. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins. The PWM control block has the following options:

- PWM output enable of each PWM signal
- Optional output inversion of each PWM signal (polarity control)
- Optional fault handling for each PWM signal
- Synchronization of timers in the PWM generator blocks
- Synchronization of timer/comparator updates across the PWM generator blocks
- Extended PWM synchronization of timer/comparator updates across the PWM generator blocks
- Interrupt status summary of the PWM generator blocks
- Extended PWM fault handling, with multiple fault signals, programmable polarities, and filtering
- PWM generators can be operated independently or synchronized with other generators

## 28.1 Block Diagram

Figure 28-1 on page 1927 provides the TM4C129XKCZAD PWM module diagram and Figure 28-2 on page 1927 provides a more detailed diagram of a TM4C129XKCZAD PWM generator. The TM4C129XKCZAD controller contains four generator blocks that generate eight independent PWM signals or four paired PWM signals with dead-band delays inserted.

**Figure 28-1. PWM Module Diagram**

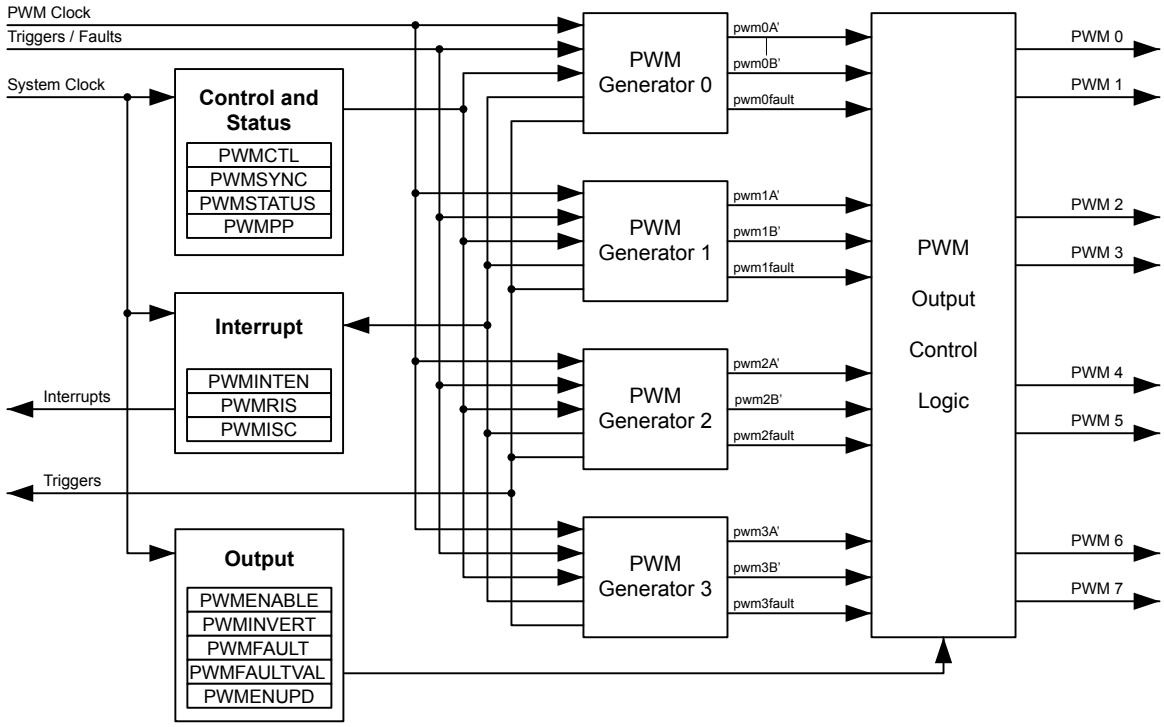
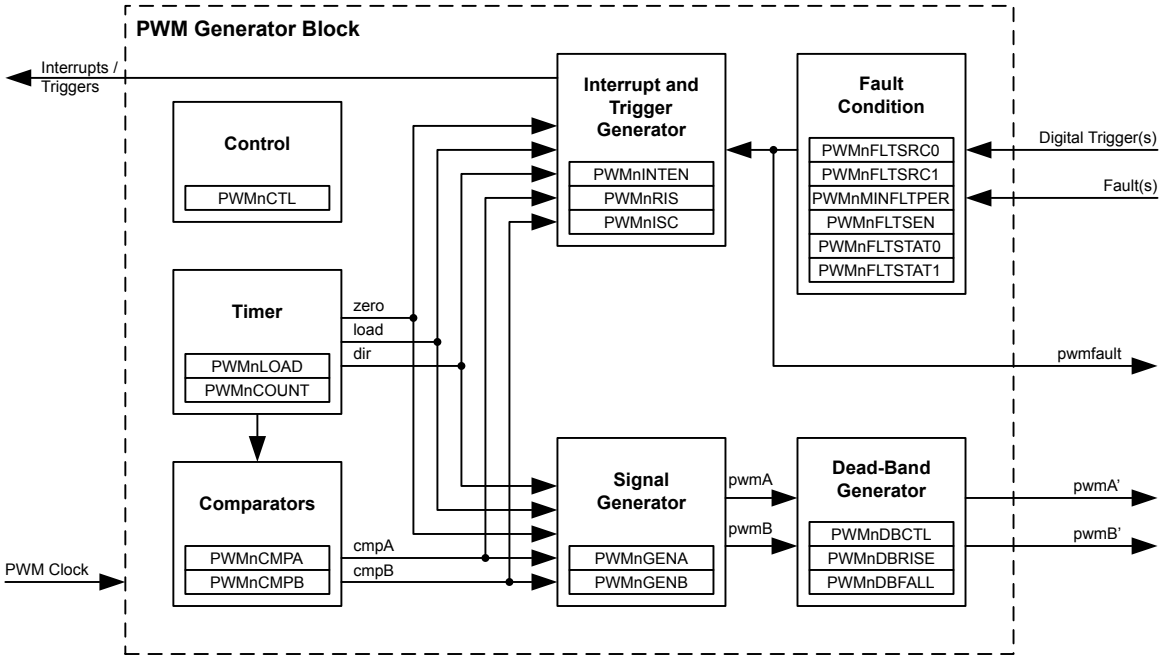



**Figure 28-2. PWM Generator Block Diagram**



## 28.2 Signal Description

The following table lists the external signals of the PWM module and describes the function of each. The PWM controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these PWM signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the PWM function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the PWM signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 28-1. PWM Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| M0FAULT0 | V7<br>D12 | PF4 (6)<br>PS0 (6) | I | TTL | Motion Control Module 0 PWM Fault 0. |
| M0FAULT1 | V16<br>D13 | PK6 (6)<br>PS1 (6) | I | TTL | Motion Control Module 0 PWM Fault 1. |
| M0FAULT2 | W16<br>B14 | PK7 (6)<br>PS2 (6) | I | TTL | Motion Control Module 0 PWM Fault 2. |
| M0FAULT3 | G16<br>A14 | PL0 (6)<br>PS3 (6) | I | TTL | Motion Control Module 0 PWM Fault 3. |
| M0PWM0 | U6<br>N5 | PF0 (6)<br>PR0 (6) | O | TTL | Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0. |
| M0PWM1 | V6<br>N4 | PF1 (6)<br>PR1 (6) | O | TTL | Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0. |
| M0PWM2 | W6<br>N2 | PF2 (6)<br>PR2 (6) | O | TTL | Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1. |
| M0PWM3 | T7<br>V8 | PF3 (6)<br>PR3 (6) | O | TTL | Motion Control Module 0 PWM 3. This signal is controlled by Module 0 PWM Generator 1. |
| M0PWM4 | N15<br>P3 | PG0 (6)<br>PR4 (6) | O | TTL | Motion Control Module 0 PWM 4. This signal is controlled by Module 0 PWM Generator 2. |
| M0PWM5 | T14<br>P2 | PG1 (6)<br>PR5 (6) | O | TTL | Motion Control Module 0 PWM 5. This signal is controlled by Module 0 PWM Generator 2. |
| M0PWM6 | U19<br>W9 | PK4 (6)<br>PR6 (6) | O | TTL | Motion Control Module 0 PWM 6. This signal is controlled by Module 0 PWM Generator 3. |
| M0PWM7 | V17<br>R10 | PK5 (6)<br>PR7 (6) | O | TTL | Motion Control Module 0 PWM 7. This signal is controlled by Module 0 PWM Generator 3. |

## 28.3 Functional Description

### 28.3.1 Clock Configuration

The PWM has two clock source options:

- The System Clock
- A predivided System Clock

The clock source is selected by programming the USEPWM bit in the **PWM Clock Configuration (PWMCC)** register. The PWMDIV bitfield specifies the divisor of the System Clock that is used to create the PWM Clock.

### 28.3.2 PWM Timer

The timer in each PWM generator runs in one of two modes: Count-Down mode or Count-Up/Down mode. In Count-Down mode, the timer counts from the load value to zero, goes back to the load value, and continues counting down. In Count-Up/Down mode, the timer counts from zero up to the load value, back down to zero, back up to the load value, and so on. Generally, Count-Down mode is used for generating left- or right-aligned PWM signals, while the Count-Up/Down mode is used for generating center-aligned PWM signals.

The timers output three signals that are used in the PWM generation process: the direction signal (this is always Low in Count-Down mode, but alternates between Low and High in Count-Up/Down mode), a single-clock-cycle-width High pulse when the counter is zero, and a single-clock-cycle-width High pulse when the counter is equal to the load value. Note that in Count-Down mode, the zero pulse is immediately followed by the load pulse. In the figures in this chapter, these signals are labelled "dir," "zero," and "load."

### 28.3.3 PWM Comparators

Each PWM generator has two comparators that monitor the value of the counter; when either comparator matches the counter, they output a single-clock-cycle-width High pulse, labeled "cmpA" and "cmpB" in the figures in this chapter. When in Count-Up/Down mode, these comparators match both when counting up and when counting down, and thus are qualified by the counter direction signal. These qualified pulses are used in the PWM generation process. If either comparator match value is greater than the counter load value, then that comparator never outputs a High pulse.

Figure 28-3 on page 1930 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Down mode. Figure 28-4 on page 1930 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Up/Down mode. In these figures, the following definitions apply:

- LOAD is the value in the **PWMnLOAD** register
- COMPA is the value in the **PWMnCMPA** register
- COMPB is the value in the **PWMnCMPB** register
- 0 is the value zero
- load is the internal signal that has a single-clock-cycle-width High pulse when the counter is equal to the load value
- zero is the internal signal that has a single-clock-cycle-width High pulse when the counter is zero
- cmpA is the internal signal that has a single-clock-cycle-width High pulse when the counter is equal to COMPA
- cmpB is the internal signal that has a single-clock-cycle-width High pulse when the counter is equal to COMPB
- dir is the internal signal that indicates the count direction

**Figure 28-3. PWM Count-Down Mode**

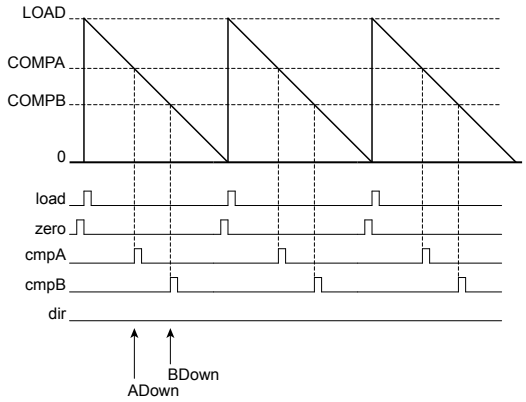
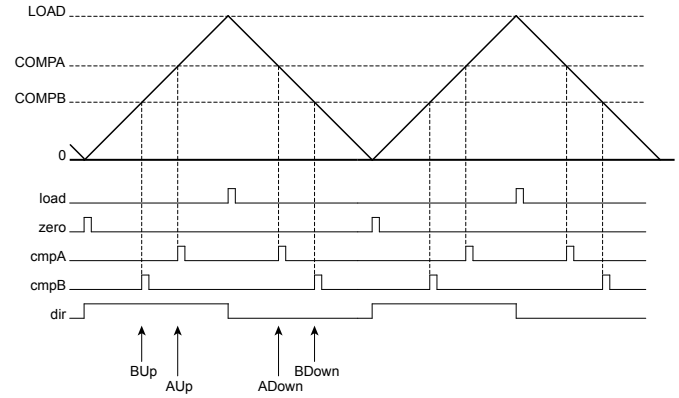



**Figure 28-4. PWM Count-Up/Down Mode**

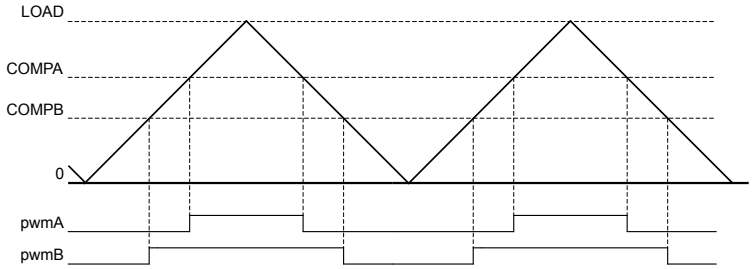



## 28.3.4 PWM Signal Generator

Each PWM generator takes the load, zero, cmpA, and cmpB pulses (qualified by the dir signal) and generates two internal PWM signals, pwmA and pwmB. In Count-Down mode, there are four events that can affect these signals: zero, load, match A down, and match B down. In Count-Up/Down mode, there are six events that can affect these signals: zero, load, match A down, match A up, match B down, and match B up. The match A or match B events are ignored when they coincide with the zero or load events. If the match A and match B events coincide, the first signal, pwmA, is generated based only on the match A event, and the second signal, pwmB, is generated based only on the match B event.

For each event, the effect on each output PWM signal is programmable: it can be left alone (ignoring the event), it can be toggled, it can be driven Low, or it can be driven High. These actions can be used to generate a pair of PWM signals of various positions and duty cycles, which do or do not overlap. Figure 28-5 on page 1931 shows the use of Count-Up/Down mode to generate a pair of center-aligned, overlapped PWM signals that have different duty cycles. This figure shows the pwmA and pwmB signals before they have passed through the dead-band generator.

**Figure 28-5. PWM Generation Example In Count-Up/Down Mode**



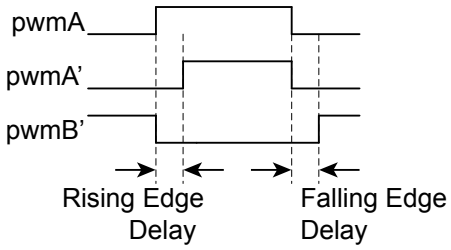

In this example, the first generator is set to drive High on match A up, drive Low on match A down, and ignore the other four events. The second generator is set to drive High on match B up, drive Low on match B down, and ignore the other four events. Changing the value of comparator A changes the duty cycle of the pwmA signal, and changing the value of comparator B changes the duty cycle of the pwmB signal.

### 28.3.5 Dead-Band Generator

The pwmA and pwmB signals produced by each PWM generator are passed to the dead-band generator. If the dead-band generator is disabled, the PWM signals simply pass through to the pwmA' and pwmB' signals unmodified. If the dead-band generator is enabled, the pwmB signal is lost and two PWM signals are generated based on the pwmA signal. The first output PWM signal, pwmA' is the pwmA signal with the rising edge delayed by a programmable amount. The second output PWM signal, pwmB', is the inversion of the pwmA signal with a programmable delay added between the falling edge of the pwmA signal and the rising edge of the pwmB' signal.

The resulting signals are a pair of active High signals where one is always High, except for a programmable amount of time at transitions where both are Low. These signals are therefore suitable for driving a half-H bridge, with the dead-band delays preventing shoot-through current from damaging the power electronics. Figure 28-6 on page 1931 shows the effect of the dead-band generator on the pwmA signal and the resulting pwmA' and pwmB' signals that are transmitted to the output control block.

**Figure 28-6. PWM Dead-Band Generator**




### 28.3.6 Interrupt/ADC-Trigger Selector

Each PWM generator also takes the same four (or six) counter events and uses them to generate an interrupt or an ADC trigger. Any of these events or a set of these events can be selected as a source for an interrupt; when any of the selected events occur, an interrupt is generated. Additionally, the same event, a different event, the same set of events, or a different set of events can be selected as a source for an ADC trigger; when any of these selected events occur, an ADC trigger pulse is generated. The selection of events allows the interrupt or ADC trigger to occur at a specific position

within the pwmA or pwmB signal. Note that interrupts and ADC triggers are based on the raw events; delays in the PWM signal edges caused by the dead-band generator are not taken into account.

### 28.3.7 Synchronization Methods

The PWM module provides four PWM generators, each providing two PWM outputs that may be used in a wide variety of applications. Generally speaking, the PWM is used in one of two categories of operation:

- **Unsynchronized.** The PWM generator and its two output signals are used alone, independent of other PWM generators.
- **Synchronized.** The PWM generator and its two outputs signals are used in conjunction with other PWM generators using a common, unified time base. If multiple PWM generators are configured with the same counter load value, synchronization can be used to guarantee that they also have the same count value (the PWM generators must be configured before they are synchronized). With this feature, more than two MnPWMn signals can be produced with a known relationship between the edges of those signals because the counters always have the same values. Other states in the module provide mechanisms to maintain the common time base and mutual synchronization.

The counter in a PWM generator can be reset to zero by writing the **PWM Time Base Sync (PWMSYNC)** register and setting the SYNCn bit associated with the generator. Multiple PWM generators can be synchronized together by setting all necessary SYNCn bits in one access. For example, setting the SYNC0 and SYNC1 bits in the **PWMSYNC** register causes the counters in PWM generators 0 and 1 to reset together.

Additional synchronization can occur between multiple PWM generators by updating register contents in one of the following three ways:

- **Immediately.** The write value has immediate effect, and the hardware reacts immediately.
- **Locally Synchronized.** The write value does not affect the logic until the counter reaches the value zero at the end of the PWM cycle. In this case, the effect of the write is deferred, providing a guaranteed defined behavior and preventing overly short or overly long output PWM pulses.
- **Globally Synchronized.** The write value does not affect the logic until two sequential events have occurred: (1) the Update mode for the generator function is programmed for global synchronization in the **PWMnCTL** register, and (2) the counter reaches zero at the end of the PWM cycle. In this case, the effect of the write is deferred until the end of the PWM cycle following the end of all updates. This mode allows multiple items in multiple PWM generators to be updated simultaneously without odd effects during the update; everything runs from the old values until a point at which they all run from the new values. The Update mode of the load and comparator match values can be individually configured in each PWM generator block. It typically makes sense to use the synchronous update mechanism across PWM generator blocks when the timers in those blocks are synchronized, although this is not required in order for this mechanism to function properly.

The following registers provide either local or global synchronization based on the state of various Update mode bits and fields in the **PWMnCTL** register (LOADUPD; CMPAUPD; CMPBUPD):

- Generator Registers: **PWMnLOAD**, **PWMnCMPA**, and **PWMnCMPB**

The following registers default to immediate update, but are provided with the optional functionality of synchronously updating rather than having all updates take immediate effect:

- Module-Level Register: **PWMENABLE** (based on the state of the ENUPDn bits in the PWMENUPD register).
- Generator Register: **PWMnGENA**, **PWMnGENB**, **PWMnDBCTL**, **PWMnDBRISE**, and **PWMnDBFALL** (based on the state of various Update mode bits and fields in the **PWMnCTL** register (GENAUPD; GENBUPD; DBCTLUPD; DBRISEUPD; DBFALLUPD)).

All other registers are considered statically provisioned for the execution of an application or are used dynamically for purposes unrelated to maintaining synchronization and therefore do not need synchronous update functionality.

### 28.3.8 Fault Conditions

A fault condition is one in which the controller must be signaled to stop normal PWM function and then set the MnPWMn signals to a safe state. Two basic situations cause fault conditions:

- The microcontroller is stalled and cannot perform the necessary computation in the time required for motion control
- An external error or event is detected

The PWM generator can use the following inputs to generate a fault condition, including:

- MnFAULTn pin assertion
- A stall of the controller generated by the debugger
- The trigger of an ADC digital comparator

Fault conditions are calculated on a per-PWM generator basis. Each PWM generator configures the necessary conditions to indicate a fault condition exists. This method allows the development of applications with dependent and independent control.

Four fault input pins (MnFAULTn) are available. These inputs may be used with circuits that generate an active High or active Low signal to indicate an error condition. A MnFAULTn pins may be individually programmed for the appropriate logic sense using the **PWMnFLTSEN** register.

The PWM generator's mode control, including fault condition handling, is provided in the **PWMnCTL** register. This register determines whether the input or a combination of MnFAULTn input signals and/or digital comparator triggers (as configured by the **PWMnFLTSRC0** and **PWMnFLTSRC1** registers) is used to generate a fault condition. The **PWMnCTL** register also selects whether the fault condition is maintained as long as the external condition lasts or if it is latched until the fault condition until cleared by software. Finally, this register also enables a counter that may be used to extend the period of a fault condition for external events to assure that the duration is a minimum length. The minimum fault period count is specified in the **PWMnMINFLTPER** register.

**Note:** When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the **PWMnCTL** register should be set to 1 to ensure trigger assertions are captured.

Status regarding the specific fault cause is provided in the **PWMnFLTSTAT0** and **PWMnFLTSTAT1** registers. Note that the fault status registers, **PWMnFLTSTAT0** and **PWMnFLTSTAT1**, reflect the status of all fault sources, regardless of what fault sources are enabled for that particular generator.

PWM generator fault conditions may be promoted to a controller interrupt using the **PWMINTEN** register.

### 28.3.9 Output Control Block

The output control block takes care of the final conditioning of the pwmA' and pwmB' signals before they go to the pins as the MnPWMn signals. Via a single register, the **PWM Output Enable (PWNENABLE)** register, the set of PWM signals that are actually enabled to the pins can be modified. This function can be used, for example, to perform commutation of a brushless DC motor with a single register write (and without modifying the individual PWM generators, which are modified by the feedback control loop). In addition, the updating of the bits in the **PWMENABLE** register can be configured to be immediate or locally or globally synchronized to the next synchronous update using the **PWM Enable Update (PWMENUPD)** register.

During fault conditions, the PWM output signals, MnPWMn, usually must be driven to safe values so that external equipment may be safely controlled. The **PWMFAULT** register specifies whether during a fault condition, the generated signal continues to be passed driven or to an encoding specified in the **PWMFAULTVAL** register.

A final inversion can be applied to any of the MnPWMn signals, making them active Low instead of the default active High using the **PWM Output Inversion (PWMINVERT)**. The inversion is applied even if a value has been enabled in the **PWMFAULT** register and specified in the **PWMFAULTVAL** register. In other words, if a bit is set in the **PWMFAULT**, **PWMFAULTVAL**, and **PWMINVERT** registers, the output on the MnPWMn signal is 0, not 1 as specified in the **PWMFAULTVAL** register.

## 28.4 Initialization and Configuration

The following example shows how to initialize PWM Generator 0 with a 25-kHz frequency, a 25% duty cycle on the MnPWM0 pin, and a 75% duty cycle on the MnPWM1 pin. This example assumes the system clock is 20 MHz.

1. Enable the PWM clock by setting its corresponding bit in the **RCGCPWM** register in the System Control module (see page 415).
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register in the System Control module (see page 399).
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. To determine which GPIOs to configure, see Table 31-4 on page 2064.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the PWM signals to the appropriate pins (see page 820 and Table 31-5 on page 2081).
5. Configure the **PWM Clock Configuration (PWMCC)** register to use the PWM divide (USEPWMDIV) and set the divider (PWMDIV) to divide by 2 (0x0).
6. Configure the PWM generator for countdown mode with immediate updates to the parameters.
   - Write the **PWM0CTL** register with a value of 0x0000.0000.
   - Write the **PWM0GENA** register with a value of 0x0000.008C.
   - Write the **PWM0GENB** register with a value of 0x0000.080C.
7. Set the period. For a 25-KHz frequency, the period = 1/25,000, or 40 microseconds. The PWM clock source is 10 MHz; the system clock divided by 2. Thus there are 400 clock ticks per period. Use this value to set the **PWM0LOAD** register. In Count-Down mode, set the LOAD field in the **PWM0LOAD** register to the requested period minus one.

- Write the **PWM0LOAD** register with a value of 0x0000.018F.

8. Set the pulse width of the MnPWM0 pin for a 25% duty cycle.
   - Write the **PWM0CMPA** register with a value of 0x0000.012B.
9. Set the pulse width of the MnPWM1 pin for a 75% duty cycle.
   - Write the **PWM0CMPB** register with a value of 0x0000.0063.
10. Start the timers in PWM generator 0.
    - Write the **PWM0CTL** register with a value of 0x0000.0001.
11. Enable PWM outputs.
    - Write the **PWMENABLE** register with a value of 0x0000.0003.

## 28.5 Register Map

Table 28-2 on page 1935 lists the PWM registers. The offset listed is a hexadecimal increment to the register's address, relative to the PWM module's base address:

- PWM0: 0x4002.8000

Note that the PWM module clock must be enabled before the registers can be programmed. There must be a delay of 3 system clocks after the PWM module clock is enabled before any PWM module registers are accessed.

**Table 28-2. PWM Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | PWMCTL | RW | 0x0000.0000 | PWM Master Control | 1939 |
| 0x004 | PWMSYNC | RW | 0x0000.0000 | PWM Time Base Sync | 1941 |
| 0x008 | PWMENABLE | RW | 0x0000.0000 | PWM Output Enable | 1942 |
| 0x00C | PWMINVERT | RW | 0x0000.0000 | PWM Output Inversion | 1944 |
| 0x010 | PWMFAULT | RW | 0x0000.0000 | PWM Output Fault | 1946 |
| 0x014 | PWMINTEN | RW | 0x0000.0000 | PWM Interrupt Enable | 1948 |
| 0x018 | PWMRIS | RO | 0x0000.0000 | PWM Raw Interrupt Status | 1950 |
| 0x01C | PWMISC | RW1C | 0x0000.0000 | PWM Interrupt Status and Clear | 1953 |
| 0x020 | PWMSTATUS | RO | 0x0000.0000 | PWM Status | 1956 |
| 0x024 | PWMFAULTVAL | RW | 0x0000.0000 | PWM Fault Condition Value | 1958 |
| 0x028 | PWMENUPD | RW | 0x0000.0000 | PWM Enable Update | 1960 |
| 0x040 | PWM0CTL | RW | 0x0000.0000 | PWM0 Control | 1964 |
| 0x044 | PWM0INTEN | RW | 0x0000.0000 | PWM0 Interrupt and Trigger Enable | 1969 |
| 0x048 | PWM0RIS | RO | 0x0000.0000 | PWM0 Raw Interrupt Status | 1972 |

**Table 28-2. PWM Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x04C | PWM0ISC | RW1C | 0x0000.0000 | PWM0 Interrupt Status and Clear | 1974 |
| 0x050 | PWM0LOAD | RW | 0x0000.0000 | PWM0 Load | 1976 |
| 0x054 | PWM0COUNT | RO | 0x0000.0000 | PWM0 Counter | 1977 |
| 0x058 | PWM0CMPA | RW | 0x0000.0000 | PWM0 Compare A | 1978 |
| 0x05C | PWM0CMPB | RW | 0x0000.0000 | PWM0 Compare B | 1979 |
| 0x060 | PWM0GENA | RW | 0x0000.0000 | PWM0 Generator A Control | 1980 |
| 0x064 | PWM0GENB | RW | 0x0000.0000 | PWM0 Generator B Control | 1983 |
| 0x068 | PWM0DBCTL | RW | 0x0000.0000 | PWM0 Dead-Band Control | 1986 |
| 0x06C | PWM0DBRISE | RW | 0x0000.0000 | PWM0 Dead-Band Rising-Edge Delay | 1987 |
| 0x070 | PWM0DBFALL | RW | 0x0000.0000 | PWM0 Dead-Band Falling-Edge-Delay | 1988 |
| 0x074 | PWM0FLTSRC0 | RW | 0x0000.0000 | PWM0 Fault Source 0 | 1989 |
| 0x078 | PWM0FLTSRC1 | RW | 0x0000.0000 | PWM0 Fault Source 1 | 1991 |
| 0x07C | PWM0MINFLTPER | RW | 0x0000.0000 | PWM0 Minimum Fault Period | 1994 |
| 0x080 | PWM1CTL | RW | 0x0000.0000 | PWM1 Control | 1964 |
| 0x084 | PWM1INTEN | RW | 0x0000.0000 | PWM1 Interrupt and Trigger Enable | 1969 |
| 0x088 | PWM1RIS | RO | 0x0000.0000 | PWM1 Raw Interrupt Status | 1972 |
| 0x08C | PWM1ISC | RW1C | 0x0000.0000 | PWM1 Interrupt Status and Clear | 1974 |
| 0x090 | PWM1LOAD | RW | 0x0000.0000 | PWM1 Load | 1976 |
| 0x094 | PWM1COUNT | RO | 0x0000.0000 | PWM1 Counter | 1977 |
| 0x098 | PWM1CMPA | RW | 0x0000.0000 | PWM1 Compare A | 1978 |
| 0x09C | PWM1CMPB | RW | 0x0000.0000 | PWM1 Compare B | 1979 |
| 0x0A0 | PWM1GENA | RW | 0x0000.0000 | PWM1 Generator A Control | 1980 |
| 0x0A4 | PWM1GENB | RW | 0x0000.0000 | PWM1 Generator B Control | 1983 |
| 0x0A8 | PWM1DBCTL | RW | 0x0000.0000 | PWM1 Dead-Band Control | 1986 |
| 0x0AC | PWM1DBRISE | RW | 0x0000.0000 | PWM1 Dead-Band Rising-Edge Delay | 1987 |
| 0x0B0 | PWM1DBFALL | RW | 0x0000.0000 | PWM1 Dead-Band Falling-Edge-Delay | 1988 |
| 0x0B4 | PWM1FLTSRC0 | RW | 0x0000.0000 | PWM1 Fault Source 0 | 1989 |
| 0x0B8 | PWM1FLTSRC1 | RW | 0x0000.0000 | PWM1 Fault Source 1 | 1991 |
| 0x0BC | PWM1MINFLTPER | RW | 0x0000.0000 | PWM1 Minimum Fault Period | 1994 |
| 0x0C0 | PWM2CTL | RW | 0x0000.0000 | PWM2 Control | 1964 |
| 0x0C4 | PWM2INTEN | RW | 0x0000.0000 | PWM2 Interrupt and Trigger Enable | 1969 |
| 0x0C8 | PWM2RIS | RO | 0x0000.0000 | PWM2 Raw Interrupt Status | 1972 |

**Table 28-2. PWM Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x0CC | PWM2ISC | RW1C | 0x0000.0000 | PWM2 Interrupt Status and Clear | 1974 |
| 0x0D0 | PWM2LOAD | RW | 0x0000.0000 | PWM2 Load | 1976 |
| 0x0D4 | PWM2COUNT | RO | 0x0000.0000 | PWM2 Counter | 1977 |
| 0x0D8 | PWM2CMPA | RW | 0x0000.0000 | PWM2 Compare A | 1978 |
| 0x0DC | PWM2CMPB | RW | 0x0000.0000 | PWM2 Compare B | 1979 |
| 0x0E0 | PWM2GENA | RW | 0x0000.0000 | PWM2 Generator A Control | 1980 |
| 0x0E4 | PWM2GENB | RW | 0x0000.0000 | PWM2 Generator B Control | 1983 |
| 0x0E8 | PWM2DBCTL | RW | 0x0000.0000 | PWM2 Dead-Band Control | 1986 |
| 0x0EC | PWM2DBRISE | RW | 0x0000.0000 | PWM2 Dead-Band Rising-Edge Delay | 1987 |
| 0x0F0 | PWM2DBFALL | RW | 0x0000.0000 | PWM2 Dead-Band Falling-Edge-Delay | 1988 |
| 0x0F4 | PWM2FLTSRC0 | RW | 0x0000.0000 | PWM2 Fault Source 0 | 1989 |
| 0x0F8 | PWM2FLTSRC1 | RW | 0x0000.0000 | PWM2 Fault Source 1 | 1991 |
| 0x0FC | PWM2MINFLTPER | RW | 0x0000.0000 | PWM2 Minimum Fault Period | 1994 |
| 0x100 | PWM3CTL | RW | 0x0000.0000 | PWM3 Control | 1964 |
| 0x104 | PWM3INTEN | RW | 0x0000.0000 | PWM3 Interrupt and Trigger Enable | 1969 |
| 0x108 | PWM3RIS | RO | 0x0000.0000 | PWM3 Raw Interrupt Status | 1972 |
| 0x10C | PWM3ISC | RW1C | 0x0000.0000 | PWM3 Interrupt Status and Clear | 1974 |
| 0x110 | PWM3LOAD | RW | 0x0000.0000 | PWM3 Load | 1976 |
| 0x114 | PWM3COUNT | RO | 0x0000.0000 | PWM3 Counter | 1977 |
| 0x118 | PWM3CMPA | RW | 0x0000.0000 | PWM3 Compare A | 1978 |
| 0x11C | PWM3CMPB | RW | 0x0000.0000 | PWM3 Compare B | 1979 |
| 0x120 | PWM3GENA | RW | 0x0000.0000 | PWM3 Generator A Control | 1980 |
| 0x124 | PWM3GENB | RW | 0x0000.0000 | PWM3 Generator B Control | 1983 |
| 0x128 | PWM3DBCTL | RW | 0x0000.0000 | PWM3 Dead-Band Control | 1986 |
| 0x12C | PWM3DBRISE | RW | 0x0000.0000 | PWM3 Dead-Band Rising-Edge Delay | 1987 |
| 0x130 | PWM3DBFALL | RW | 0x0000.0000 | PWM3 Dead-Band Falling-Edge-Delay | 1988 |
| 0x134 | PWM3FLTSRC0 | RW | 0x0000.0000 | PWM3 Fault Source 0 | 1989 |
| 0x138 | PWM3FLTSRC1 | RW | 0x0000.0000 | PWM3 Fault Source 1 | 1991 |
| 0x13C | PWM3MINFLTPER | RW | 0x0000.0000 | PWM3 Minimum Fault Period | 1994 |
| 0x800 | PWM0FLTSEN | RW | 0x0000.0000 | PWM0 Fault Pin Logic Sense | 1995 |
| 0x804 | PWM0FLTSTAT0 | - | 0x0000.0000 | PWM0 Fault Status 0 | 1996 |
| 0x808 | PWM0FLTSTAT1 | - | 0x0000.0000 | PWM0 Fault Status 1 | 1998 |

**Table 28-2. PWM Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x880 | PWM1FLTSEN | RW | 0x0000.0000 | PWM1 Fault Pin Logic Sense | 1995 |
| 0x884 | PWM1FLTSTAT0 | - | 0x0000.0000 | PWM1 Fault Status 0 | 1996 |
| 0x888 | PWM1FLTSTAT1 | - | 0x0000.0000 | PWM1 Fault Status 1 | 1998 |
| 0x900 | PWM2FLTSEN | RW | 0x0000.0000 | PWM2 Fault Pin Logic Sense | 1995 |
| 0x904 | PWM2FLTSTAT0 | - | 0x0000.0000 | PWM2 Fault Status 0 | 1996 |
| 0x908 | PWM2FLTSTAT1 | - | 0x0000.0000 | PWM2 Fault Status 1 | 1998 |
| 0x980 | PWM3FLTSEN | RW | 0x0000.0000 | PWM3 Fault Pin Logic Sense | 1995 |
| 0x984 | PWM3FLTSTAT0 | - | 0x0000.0000 | PWM3 Fault Status 0 | 1996 |
| 0x988 | PWM3FLTSTAT1 | - | 0x0000.0000 | PWM3 Fault Status 1 | 1998 |
| 0xFC0 | PWMPP | RO | 0x0000.0344 | PWM Peripheral Properties | 2001 |
| 0xFC8 | PWMCC | RW | 0x0000.0005 | PWM Clock Configuration | 2003 |

## 28.6 Register Descriptions

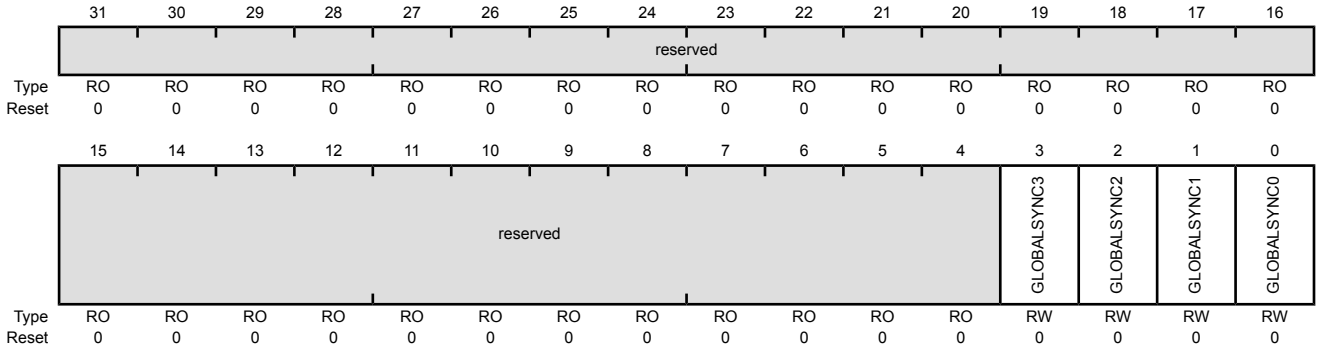
The remainder of this section lists and describes the PWM registers, in numerical order by address offset.

## Register 1: PWM Master Control (PWMCTL), offset 0x000

This register provides master control over the PWM generation blocks.

PWM Master Control (PWMCTL)

PWM0 base: 0x4002.8000
Offset 0x000
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000 | 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. |
| 3 | GLOBALSYNC3 | RW | 0 | Update PWM Generator 3<br><br>Value Description<br>0 No effect.<br>1 Any queued update to a load or comparator register in PWM generator 3 is applied the next time the corresponding counter becomes zero.<br><br>This bit automatically clears when the updates have completed; it cannot be cleared by software. |
| 2 | GLOBALSYNC2 | RW | 0 | Update PWM Generator 2<br><br>Value Description<br>0 No effect.<br>1 Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.<br><br>This bit automatically clears when the updates have completed; it cannot be cleared by software. |

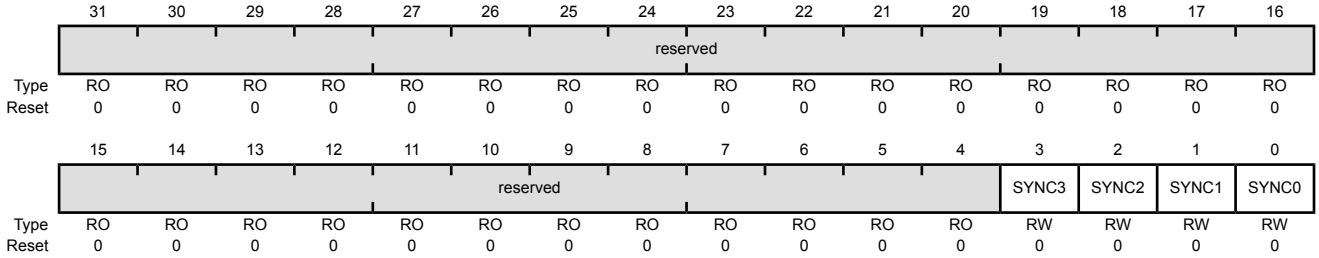
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | GLOBALSYNC1 | RW | 0 | Update PWM Generator 1<br><br>Value Description<br>0 No effect.<br>1 Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.<br><br>This bit automatically clears when the updates have completed; it cannot be cleared by software. |
| 0 | GLOBALSYNC0 | RW | 0 | Update PWM Generator 0<br><br>Value Description<br>0 No effect.<br>1 Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.<br><br>This bit automatically clears when the updates have completed; it cannot be cleared by software. |

## Register 2: PWM Time Base Sync (PWMSYNC), offset 0x004

This register provides a method to perform synchronization of the counters in the PWM generation blocks. Setting a bit in this register causes the specified counter to reset back to 0; setting multiple bits resets multiple counters simultaneously. The bits auto-clear after the reset has occurred; reading them back as zero indicates that the synchronization has completed.

PWM Time Base Sync (PWMSYNC)

PWM0 base: 0x4002.8000
Offset 0x004
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | SYNC3 | RW | 0 | Reset Generator 3 Counter<br>Value Description<br>0 No effect.<br>1 Resets the PWM generator 3 counter. |
| 2 | SYNC2 | RW | 0 | Reset Generator 2 Counter<br>Value Description<br>0 No effect.<br>1 Resets the PWM generator 2 counter. |
| 1 | SYNC1 | RW | 0 | Reset Generator 1 Counter<br>Value Description<br>0 No effect.<br>1 Resets the PWM generator 1 counter. |
| 0 | SYNC0 | RW | 0 | Reset Generator 0 Counter<br>Value Description<br>0 No effect.<br>1 Resets the PWM generator 0 counter. |

## Register 3: PWM Output Enable (PWMENABLE), offset 0x008

This register provides a master control of which generated pwmA' and pwmB' signals are output to the MnPWMn pins. By disabling a PWM output, the generation process can continue (for example, when the time bases are synchronized) without driving PWM signals to the pins. When bits in this register are set, the corresponding pwmA' or pwmB' signal is passed through to the output stage. When bits are clear, the pwmA' or pwmB' signal is replaced by a zero value which is also passed to the output stage. The **PWMINVERT** register controls the output stage, so if the corresponding bit is set in that register, the value seen on the MnPWMn signal is inverted from what is configured by the bits in this register. Updates to the bits in this register can be immediate or locally or globally synchronized to the next synchronous update as controlled by the ENUPDn fields in the **PWMENUPD** register.

PWM Output Enable (PWMENABLE)

PWM0 base: 0x4002.8000
Offset 0x008
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | PWM7EN | PWM6EN | PWM5EN | PWM4EN | PWM3EN | PWM2EN | PWM1EN | PWM0EN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | PWM7EN | RW | 0 | MnPWM7 Output Enable<br><br>Value Description<br>0 The MnPWM7 signal has a zero value.<br>1 The generated pwm3B' signal is passed to the MnPWM7 pin. |
| 6 | PWM6EN | RW | 0 | MnPWM6 Output Enable<br><br>Value Description<br>0 The MnPWM6 signal has a zero value.<br>1 The generated pwm3A' signal is passed to the MnPWM6 pin. |
| 5 | PWM5EN | RW | 0 | MnPWM5 Output Enable<br><br>Value Description<br>0 The MnPWM5 signal has a zero value.<br>1 The generated pwm2B' signal is passed to the MnPWM5 pin. |

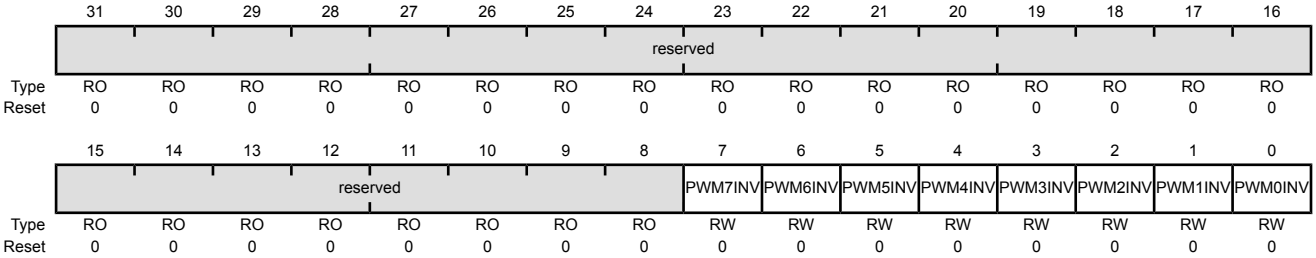
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | PWM4EN | RW | 0 | MnPWM4 Output Enable<br>Value Description<br>0 The MnPWM4 signal has a zero value.<br>1 The generated pwm2A' signal is passed to the MnPWM4 pin. |
| 3 | PWM3EN | RW | 0 | MnPWM3 Output Enable<br>Value Description<br>0 The MnPWM3 signal has a zero value.<br>1 The generated pwm1B' signal is passed to the MnPWM3 pin. |
| 2 | PWM2EN | RW | 0 | MnPWM2 Output Enable<br>Value Description<br>0 The MnPWM2 signal has a zero value.<br>1 The generated pwm1A' signal is passed to the MnPWM2 pin. |
| 1 | PWM1EN | RW | 0 | MnPWM1 Output Enable<br>Value Description<br>0 The MnPWM1 signal has a zero value.<br>1 The generated pwm0B' signal is passed to the MnPWM1 pin. |
| 0 | PWM0EN | RW | 0 | MnPWM0 Output Enable<br>Value Description<br>0 The MnPWM0 signal has a zero value.<br>1 The generated pwm0A' signal is passed to the MnPWM0 pin. |

## Register 4: PWM Output Inversion (PWMINVERT), offset 0x00C

This register provides a master control of the polarity of the MnPWMn signals on the device pins. The pwmA' and pwmB' signals generated by the PWM generator are active High; but can be made active Low via this register. Disabled PWM channels are also passed through the output inverter (if so configured) so that inactive signals can be High. In addition, if the **PWMFAULT** register enables a specific value to be placed on the MnPWMn signals during a fault condition, that value is inverted if the corresponding bit in this register is set.

PWM Output Inversion (PWMINVERT)

PWM0 base: 0x4002.8000
Offset 0x00C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | PWM7INV | RW | 0 | Invert MnPWM7 Signal<br>Value Description<br>0 The MnPWM7 signal is not inverted.<br>1 The MnPWM7 signal is inverted. |
| 6 | PWM6INV | RW | 0 | Invert MnPWM6 Signal<br>Value Description<br>0 The MnPWM6 signal is not inverted.<br>1 The MnPWM6 signal is inverted. |
| 5 | PWM5INV | RW | 0 | Invert MnPWM5 Signal<br>Value Description<br>0 The MnPWM5 signal is not inverted.<br>1 The MnPWM5 signal is inverted. |
| 4 | PWM4INV | RW | 0 | Invert MnPWM4 Signal<br>Value Description<br>0 The MnPWM4 signal is not inverted.<br>1 The MnPWM4 signal is inverted. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | PWM3INV | RW | 0 | Invert MnPWM3 Signal<br>Value Description<br>0 The MnPWM3 signal is not inverted.<br>1 The MnPWM3 signal is inverted. |
| 2 | PWM2INV | RW | 0 | Invert MnPWM2 Signal<br>Value Description<br>0 The MnPWM2 signal is not inverted.<br>1 The MnPWM2 signal is inverted. |
| 1 | PWM1INV | RW | 0 | Invert MnPWM1 Signal<br>Value Description<br>0 The MnPWM1 signal is not inverted.<br>1 The MnPWM1 signal is inverted. |
| 0 | PWM0INV | RW | 0 | Invert MnPWM0 Signal<br>Value Description<br>0 The MnPWM0 signal is not inverted.<br>1 The MnPWM0 signal is inverted. |

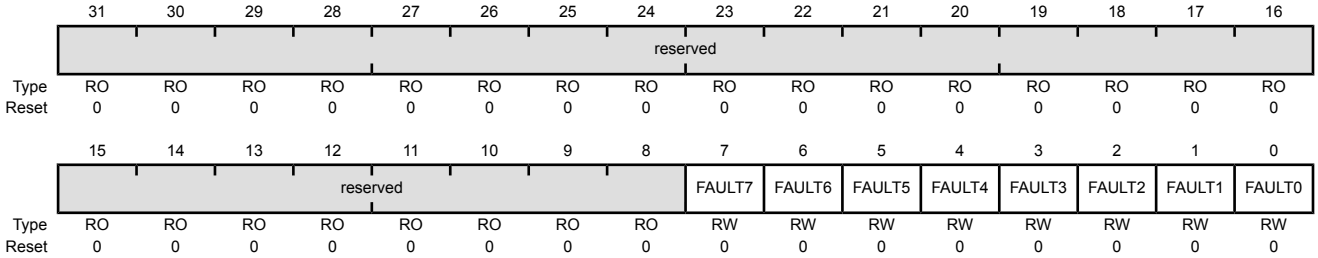
## Register 5: PWM Output Fault (PWMFAULT), offset 0x010

This register controls the behavior of the MnPWMn outputs in the presence of fault conditions. Both the fault inputs (MnFAULTn pins and digital comparator outputs) and debug events are considered fault conditions. On a fault condition, each pwmA' or pwmB' signal can be passed through unmodified or driven to the value specified by the corresponding bit in the **PWMFAULTVAL** register. For outputs that are configured for pass-through, the debug event handling on the corresponding PWM generator also determines if the pwmA' or pwmB' signal continues to be generated.

Fault condition control occurs before the output inverter, so PWM signals driven to a specified value on fault are inverted if the channel is configured for inversion (therefore, the pin is driven to the logical complement of the specified value on a fault condition).

PWM Output Fault (PWMFAULT)

PWM0 base: 0x4002.8000
Offset 0x010
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | FAULT7 | RW | 0 | MnPWM7 Fault<br>Value Description<br>0 The generated pwm3B' signal is passed to the MnPWM7 pin.<br>1 The MnPWM7 output signal is driven to the value specified by the PWM7 bit in the **PWMFAULTVAL** register. |
| 6 | FAULT6 | RW | 0 | MnPWM6 Fault<br>Value Description<br>0 The generated pwm3A' signal is passed to the MnPWM6 pin.<br>1 The MnPWM6 output signal is driven to the value specified by the PWM6 bit in the **PWMFAULTVAL** register. |
| 5 | FAULT5 | RW | 0 | MnPWM5 Fault<br>Value Description<br>0 The generated pwm2B' signal is passed to the MnPWM5 pin.<br>1 The MnPWM5 output signal is driven to the value specified by the PWM5 bit in the **PWMFAULTVAL** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | FAULT4 | RW | 0 | MnPWM4 Fault<br>Value Description<br>0 The generated pwm2A' signal is passed to the MnPWM4 pin.<br>1 The MnPWM4 output signal is driven to the value specified by the PWM4 bit in the **PWMFAULTVAL** register. |
| 3 | FAULT3 | RW | 0 | MnPWM3 Fault<br>Value Description<br>0 The generated pwm1B' signal is passed to the MnPWM3 pin.<br>1 The MnPWM3 output signal is driven to the value specified by the PWM3 bit in the **PWMFAULTVAL** register. |
| 2 | FAULT2 | RW | 0 | MnPWM2 Fault<br>Value Description<br>0 The generated pwm1A' signal is passed to the MnPWM2 pin.<br>1 The MnPWM2 output signal is driven to the value specified by the PWM2 bit in the **PWMFAULTVAL** register. |
| 1 | FAULT1 | RW | 0 | MnPWM1 Fault<br>Value Description<br>0 The generated pwm0B' signal is passed to the MnPWM1 pin.<br>1 The MnPWM1 output signal is driven to the value specified by the PWM1 bit in the **PWMFAULTVAL** register. |
| 0 | FAULT0 | RW | 0 | MnPWM0 Fault<br>Value Description<br>0 The generated pwm0A' signal is passed to the MnPWM0 pin.<br>1 The MnPWM0 output signal is driven to the value specified by the PWM0 bit in the **PWMFAULTVAL** register. |

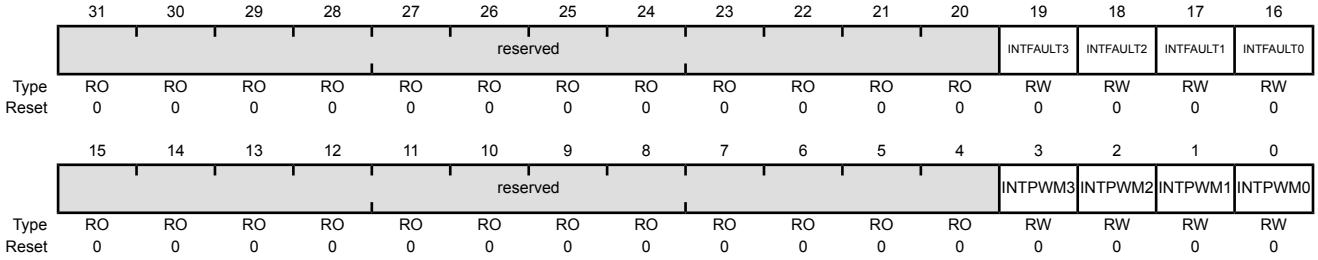
## Register 6: PWM Interrupt Enable (PWMINTEN), offset 0x014

This register controls the global interrupt generation capabilities of the PWM module. The events that can cause an interrupt are the fault input and the individual interrupts from the PWM generators.

**Note:** The "n" in the INTFAULTn and INTPWMn bits in this register correspond to the PWM generators, not to the FAULTn signals.

PWM Interrupt Enable (PWMINTEN)

PWM0 base: 0x4002.8000
Offset 0x014
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x000 | 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. |
| 19 | INTFAULT3 | RW | 0 | Interrupt Fault 3<br>Value Description<br>0 The fault condition for PWM generator 3 is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the fault condition for PWM generator 3 is asserted. |
| 18 | INTFAULT2 | RW | 0 | Interrupt Fault 2<br>Value Description<br>0 The fault condition for PWM generator 2 is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the fault condition for PWM generator 2 is asserted. |
| 17 | INTFAULT1 | RW | 0 | Interrupt Fault 1<br>Value Description<br>0 The fault condition for PWM generator 1 is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the fault condition for PWM generator 1 is asserted. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | INTFAULT0 | RW | 0 | Interrupt Fault 0<br>Value Description<br>0 The fault condition for PWM generator 0 is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the fault condition for PWM generator 0 is asserted. |
| 15:4 | reserved | RO | 0x000 | 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. |
| 3 | INTPWM3 | RW | 0 | PWM3 Interrupt Enable<br>Value Description<br>0 The PWM generator 3 interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PWM generator 3 block asserts an interrupt. |
| 2 | INTPWM2 | RW | 0 | PWM2 Interrupt Enable<br>Value Description<br>0 The PWM generator 2 interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PWM generator 2 block asserts an interrupt. |
| 1 | INTPWM1 | RW | 0 | PWM1 Interrupt Enable<br>Value Description<br>0 The PWM generator 1 interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PWM generator 1 block asserts an interrupt. |
| 0 | INTPWM0 | RW | 0 | PWM0 Interrupt Enable<br>Value Description<br>0 The PWM generator 0 interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the PWM generator 0 block asserts an interrupt. |

## Register 7: PWM Raw Interrupt Status (PWMRIS), offset 0x018

This register provides the current set of interrupt sources that are asserted, regardless of whether they are enabled to cause an interrupt to be asserted to the interrupt controller. The fault interrupt is asserted based on the fault condition source that is specified by the **PWMnCTL**, **PWMnFLTSRC0** and **PWMnFLTSRC1** registers. The fault interrupt is latched on detection and must be cleared through the **PWM Interrupt Status and Clear (PWMISC)** register. The actual value of the MnFAULTn signals can be observed using the **PWMSTATUS** register.

The PWM generator interrupts simply reflect the status of the PWM generators and are cleared via the interrupt status register in the PWM generator blocks. If a bit is set, the event is active; if a bit is clear the event is not active.

PWM Raw Interrupt Status (PWMRIS)

PWM0 base: 0x4002.8000
Offset 0x018
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | INTFAULT3 | INTFAULT2 | INTFAULT1 | INTFAULT0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | INTPWM3 | INTPWM2 | INTPWM1 | INTPWM0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x000 | 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. |
| 19 | INTFAULT3 | RO | 0 | Interrupt Fault PWM 3 |

| Value | Description |
|---|---|
| 0 | The fault condition for PWM generator 3 has not been asserted. |
| 1 | The fault condition for PWM generator 3 is asserted. |

**Note:** If the LATCH bit is set in the **PWM3CTL** register, the INTFAULT3 bit in this register can be cleared by writing a 1 to the INTFAULT3 bit in the **PWMISC** register. If the LATCH bit is 0 in the **PWM3CTL** register, writing a 1 to the INTFAULT3 bit in the **PWMISC** register has no effect.

| | | | | |
|---|---|---|---|---|
| 18 | INTFAULT2 | RO | 0 | Interrupt Fault PWM 2 |

| Value | Description |
|---|---|
| 0 | The fault condition for PWM generator 2 has not been asserted. |
| 1 | The fault condition for PWM generator 2 is asserted. |

**Note:** If the LATCH bit is set in the **PWM2CTL** register, the INTFAULT2 bit in this register can be cleared by writing a 1 to the INTFAULT2 bit in the **PWMISC** register. If the LATCH bit is 0 in the **PWM2CTL** register, writing a 1 to the INTFAULT2 bit in the **PWMISC** register has no effect.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | INTFAULT1 | RO | 0 | Interrupt Fault PWM 1<br><br>Value Description<br>0 The fault condition for PWM generator 1 has not been asserted.<br>1 The fault condition for PWM generator 1 is asserted.<br><br>**Note:** If the LATCH bit is set in the **PWM1CTL** register, the INTFAULT1 bit in this register can be cleared by writing a 1 to the INTFAULT1 bit in the **PWMISC** register. If the LATCH bit is 0 in the **PWM1CTL** register, writing a 1 to the INTFAULT1 bit in the **PWMISC** register has no effect. |
| 16 | INTFAULT0 | RO | 0 | Interrupt Fault PWM 0<br><br>Value Description<br>0 The fault condition for PWM generator 0 has not been asserted.<br>1 The fault condition for PWM generator 0 is asserted.<br><br>**Note:** If the LATCH bit is set in the **PWM0CTL** register, the INTFAULT0 bit in this register can be cleared by writing a 1 to the INTFAULT0 bit in the **PWMISC** register. If the LATCH bit is 0 in the **PWM0CTL** register, writing a 1 to the INTFAULT0 bit in the **PWMISC** register has no effect. |
| 15:4 | reserved | RO | 0x000 | 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. |
| 3 | INTPWM3 | RO | 0 | PWM3 Interrupt Asserted<br><br>Value Description<br>0 The PWM generator 3 block interrupt has not been asserted.<br>1 The PWM generator 3 block interrupt is asserted.<br><br>The **PWM3RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM3ISC** register. |
| 2 | INTPWM2 | RO | 0 | PWM2 Interrupt Asserted<br><br>Value Description<br>0 The PWM generator 2 block interrupt has not been asserted.<br>1 The PWM generator 2 block interrupt is asserted.<br><br>The **PWM2RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM2ISC** register. |
| 1 | INTPWM1 | RO | 0 | PWM1 Interrupt Asserted<br><br>Value Description<br>0 The PWM generator 1 block interrupt has not been asserted.<br>1 The PWM generator 1 block interrupt is asserted.<br><br>The **PWM1RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM1ISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTPWM0 | RO | 0 | PWM0 Interrupt Asserted |

| Value | Description |
|---|---|
| 0 | The PWM generator 0 block interrupt has not been asserted. |
| 1 | The PWM generator 0 block interrupt is asserted. |

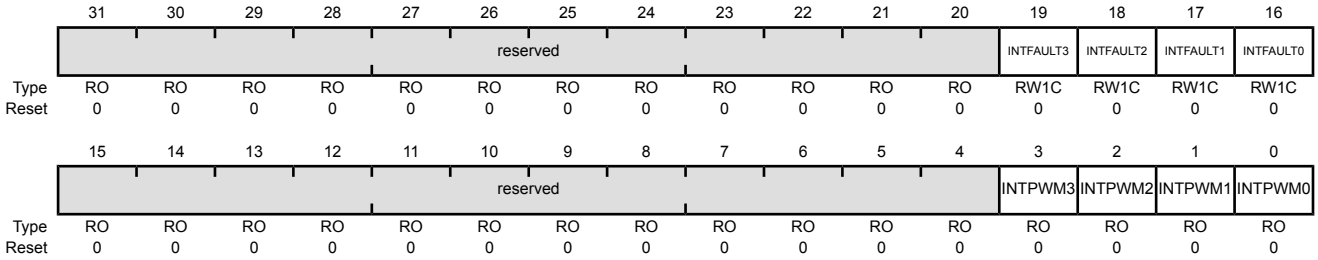
The **PWM0RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM0ISC** register.

## Register 8: PWM Interrupt Status and Clear (PWMISC), offset 0x01C

This register provides a summary of the interrupt status of the individual PWM generator blocks. If a fault interrupt is set, the corresponding MnFAULTn input has caused an interrupt. For the fault interrupt, a write of 1 to that bit position clears the latched interrupt status. If an block interrupt bit is set, the corresponding generator block is asserting an interrupt. The individual interrupt status registers, **PWMnISC**, in each block must be consulted to determine the reason for the interrupt and used to clear the interrupt.

PWM Interrupt Status and Clear (PWMISC)

PWM0 base: 0x4002.8000
Offset 0x01C
Type RW1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x000 | 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. |
| 19 | INTFAULT3 | RW1C | 0 | FAULT3 Interrupt Asserted<br><br>Value Description<br>0 The fault condition for PWM generator 3 has not been asserted or is not enabled.<br>1 An enabled interrupt for the fault condition for PWM generator 3 is asserted or is latched.<br><br>Writing a 1 to this bit clears it and the INTFAULT3 bit in the **PWMRIS** register. |
| 18 | INTFAULT2 | RW1C | 0 | FAULT2 Interrupt Asserted<br><br>Value Description<br>0 The fault condition for PWM generator 2 has not been asserted or is not enabled.<br>1 An enabled interrupt for the fault condition for PWM generator 2 is asserted or is latched.<br><br>Writing a 1 to this bit clears it and the INTFAULT2 bit in the **PWMRIS** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | INTFAULT1 | RW1C | 0 | FAULT1 Interrupt Asserted<br><br>Value Description<br>0 The fault condition for PWM generator 1 has not been asserted or is not enabled.<br>1 An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.<br><br>Writing a 1 to this bit clears it and the `INTFAULT1` bit in the **PWMRIS** register. |
| 16 | INTFAULT0 | RW1C | 0 | FAULT0 Interrupt Asserted<br><br>Value Description<br>0 The fault condition for PWM generator 0 has not been asserted or is not enabled.<br>1 An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.<br><br>Writing a 1 to this bit clears it and the `INTFAULT0` bit in the **PWMRIS** register. |
| 15:4 | reserved | RO | 0x000 | 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. |
| 3 | INTPWM3 | RO | 0 | PWM3 Interrupt Status<br><br>Value Description<br>0 The PWM generator 3 block interrupt is not asserted or is not enabled.<br>1 An enabled interrupt for the PWM generator 3 block is asserted.<br><br>The **PWM3RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM3ISC** register. |
| 2 | INTPWM2 | RO | 0 | PWM2 Interrupt Status<br><br>Value Description<br>0 The PWM generator 2 block interrupt is not asserted or is not enabled.<br>1 An enabled interrupt for the PWM generator 2 block is asserted.<br><br>The **PWM2RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM2ISC** register. |
| 1 | INTPWM1 | RO | 0 | PWM1 Interrupt Status<br><br>Value Description<br>0 The PWM generator 1 block interrupt is not asserted or is not enabled.<br>1 An enabled interrupt for the PWM generator 1 block is asserted.<br><br>The **PWM1RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM1ISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTPWM0 | RO | 0 | PWM0 Interrupt Status |

| Value | Description |
|---|---|
| 0 | The PWM generator 0 block interrupt is not asserted or is not enabled. |
| 1 | An enabled interrupt for the PWM generator 0 block is asserted. |

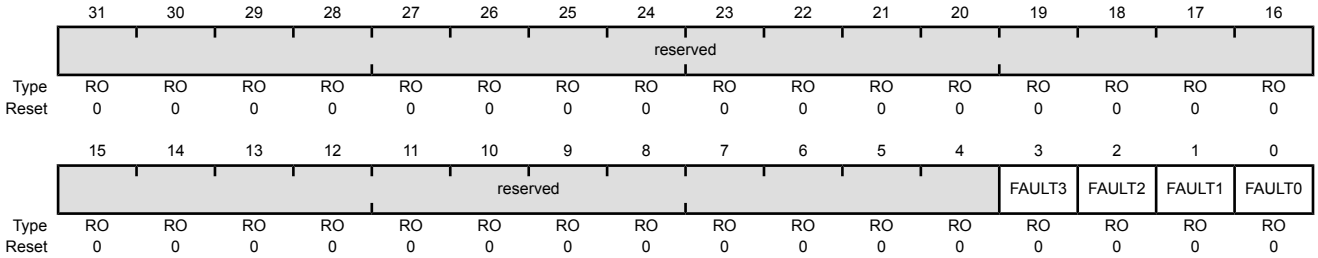
The **PWM0RIS** register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the **PWM0ISC** register.

## Register 9: PWM Status (PWMSTATUS), offset 0x020

This register provides the unlatched status of the PWM generator fault condition.

PWM Status (PWMSTATUS)

PWM0 base: 0x4002.8000
Offset 0x020
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | FAULT3 | RO | 0 | Generator 3 Fault Status<br><br>Value Description<br>0 The fault condition for PWM generator 3 is not asserted.<br>1 The fault condition for PWM generator 3 is asserted.<br>If the FLTSRC bit in the **PWM3CTL** register is clear, the input is the source of the fault condition, and is therefore asserted. |
| 2 | FAULT2 | RO | 0 | Generator 2 Fault Status<br><br>Value Description<br>0 The fault condition for PWM generator 2 is not asserted.<br>1 The fault condition for PWM generator 2 is asserted.<br>If the FLTSRC bit in the **PWM2CTL** register is clear, the input is the source of the fault condition, and is therefore asserted. |
| 1 | FAULT1 | RO | 0 | Generator 1 Fault Status<br><br>Value Description<br>0 The fault condition for PWM generator 1 is not asserted.<br>1 The fault condition for PWM generator 1 is asserted.<br>If the FLTSRC bit in the **PWM1CTL** register is clear, the input is the source of the fault condition, and is therefore asserted. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | FAULT0 | RO | 0 | Generator 0 Fault Status |

| Value | Description |
|---|---|
| 0 | The fault condition for PWM generator 0 is not asserted. |
| 1 | The fault condition for PWM generator 0 is asserted.<br>If the FLTSRC bit in the **PWM0CTL** register is clear, the input is the source of the fault condition, and is therefore asserted. |

## Register 10: PWM Fault Condition Value (PWMFAULTVAL), offset 0x024

This register specifies the output value driven on the MnPWMn signals during a fault condition if enabled by the corresponding bit in the **PWMFAULT** register. Note that if the corresponding bit in the **PWMINVERT** register is set, the output value is driven to the logical NOT of the bit value in this register.

PWM Fault Condition Value (PWMFAULTVAL)

PWM0 base: 0x4002.8000
Offset 0x024
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | PWM7 | PWM6 | PWM5 | PWM4 | PWM3 | PWM2 | PWM1 | PWM0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | PWM7 | RW | 0 | MnPWM7 Fault Value<br>Value Description<br>0 The MnPWM7 output signal is driven Low during fault conditions if the FAULT7 bit in the **PWMFAULT** register is set.<br>1 The MnPWM7 output signal is driven High during fault conditions if the FAULT7 bit in the **PWMFAULT** register is set. |
| 6 | PWM6 | RW | 0 | MnPWM6 Fault Value<br>Value Description<br>0 The MnPWM6 output signal is driven Low during fault conditions if the FAULT6 bit in the **PWMFAULT** register is set.<br>1 The MnPWM6 output signal is driven High during fault conditions if the FAULT6 bit in the **PWMFAULT** register is set. |
| 5 | PWM5 | RW | 0 | MnPWM5 Fault Value<br>Value Description<br>0 The MnPWM5 output signal is driven Low during fault conditions if the FAULT5 bit in the **PWMFAULT** register is set.<br>1 The MnPWM5 output signal is driven High during fault conditions if the FAULT5 bit in the **PWMFAULT** register is set. |

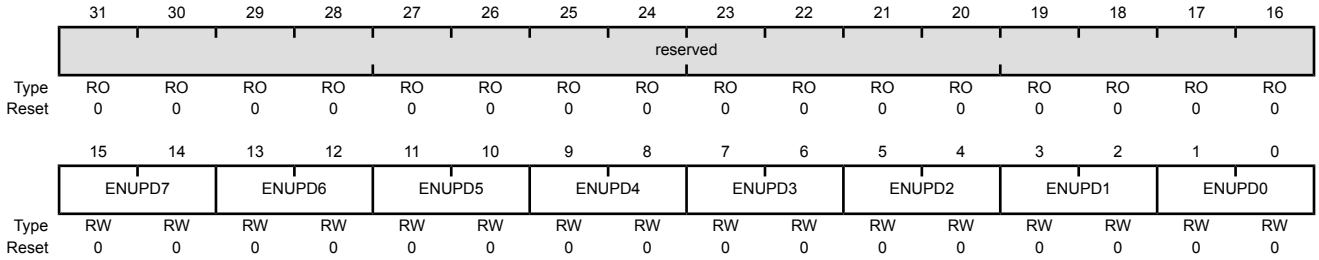
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | PWM4 | RW | 0 | MnPWM4 Fault Value<br><br>Value Description<br>0 The MnPWM4 output signal is driven Low during fault conditions if the FAULT4 bit in the **PWMFAULT** register is set.<br>1 The MnPWM4 output signal is driven High during fault conditions if the FAULT4 bit in the **PWMFAULT** register is set. |
| 3 | PWM3 | RW | 0 | MnPWM3 Fault Value<br><br>Value Description<br>0 The MnPWM3 output signal is driven Low during fault conditions if the FAULT3 bit in the **PWMFAULT** register is set.<br>1 The MnPWM3 output signal is driven High during fault conditions if the FAULT3 bit in the **PWMFAULT** register is set. |
| 2 | PWM2 | RW | 0 | MnPWM2 Fault Value<br><br>Value Description<br>0 The MnPWM2 output signal is driven Low during fault conditions if the FAULT2 bit in the **PWMFAULT** register is set.<br>1 The MnPWM2 output signal is driven High during fault conditions if the FAULT2 bit in the **PWMFAULT** register is set. |
| 1 | PWM1 | RW | 0 | MnPWM1 Fault Value<br><br>Value Description<br>0 The MnPWM1 output signal is driven Low during fault conditions if the FAULT1 bit in the **PWMFAULT** register is set.<br>1 The MnPWM1 output signal is driven High during fault conditions if the FAULT1 bit in the **PWMFAULT** register is set. |
| 0 | PWM0 | RW | 0 | MnPWM0 Fault Value<br><br>Value Description<br>0 The MnPWM0 output signal is driven Low during fault conditions if the FAULT0 bit in the **PWMFAULT** register is set.<br>1 The MnPWM0 output signal is driven High during fault conditions if the FAULT0 bit in the **PWMFAULT** register is set. |

## Register 11: PWM Enable Update (PWMENUPD), offset 0x028

This register specifies when updates to the PWMnEN bit in the **PWMENABLE** register are performed. The PWMnEN bit enables the pwmA' or pwmB' output to be passed to the microcontroller's pin. Updates can be immediate or locally or globally synchronized to the next synchronous update.

PWM Enable Update (PWMENUPD)

PWM0 base: 0x4002.8000
Offset 0x028
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x00 | 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. |
| 15:14 | ENUPD7 | RW | 0 | MnPWM7 Enable Update Mode |

| Value | Description |
|---|---|
| 0x0 | Immediate<br>Writes to the PWM7EN bit in the **PWMENABLE** register are used by the PWM generator immediately. |
| 0x1 | Reserved |
| 0x2 | Locally Synchronized<br>Writes to the PWM7EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0. |
| 0x3 | Globally Synchronized<br>Writes to the PWM7EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13:12 | ENUPD6 | RW | 0 | MnPWM6 Enable Update Mode<br><br>Value Description<br>0x0 Immediate<br>Writes to the PWM6EN bit in the **PWMENABLE** register are used by the PWM generator immediately.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Writes to the PWM6EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Writes to the PWM6EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |
| 11:10 | ENUPD5 | RW | 0 | MnPWM5 Enable Update Mode<br><br>Value Description<br>0x0 Immediate<br>Writes to the PWM5EN bit in the **PWMENABLE** register are used by the PWM generator immediately.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Writes to the PWM5EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Writes to the PWM5EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |
| 9:8 | ENUPD4 | RW | 0 | MnPWM4 Enable Update Mode<br><br>Value Description<br>0x0 Immediate<br>Writes to the PWM4EN bit in the **PWMENABLE** register are used by the PWM generator immediately.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Writes to the PWM4EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Writes to the PWM4EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | ENUPD3 | RW | 0 | MnPWM3 Enable Update Mode<br><br>Value Description<br>0x0 Immediate<br>Writes to the PWM3EN bit in the **PWMENABLE** register are used by the PWM generator immediately.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Writes to the PWM3EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Writes to the PWM3EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |
| 5:4 | ENUPD2 | RW | 0 | MnPWM2 Enable Update Mode<br><br>Value Description<br>0x0 Immediate<br>Writes to the PWM2EN bit in the **PWMENABLE** register are used by the PWM generator immediately.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Writes to the PWM2EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Writes to the PWM2EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |
| 3:2 | ENUPD1 | RW | 0 | MnPWM1 Enable Update Mode<br><br>Value Description<br>0x0 Immediate<br>Writes to the PWM1EN bit in the **PWMENABLE** register are used by the PWM generator immediately.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Writes to the PWM1EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Writes to the PWM1EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1:0 | ENUPD0 | RW | 0 | MnPWM0 Enable Update Mode |

| Value | Description |
|---|---|
| 0x0 | Immediate<br>Writes to the PWM0EN bit in the **PWMENABLE** register are used by the PWM generator immediately. |
| 0x1 | Reserved |
| 0x2 | Locally Synchronized<br>Writes to the PWM0EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0. |
| 0x3 | Globally Synchronized<br>Writes to the PWM0EN bit in the **PWMENABLE** register are used by the PWM generator the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register. |

### Register 12: PWM0 Control (PWM0CTL), offset 0x040

### Register 13: PWM1 Control (PWM1CTL), offset 0x080

### Register 14: PWM2 Control (PWM2CTL), offset 0x0C0

### Register 15: PWM3 Control (PWM3CTL), offset 0x100

These registers configure the PWM signal generation blocks (PWM0CTL controls the PWM generator 0 block, and so on). The Register Update mode, Debug mode, Counting mode, and Block Enable mode are all controlled via these registers. The blocks produce the PWM signals, which can be either two independent PWM signals (from the same counter), or a paired set of PWM signals with dead-band delays added.

The PWM0 block produces the MnPWM0 and MnPWM1 outputs, the PWM1 block produces the MnPWM2 and MnPWM3 outputs, the PWM2 block produces the MnPWM4 and MnPWM5 outputs, and the PWM3 block produces the MnPWM6 and MnPWM7 outputs.

PWMn Control (PWMnCTL)

PWM0 base: 0x4002.8000
Offset 0x040
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | LATCH | MINFLTPER | FLTSRC |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DBFALLUPD | | DBRISEUPD | | DBCTLUPD | | GENBUPD | | GENAUPD | | CMPBUPD | CMPAUPD | LOADUPD | DEBUG | MODE | ENABLE |
| Type | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x000 | 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. |
| 18 | LATCH | RW | 0 | Latch Fault Input |

| Value | Description |
|---|---|
| 0 | Fault Condition Not Latched<br>A fault condition is in effect for as long as the generating source is asserting. |
| 1 | Fault Condition Latched<br>A fault condition is set as the result of the assertion of the faulting source and is held (latched) while the **PWMISC** INTFAULTn bit is set. Clearing the INTFAULTn bit clears the fault condition. |

**Note:** When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the **PWMnCTL** register should be set to 1 to ensure trigger assertions are captured.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 17 | MINFLTPER | RW | 0 | Minimum Fault Period<br>This bit specifies that the PWM generator enables a one-shot counter to provide a minimum fault condition period.<br>The timer begins counting on the rising edge of the fault condition to extend the condition for a minimum duration of the count value. The timer ignores the state of the fault condition while counting.<br>The minimum fault delay is in effect only when the MINFLTPER bit is set. If a detected fault is in the process of being extended when the MINFLTPER bit is cleared, the fault condition extension is aborted.<br>The delay time is specified by the **PWMnMINFLTPER** register MFP field value. The effect of this is to pulse stretch the fault condition input.<br>The delay value is defined by the PWM clock period. Because the fault input is not synchronized to the PWM clock, the period of the time is PWMClock * (MFP value + 1) or PWMClock * (MFP value + 2).<br>The delay function makes sense only if the fault source is unlatched. A latched fault source makes the fault condition appear asserted until cleared by software and negates the utility of the extend feature. It applies to all fault condition sources as specified in the FLTSRC field.<br>Value Description<br>0 The FAULT input deassertion is unaffected.<br>1 The **PWMnMINFLTPER** one-shot counter is active and extends the period of the fault condition to a minimum period.<br>**Note:** When using an ADC digital comparator as a fault source, the LATCH and MINFLTPER bits in the **PWMnCTL** register should be set to 1 to ensure trigger assertions are captured. |
| 16 | FLTSRC | RW | 0 | Fault Condition Source<br>Value Description<br>0 The Fault condition is determined by the Fault0 input.<br>1 The Fault condition is determined by the configuration of the **PWMnFLTSRC0** and **PWMnFLTSRC1** registers. |
| 15:14 | DBFALLUPD | RW | 0x0 | **PWMnDBFALL** Update Mode<br>Value Description<br>0x0 Immediate<br>The **PWMnDBFALL** register value is immediately updated on a write.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Updates to the register are reflected to the generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13:12 | DBRISEUPD | RW | 0x0 | **PWMnDBRISE** Update Mode<br><br>Value Description<br>0x0 Immediate<br>The **PWMnDBRISE** register value is immediately updated on a write.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Updates to the register are reflected to the generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |
| 11:10 | DBCTLUPD | RW | 0x0 | **PWMnDBCTL** Update Mode<br><br>Value Description<br>0x0 Immediate<br>The **PWMnDBCTL** register value is immediately updated on a write.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Updates to the register are reflected to the generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |
| 9:8 | GENBUPD | RW | 0x0 | **PWMnGENB** Update Mode<br><br>Value Description<br>0x0 Immediate<br>The **PWMnGENB** register value is immediately updated on a write.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Updates to the register are reflected to the generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:6 | GENAUPD | RW | 0x0 | **PWMnGENA** Update Mode<br><br>Value Description<br>0x0 Immediate<br>The **PWMnGENA** register value is immediately updated on a write.<br>0x1 Reserved<br>0x2 Locally Synchronized<br>Updates to the register are reflected to the generator the next time the counter is 0.<br>0x3 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |
| 5 | CMPBUPD | RW | 0 | Comparator B Update Mode<br><br>Value Description<br>0 Locally Synchronized<br>Updates to the **PWMnCMPB** register are reflected to the generator the next time the counter is 0.<br>1 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |
| 4 | CMPAUPD | RW | 0 | Comparator A Update Mode<br><br>Value Description<br>0 Locally Synchronized<br>Updates to the **PWMnCMPA** register are reflected to the generator the next time the counter is 0.<br>1 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |
| 3 | LOADUPD | RW | 0 | Load Register Update Mode<br><br>Value Description<br>0 Locally Synchronized<br>Updates to the **PWMnLOAD** register are reflected to the generator the next time the counter is 0.<br>1 Globally Synchronized<br>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWMCTL** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | DEBUG | RW | 0 | Debug Mode<br><br>Value Description<br>0 The counter stops running when it next reaches 0 and continues running again when no longer in Debug mode.<br>1 The counter always runs when in Debug mode. |
| 1 | MODE | RW | 0 | Counter Mode<br><br>Value Description<br>0 The counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode).<br>1 The counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode). |
| 0 | ENABLE | RW | 0 | PWM Block Enable<br><br>**Note:** Disabling the PWM by clearing the ENABLE bit does not clear the COUNT field of the **PWMnCOUNT** register. Before re-enabling the PWM (ENABLE = 0x1), the COUNT field should be cleared by resetting the PWM registers through the **SRPWM** register in the System Control Module.<br><br>Value Description<br>0 The entire PWM generation block is disabled and not clocked.<br>1 The PWM generation block is enabled and produces PWM signals. |

### Register 16: PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044

### Register 17: PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084

### Register 18: PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4

### Register 19: PWM3 Interrupt and Trigger Enable (PWM3INTEN), offset 0x104

These registers control the interrupt and ADC trigger generation capabilities of the PWM generators (**PWM0INTEN** controls the PWM generator 0 block, and so on). The events that can cause an interrupt,or an ADC trigger are:

- The counter being equal to the load register
- The counter being equal to zero
- The counter being equal to the **PWMnCMPA** register while counting up
- The counter being equal to the **PWMnCMPA** register while counting down
- The counter being equal to the **PWMnCMPB** register while counting up
- The counter being equal to the **PWMnCMPB** register while counting down

Any combination of these events can generate either an interrupt or an ADC trigger, though no determination can be made as to the actual event that caused an ADC trigger if more than one is specified. The **PWMnRIS** register provides information about which events have caused raw interrupts.

PWMn Interrupt and Trigger Enable (PWMnINTEN)

PWM0 base: 0x4002.8000
Offset 0x044
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | TRCMPBD | TRCMPBU | TRCMPAD | TRCMPAU | TRCNTLOAD | TRCNTZERO | reserved | | INTCMPBD | INTCMPBU | INTCMPAD | INTCMPAU | INTCNTLOAD | INTCNTZERO |
| RO | RO | RW | RW | RW | RW | RW | RW | RO | RO | RW | RW | RW | RW | RW | RW |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0 | 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. |
| 13 | TRCMPBD | RW | 0 | Trigger for Counter=**PWMnCMPB** Down<br>Value Description<br>0 No ADC trigger is output.<br>1 An ADC trigger pulse is output when the counter matches the value in the **PWMnCMPB** register value while counting down. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | TRCMPBU | RW | 0 | Trigger for Counter=**PWMnCMPB** Up<br><br>Value Description<br>0 No ADC trigger is output.<br>1 An ADC trigger pulse is output when the counter matches the value in the **PWMnCMPB** register value while counting up. |
| 11 | TRCMPAD | RW | 0 | Trigger for Counter=**PWMnCMPA** Down<br><br>Value Description<br>0 No ADC trigger is output.<br>1 An ADC trigger pulse is output when the counter matches the value in the **PWMnCMPA** register value while counting down. |
| 10 | TRCMPAU | RW | 0 | Trigger for Counter=**PWMnCMPA** Up<br><br>Value Description<br>0 No ADC trigger is output.<br>1 An ADC trigger pulse is output when the counter matches the value in the **PWMnCMPA** register value while counting up. |
| 9 | TRCNTLOAD | RW | 0 | Trigger for Counter=**PWMnLOAD**<br><br>Value Description<br>0 No ADC trigger is output.<br>1 An ADC trigger pulse is output when the counter matches the **PWMnLOAD** register. |
| 8 | TRCNTZERO | RW | 0 | Trigger for Counter=0<br><br>Value Description<br>0 No ADC trigger is output.<br>1 An ADC trigger pulse is output when the counter is 0. |
| 7:6 | reserved | RO | 0x0 | 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 | INTCMPBD | RW | 0 | Interrupt for Counter=**PWMnCMPB** Down<br><br>Value Description<br>0 No interrupt.<br>1 A raw interrupt occurs when the counter matches the value in the **PWMnCMPB** register value while counting down. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | INTCMPBU | RW | 0 | Interrupt for Counter=**PWMnCMPB** Up<br><br>Value Description<br>0 No interrupt.<br>1 A raw interrupt occurs when the counter matches the value in the **PWMnCMPB** register value while counting up. |
| 3 | INTCMPAD | RW | 0 | Interrupt for Counter=**PWMnCMPA** Down<br><br>Value Description<br>0 No interrupt.<br>1 A raw interrupt occurs when the counter matches the value in the **PWMnCMPA** register value while counting down. |
| 2 | INTCMPAU | RW | 0 | Interrupt for Counter=**PWMnCMPA** Up<br><br>Value Description<br>0 No interrupt.<br>1 A raw interrupt occurs when the counter matches the value in the **PWMnCMPA** register value while counting up. |
| 1 | INTCNTLOAD | RW | 0 | Interrupt for Counter=**PWMnLOAD**<br><br>Value Description<br>0 No interrupt.<br>1 A raw interrupt occurs when the counter matches the value in the **PWMnLOAD** register value. |
| 0 | INTCNTZERO | RW | 0 | Interrupt for Counter=0<br><br>Value Description<br>0 No interrupt.<br>1 A raw interrupt occurs when the counter is zero. |

## Register 20: PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048

## Register 21: PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088

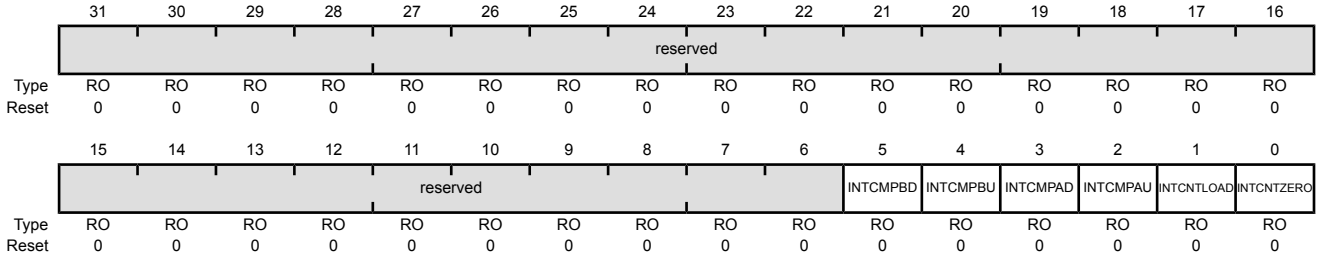
## Register 22: PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8

## Register 23: PWM3 Raw Interrupt Status (PWM3RIS), offset 0x108

These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (**PWM0RIS** controls the PWM generator 0 block, and so on). If a bit is set, the event has occurred; if a bit is clear, the event has not occurred. Bits in this register are cleared by writing a 1 to the corresponding bit in the **PWMnISC** register.

PWMn Raw Interrupt Status (PWMnRIS)

PWM0 base: 0x4002.8000
Offset 0x048
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0 | 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 | INTCMPBD | RO | 0 | Comparator B Down Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 The counter has matched the value in the **PWMnCMPB** register while counting down.<br><br>This bit is cleared by writing a 1 to the `INTCMPBD` bit in the **PWMnISC** register. |
| 4 | INTCMPBU | RO | 0 | Comparator B Up Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 The counter has matched the value in the **PWMnCMPB** register while counting up.<br><br>This bit is cleared by writing a 1 to the `INTCMPBU` bit in the **PWMnISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INTCMPAD | RO | 0 | Comparator A Down Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 The counter has matched the value in the **PWMnCMPA** register while counting down.<br><br>This bit is cleared by writing a 1 to the `INTCMPAD` bit in the **PWMnISC** register. |
| 2 | INTCMPAU | RO | 0 | Comparator A Up Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 The counter has matched the value in the **PWMnCMPA** register while counting up.<br><br>This bit is cleared by writing a 1 to the `INTCMPAU` bit in the **PWMnISC** register. |
| 1 | INTCNTLOAD | RO | 0 | Counter=Load Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 The counter has matched the value in the **PWMnLOAD** register.<br><br>This bit is cleared by writing a 1 to the `INTCNTLOAD` bit in the **PWMnISC** register. |
| 0 | INTCNTZERO | RO | 0 | Counter=0 Interrupt Status<br><br>Value Description<br>0 An interrupt has not occurred.<br>1 The counter has matched zero.<br><br>This bit is cleared by writing a 1 to the `INTCNTZERO` bit in the **PWMnISC** register. |

## Register 24: PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C

## Register 25: PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C

## Register 26: PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC

## Register 27: PWM3 Interrupt Status and Clear (PWM3ISC), offset 0x10C

These registers provide the current set of interrupt sources that are asserted to the interrupt controller (**PWM0ISC** controls the PWM generator 0 block, and so on). A bit is set if the event has occurred and is enabled in the **PWMnINTEN** register; if a bit is clear, the event has not occurred or is not enabled. These are RW1C registers; writing a 1 to a bit position clears the corresponding interrupt reason.

**Note:** The interrupt status can only be cleared one PWM Clock cycle after the interrupt occurs. The larger the PWM Clock Divider (PWMDIV) value in **PWMCC** register, the longer the system delay is to clear the interrupt.

PWMn Interrupt Status and Clear (PWMnISC)

PWM0 base: 0x4002.8000
Offset 0x04C
Type RW1C, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | INTCMPBD | INTCMPBU | INTCMPAD | INTCMPAU | INTCNTLOAD | INTCNTZERO |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0 | 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 | INTCMPBD | RW1C | 0 | Comparator B Down Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The INTCMPBD bits in the **PWMnRIS** and **PWMnINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBD bit in the **PWMnRIS** register. |
| 4 | INTCMPBU | RW1C | 0 | Comparator B Up Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The INTCMPBU bits in the **PWMnRIS** and **PWMnINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBU bit in the **PWMnRIS** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | INTCMPAD | RW1C | 0 | Comparator A Down Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The INTCMPAD bits in the **PWMnRIS** and **PWMnINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAD bit in the **PWMnRIS** register. |
| 2 | INTCMPAU | RW1C | 0 | Comparator A Up Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The INTCMPAU bits in the **PWMnRIS** and **PWMnINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAU bit in the **PWMnRIS** register. |
| 1 | INTCNTLOAD | RW1C | 0 | Counter=Load Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The INTCNTLOAD bits in the **PWMnRIS** and **PWMnINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTLOAD bit in the **PWMnRIS** register. |
| 0 | INTCNTZERO | RW1C | 0 | Counter=0 Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The INTCNTZERO bits in the **PWMnRIS** and **PWMnINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTZERO bit in the **PWMnRIS** register. |

### Register 28: PWM0 Load (PWM0LOAD), offset 0x050

### Register 29: PWM1 Load (PWM1LOAD), offset 0x090

### Register 30: PWM2 Load (PWM2LOAD), offset 0x0D0

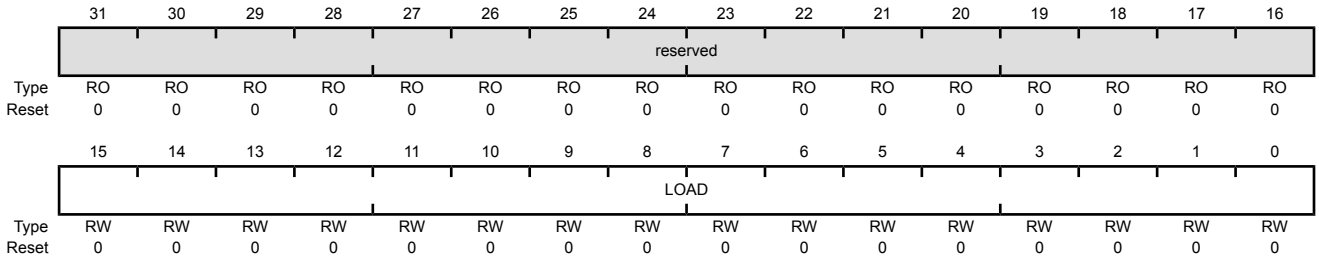
### Register 31: PWM3 Load (PWM3LOAD), offset 0x110

These registers contain the load value for the PWM counter (**PWM0LOAD** controls the PWM generator 0 block, and so on). Based on the counter mode configured by the MODE bit in the **PWMnCTL** register, this value is either loaded into the counter after it reaches zero or is the limit of up-counting after which the counter decrements back to zero. When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and/or pwmB signal (via the **PWMnGENA**/**PWMnGENB** register) or drive an interruptor ADC trigger (via the **PWMnINTEN** register).

If the Load Value Update mode is locally synchronized (based on the LOADUPD field encoding in the **PWMnCTL** register), the 16-bit LOAD value is used the next time the counter reaches zero. If the update mode is globally synchronized, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is re-written before the actual update occurs, the previous value is never used and is lost.

PWMn Load (PWMnLOAD)

PWM0 base: 0x4002.8000
Offset 0x050
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | LOAD | RW | 0x0000 | Counter Load Value<br>The counter load value. |

### Register 32: PWM0 Counter (PWM0COUNT), offset 0x054

### Register 33: PWM1 Counter (PWM1COUNT), offset 0x094

### Register 34: PWM2 Counter (PWM2COUNT), offset 0x0D4

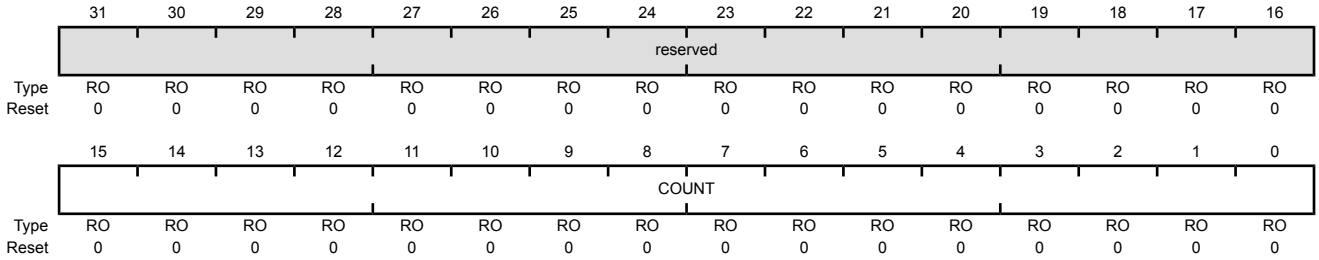
### Register 35: PWM3 Counter (PWM3COUNT), offset 0x114

These registers contain the current value of the PWM counter (**PWM0COUNT** is the value of the PWM generator 0 block, and so on). When this value matches zero or the value in the **PWMnLOAD**, **PWMnCMPA**, or **PWMnCMPB** registers, a pulse is output which can be configured to drive the generation of a PWM signal or drive an interrupt or ADC trigger.

**Note:** Disabling the PWM by clearing the ENABLE bit does not clear the COUNT field of the **PWMnCOUNT** register. Before re-enabling the PWM (ENABLE = 0x1), the COUNT field should be cleared by resetting the PWM registers through the **SRPWM** register in the System Control Module.

PWMn Counter (PWMnCOUNT)

PWM0 base: 0x4002.8000
Offset 0x054
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | COUNT | RO | 0x0000 | Counter Value<br>The current value of the counter. |

### Register 36: PWM0 Compare A (PWM0CMPA), offset 0x058

### Register 37: PWM1 Compare A (PWM1CMPA), offset 0x098

### Register 38: PWM2 Compare A (PWM2CMPA), offset 0x0D8

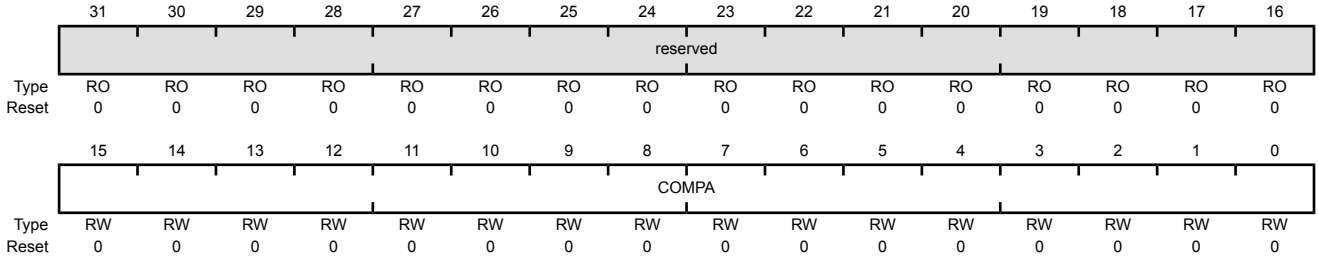
### Register 39: PWM3 Compare A (PWM3CMPA), offset 0x118

These registers contain a value to be compared against the counter (**PWM0CMPA** controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and pwmB signals (via the **PWMnGENA** and **PWMnGENB** registers) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register). If the value of this register is greater than the **PWMnLOAD** register (see page 1976), then no pulse is ever output.

If the comparator A update mode is locally synchronized (based on the CMPAUPD bit in the **PWMnCTL** register), the 16-bit COMPA value is used the next time the counter reaches zero. If the update mode is globally synchronized, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMn Compare A (PWMnCMPA)

PWM0 base: 0x4002.8000
Offset 0x058
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x00 | 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. |
| 15:0 | COMPA | RW | 0x00 | Comparator A Value<br>The value to be compared against the counter. |

## Register 40: PWM0 Compare B (PWM0CMPB), offset 0x05C

## Register 41: PWM1 Compare B (PWM1CMPB), offset 0x09C

## Register 42: PWM2 Compare B (PWM2CMPB), offset 0x0DC

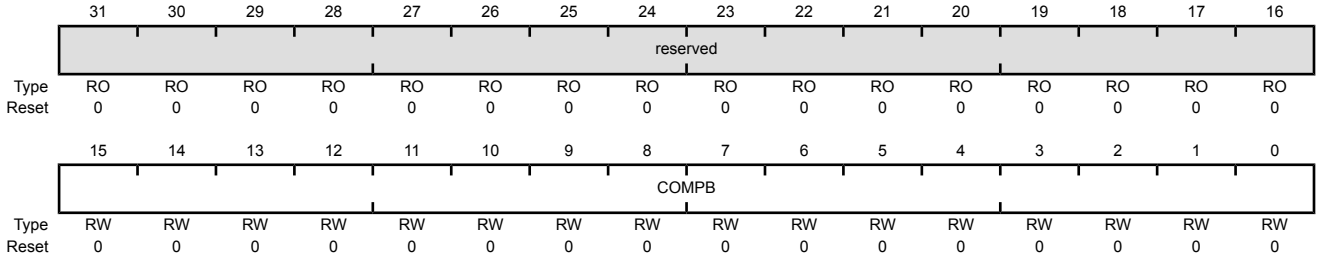
## Register 43: PWM3 Compare B (PWM3CMPB), offset 0x11C

These registers contain a value to be compared against the counter (**PWM0CMPB** controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and pwmB signals (via the **PWMnGENA** and **PWMnGENB** registers) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register). If the value of this register is greater than the **PWMnLOAD** register, no pulse is ever output.

If the comparator B update mode is locally synchronized (based on the CMPBUPD bit in the **PWMnCTL** register), the 16-bit COMPB value is used the next time the counter reaches zero. If the update mode is globally synchronized, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMn Compare B (PWMnCMPB)

PWM0 base: 0x4002.8000
Offset 0x05C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | COMPB | RW | 0x0000 | Comparator B Value<br>The value to be compared against the counter. |

## Register 44: PWM0 Generator A Control (PWM0GENA), offset 0x060

## Register 45: PWM1 Generator A Control (PWM1GENA), offset 0x0A0

## Register 46: PWM2 Generator A Control (PWM2GENA), offset 0x0E0

## Register 47: PWM3 Generator A Control (PWM3GENA), offset 0x120

These registers control the generation of the pwmA signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (**PWM0GENA** controls the PWM generator 0 block, and so on). When the counter is running in Count-Down mode, only four of these events occur; when running in Count-Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the resulting PWM signal.

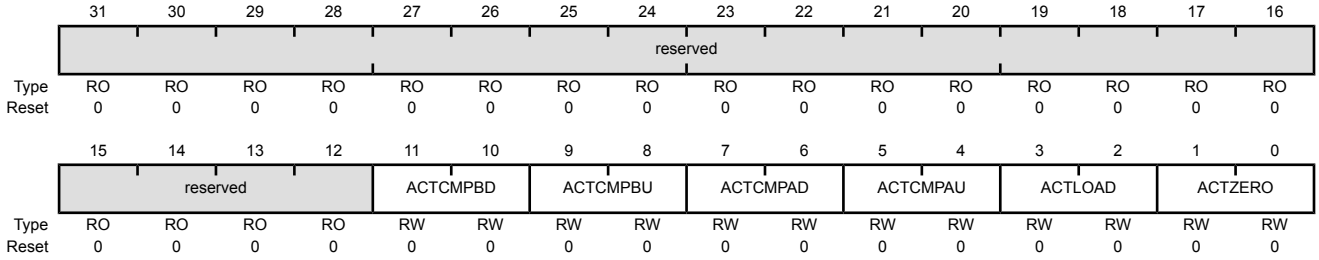
The **PWM0GENA** register controls generation of the pwm0A signal; **PWM1GENA**, the pwm1A signal; **PWM2GENA**, the pwm2A signal; and **PWM3GENA**, the pwm3A signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare A action is taken and the compare B action is ignored.

If the Generator A update mode is immediate (based on the GENAUPD field encoding in the **PWMnCTL** register), the ACTCMPBD, ACTCMPBU, ACTCMPAD, ACTCMPAU, ACTLOAD, and ACTZERO values are used immediately. If the update mode is locally synchronized, these values are used the next time the counter reaches zero. If the update mode is globally synchronized, these values are used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMn Generator A Control (PWMnGENA)

PWM0 base: 0x4002.8000
Offset 0x060
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:10 | ACTCMPBD | RW | 0x0 | Action for Comparator B Down<br>This field specifies the action to be taken when the counter matches comparator B while counting down.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmA.<br>0x2 Drive pwmA Low.<br>0x3 Drive pwmA High. |
| 9:8 | ACTCMPBU | RW | 0x0 | Action for Comparator B Up<br>This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the **PWMnCTL** register is set.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmA.<br>0x2 Drive pwmA Low.<br>0x3 Drive pwmA High. |
| 7:6 | ACTCMPAD | RW | 0x0 | Action for Comparator A Down<br>This field specifies the action to be taken when the counter matches comparator A while counting down.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmA.<br>0x2 Drive pwmA Low.<br>0x3 Drive pwmA High. |
| 5:4 | ACTCMPAU | RW | 0x0 | Action for Comparator A Up<br>This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the **PWMnCTL** register is set.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmA.<br>0x2 Drive pwmA Low.<br>0x3 Drive pwmA High. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | ACTLOAD | RW | 0x0 | Action for Counter=LOAD<br>This field specifies the action to be taken when the counter matches the value in the **PWMnLOAD** register.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmA.<br>0x2 Drive pwmA Low.<br>0x3 Drive pwmA High. |
| 1:0 | ACTZERO | RW | 0x0 | Action for Counter=0<br>This field specifies the action to be taken when the counter is zero.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmA.<br>0x2 Drive pwmA Low.<br>0x3 Drive pwmA High. |

## Register 48: PWM0 Generator B Control (PWM0GENB), offset 0x064

## Register 49: PWM1 Generator B Control (PWM1GENB), offset 0x0A4

## Register 50: PWM2 Generator B Control (PWM2GENB), offset 0x0E4

## Register 51: PWM3 Generator B Control (PWM3GENB), offset 0x124

These registers control the generation of the pwmB signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (**PWM0GENB** controls the PWM generator 0 block, and so on). When the counter is running in Count-Down mode, only four of these events occur; when running in Count-Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the resulting PWM signal.

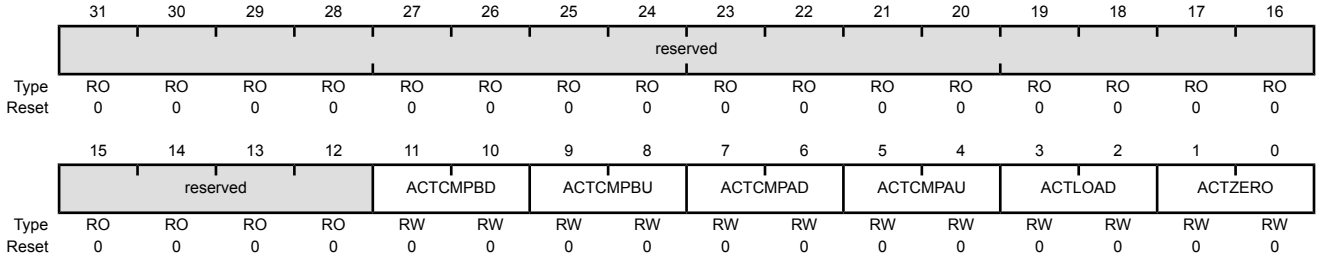
The **PWM0GENB** register controls generation of the pwm0B signal; **PWM1GENB**, the pwm1B signal; **PWM2GENB**, the pwm2B signal; and **PWM3GENB**, the pwm3B signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare B action is taken and the compare A action is ignored.

If the Generator B update mode is immediate (based on the GENBUPD field encoding in the **PWMnCTL** register), the ACTCMPBD, ACTCMPBU, ACTCMPAD, ACTCMPAU, ACTLOAD, and ACTZERO values are used immediately. If the update mode is locally synchronized, these values are used the next time the counter reaches zero. If the update mode is globally synchronized, these values are used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMn Generator B Control (PWMnGENB), offset 0x064

PWM0 base: 0x4002.8000
Offset 0x064
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:10 | ACTCMPBD | RW | 0x0 | Action for Comparator B Down<br>This field specifies the action to be taken when the counter matches comparator B while counting down.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmB.<br>0x2 Drive pwmB Low.<br>0x3 Drive pwmB High. |
| 9:8 | ACTCMPBU | RW | 0x0 | Action for Comparator B Up<br>This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the **PWMnCTL** register is set.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmB.<br>0x2 Drive pwmB Low.<br>0x3 Drive pwmB High. |
| 7:6 | ACTCMPAD | RW | 0x0 | Action for Comparator A Down<br>This field specifies the action to be taken when the counter matches comparator A while counting down.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmB.<br>0x2 Drive pwmB Low.<br>0x3 Drive pwmB High. |
| 5:4 | ACTCMPAU | RW | 0x0 | Action for Comparator A Up<br>This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the **PWMnCTL** register is set.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmB.<br>0x2 Drive pwmB Low.<br>0x3 Drive pwmB High. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | ACTLOAD | RW | 0x0 | Action for Counter=LOAD<br>This field specifies the action to be taken when the counter matches the load value.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmB.<br>0x2 Drive pwmB Low.<br>0x3 Drive pwmB High. |
| 1:0 | ACTZERO | RW | 0x0 | Action for Counter=0<br>This field specifies the action to be taken when the counter is 0.<br><br>Value Description<br>0x0 Do nothing.<br>0x1 Invert pwmB.<br>0x2 Drive pwmB Low.<br>0x3 Drive pwmB High. |

## Register 52: PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068

## Register 53: PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8

## Register 54: PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8

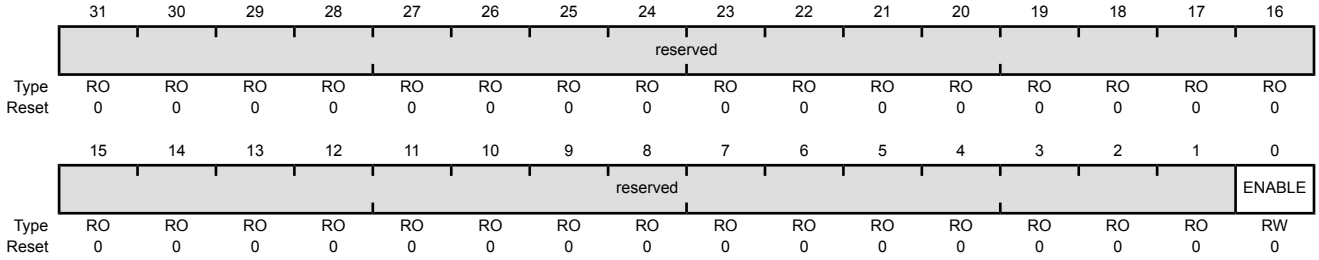
## Register 55: PWM3 Dead-Band Control (PWM3DBCTL), offset 0x128

The **PWMnDBCTL** register controls the dead-band generator, which produces the MnPWMn signals based on the pwmA and pwmB signals. When disabled, the pwmA signal passes through to the pwmA' signal and the pwmB signal passes through to the pwmB' signal. When dead-band control is enabled, the pwmB signal is ignored, the pwmA' signal is generated by delaying the rising edge(s) of the pwmA signal by the value in the **PWMnDBRISE** register (see page 1987), and the pwmB' signal is generated by inverting the pwmA signal and delaying the falling edge(s) of the pwmA signal by the value in the **PWMnDBFALL** register (see page 1988). The Output Control block outputs the pwm0A' signal on the MnPWM0 signal and the pwm0B' signal on the MnPWM1 signal. In a similar manner, MnPWM2 and MnPWM3 are produced from the pwm1A' and pwm1B' signals, MnPWM4 and MnPWM5 are produced from the pwm2A' and pwm2B' signals, and MnPWM6 and MnPWM7 are produced from the pwm3A' and pwm3B' signals.

If the Dead-Band Control mode is immediate (based on the DBCTLUPD field encoding in the **PWMnCTL** register), the ENABLE bit value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMn Dead-Band Control (PWMnDBCTL)

PWM0 base: 0x4002.8000
Offset 0x068
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | 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. |
| 0 | ENABLE | RW | 0 | Dead-Band Generator Enable |

| Value | Description |
|---|---|
| 0 | The pwmA and pwmB signals pass through to the pwmA' and pwmB' signals unmodified. |
| 1 | The dead-band generator modifies the pwmA signal by inserting dead bands into the pwmA' and pwmB' signals. |

## Register 56: PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C

## Register 57: PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC

## Register 58: PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC

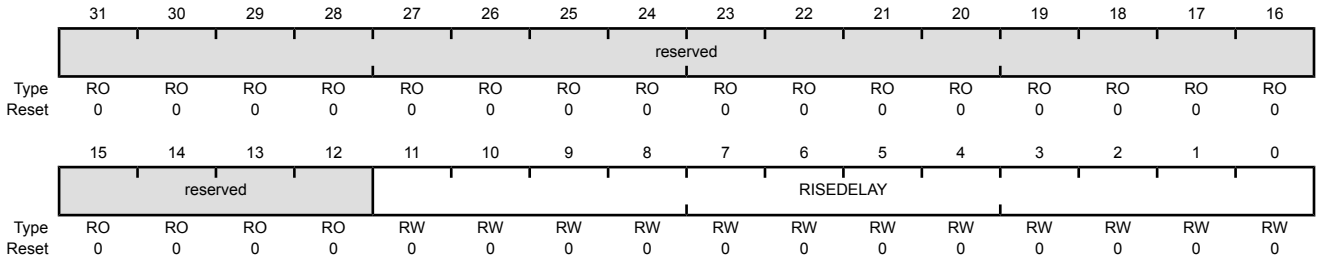
## Register 59: PWM3 Dead-Band Rising-Edge Delay (PWM3DBRISE), offset 0x12C

The **PWMnDBRISE** register contains the number of clock cycles to delay the rising edge of the pwmA signal when generating the pwmA' signal. If the dead-band generator is disabled through the **PWMnDBCTL** register, this register is ignored. If the value of this register is larger than the width of a High pulse on the pwmA signal, the rising-edge delay consumes the entire High time of the signal, resulting in no High time on the output. Care must be taken to ensure that the pwmA High time always exceeds the rising-edge delay.

If the Dead-Band Rising-Edge Delay mode is immediate (based on the DBRISEUPD field encoding in the **PWMnCTL** register), the 12-bit RISEDELAY value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMn Dead-Band Rising-Edge Delay (PWMnDBRISE)

PWM0 base: 0x4002.8000
Offset 0x06C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |
| 11:0 | RISEDELAY | RW | 0x000 | Dead-Band Rise Delay<br>The number of clock cycles to delay the rising edge of pwmA' after the rising edge of pwmA. |

## Register 60: PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070

## Register 61: PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0

## Register 62: PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0

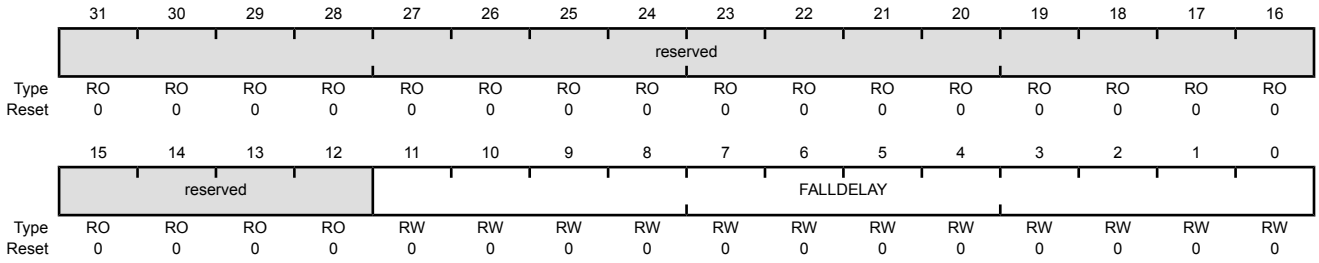
## Register 63: PWM3 Dead-Band Falling-Edge-Delay (PWM3DBFALL), offset 0x130

The **PWMnDBFALL** register contains the number of clock cycles to delay the rising edge of the pwmB' signal from the falling edge of the pwmA signal. If the dead-band generator is disabled through the **PWMnDBCTL** register, this register is ignored. If the value of this register is larger than the width of a Low pulse on the pwmA signal, the falling-edge delay consumes the entire Low time of the signal, resulting in no Low time on the output. Care must be taken to ensure that the pwmA Low time always exceeds the falling-edge delay.

If the Dead-Band Falling-Edge-Delay mode is immediate (based on the DBFALLUP field encoding in the **PWMnCTL** register), the 12-bit FALLDELAY value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1939). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWMn Dead-Band Falling-Edge-Delay (PWMnDBFALL)

PWM0 base: 0x4002.8000
Offset 0x070
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:12 | reserved | RO | 0x0000.0 | 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. |
| 11:0 | FALLDELAY | RW | 0x000 | Dead-Band Fall Delay<br>The number of clock cycles to delay the falling edge of pwmB' from the rising edge of pwmA. |

## Register 64: PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074

## Register 65: PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4

## Register 66: PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4

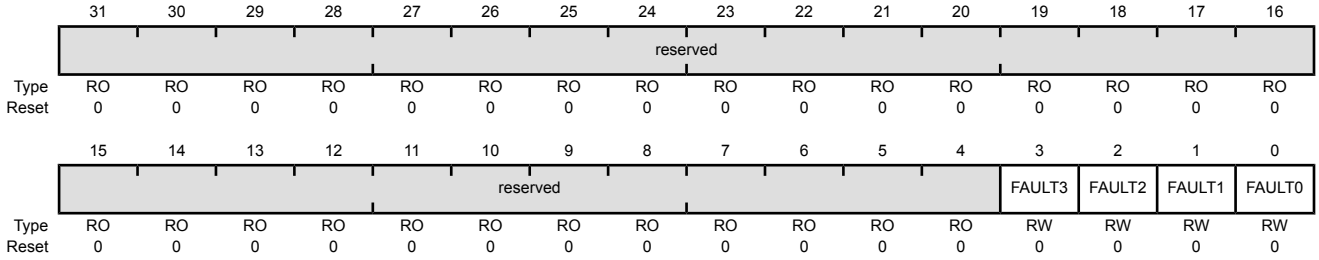
## Register 67: PWM3 Fault Source 0 (PWM3FLTSRC0), offset 0x134

This register specifies which fault pin inputs are used to generate a fault condition. Each bit in the following register indicates whether the corresponding fault pin is included in the fault condition. All enabled fault pins are ORed together to form the **PWMnFLTSRC0** portion of the fault condition. The **PWMnFLTSRC0** fault condition is then ORed with the **PWMnFLTSRC1** fault condition to generate the final fault condition for the PWM generator.

If the FLTSRC bit in the **PWMnCTL** register (see page 1964) is clear, only the Fault0 signal affects the fault condition generated. Otherwise, sources defined in **PWMnFLTSRC0** and **PWMnFLTSRC1** affect the fault condition generated.

PWMn Fault Source 0 (PWMnFLTSRC0)

PWM0 base: 0x4002.8000
Offset 0x074
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | FAULT3 | RW | 0 | Fault3 Input |

| Value | Description |
|---|---|
| 0 | The Fault3 signal is suppressed and cannot generate a fault condition. |
| 1 | The Fault3 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). |

**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | FAULT2 | RW | 0 | Fault2 Input<br><br>Value Description<br>0 The Fault2 signal is suppressed and cannot generate a fault condition.<br>1 The Fault2 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |
| 1 | FAULT1 | RW | 0 | Fault1 Input<br><br>Value Description<br>0 The Fault1 signal is suppressed and cannot generate a fault condition.<br>1 The Fault1 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |
| 0 | FAULT0 | RW | 0 | Fault0 Input<br><br>Value Description<br>0 The Fault0 signal is suppressed and cannot generate a fault condition.<br>1 The Fault0 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |

## Register 68: PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078

## Register 69: PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0x0B8

## Register 70: PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0x0F8

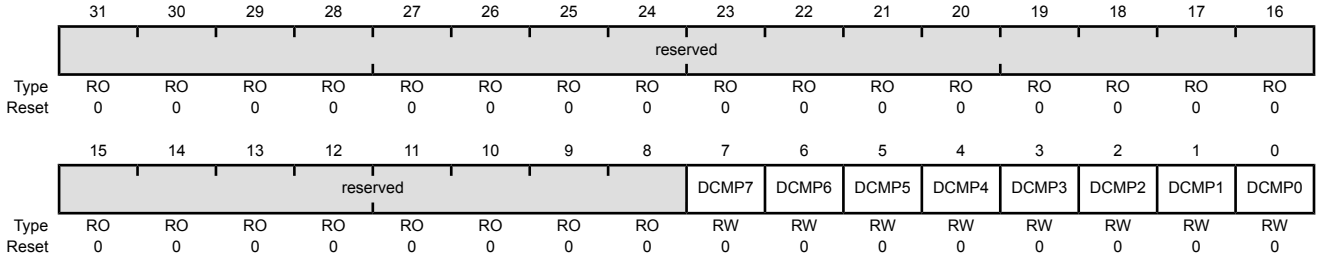
## Register 71: PWM3 Fault Source 1 (PWM3FLTSRC1), offset 0x138

This register specifies which digital comparator triggers from the ADC are used to generate a fault condition. Each bit in the following register indicates whether the corresponding digital comparator trigger is included in the fault condition. All enabled digital comparator triggers are ORed together to form the **PWMnFLTSRC1** portion of the fault condition. The **PWMnFLTSRC1** fault condition is then ORed with the **PWMnFLTSRC0** fault condition to generate the final fault condition for the PWM generator.

If the FLTSRC bit in the **PWMnCTL** register (see page 1964) is clear, only the PWM Fault0 pin affects the fault condition generated. Otherwise, sources defined in **PWMnFLTSRC0** and **PWMnFLTSRC1** affect the fault condition generated.

PWMn Fault Source 1 (PWMnFLTSRC1)

PWM0 base: 0x4002.8000
Offset 0x078
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | DCMP7 | RW | 0 | Digital Comparator 7<br>Value Description<br>0 The trigger from digital comparator 7 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 7 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | DCMP6 | RW | 0 | Digital Comparator 6<br><br>Value Description<br>0 The trigger from digital comparator 6 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 6 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |
| 5 | DCMP5 | RW | 0 | Digital Comparator 5<br><br>Value Description<br>0 The trigger from digital comparator 5 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 5 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |
| 4 | DCMP4 | RW | 0 | Digital Comparator 4<br><br>Value Description<br>0 The trigger from digital comparator 4 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 4 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |
| 3 | DCMP3 | RW | 0 | Digital Comparator 3<br><br>Value Description<br>0 The trigger from digital comparator 3 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 3 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | DCMP2 | RW | 0 | Digital Comparator 2<br><br>Value Description<br>0 The trigger from digital comparator 2 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 2 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |
| 1 | DCMP1 | RW | 0 | Digital Comparator 1<br><br>Value Description<br>0 The trigger from digital comparator 1 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 1 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |
| 0 | DCMP0 | RW | 0 | Digital Comparator 0<br><br>Value Description<br>0 The trigger from digital comparator 0 is suppressed and cannot generate a fault condition.<br>1 The trigger from digital comparator 0 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).<br><br>**Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation. |

### Register 72: PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C

### Register 73: PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC

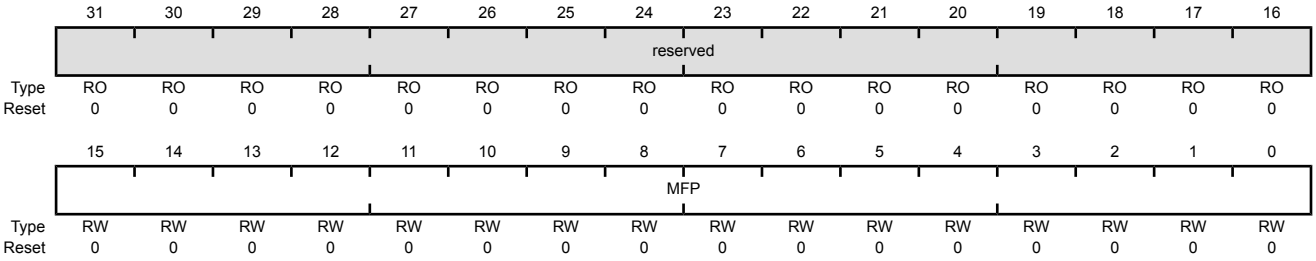
### Register 74: PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC

### Register 75: PWM3 Minimum Fault Period (PWM3MINFLTPER), offset 0x13C

If the MINFLTPER bit in the **PWMnCTL** register is set, this register specifies the 16-bit time-extension value to be used in extending the fault condition. The value is loaded into a 16-bit down counter, and the counter value is used to extend the fault condition. The fault condition is released in the clock immediately after the counter value reaches 0. The fault condition is asynchronous to the PWM clock; and the delay value is the product of the PWM clock period and the (MFP field value + 1) or (MFP field value + 2) depending on when the fault condition asserts with respect to the PWM clock. The counter decrements at the PWM clock rate, without pause or condition.

PWMn Minimum Fault Period (PWMnMINFLTPER)

PWM0 base: 0x4002.8000
Offset 0x07C
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | 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. |
| 15:0 | MFP | RW | 0x0000 | Minimum Fault Period<br>The number of PWM clocks by which a fault condition is extended when the delay is enabled by **PWMnCTL** MINFLTPER. |

## Register 76: PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800
## Register 77: PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880
## Register 78: PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900
## Register 79: PWM3 Fault Pin Logic Sense (PWM3FLTSEN), offset 0x980

This register defines the PWM fault pin logic sense.

PWMn Fault Pin Logic Sense (PWMnFLTSEN)

PWM0 base: 0x4002.8000
Offset 0x800
Type RW, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | FAULT3 | FAULT2 | FAULT1 | FAULT0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | FAULT3 | RW | 0 | Fault3 Sense<br>Value Description<br>0 An error is indicated if the Fault3 signal is High.<br>1 An error is indicated if the Fault3 signal is Low. |
| 2 | FAULT2 | RW | 0 | Fault2 Sense<br>Value Description<br>0 An error is indicated if the Fault2 signal is High.<br>1 An error is indicated if the Fault2 signal is Low. |
| 1 | FAULT1 | RW | 0 | Fault1 Sense<br>Value Description<br>0 An error is indicated if the Fault1 signal is High.<br>1 An error is indicated if the Fault1 signal is Low. |
| 0 | FAULT0 | RW | 0 | Fault0 Sense<br>Value Description<br>0 An error is indicated if the Fault0 signal is High.<br>1 An error is indicated if the Fault0 signal is Low. |

### Register 80: PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804

### Register 81: PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884

### Register 82: PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904

### Register 83: PWM3 Fault Status 0 (PWM3FLTSTAT0), offset 0x984

Along with the **PWMnFLTSTAT1** register, this register provides status regarding the fault condition inputs.

If the LATCH bit in the **PWMnCTL** register is clear, the contents of the **PWMnFLTSTAT0** register are read-only (RO) and provide the current state of the MnFAULTn inputs.

If the LATCH bit in the **PWMnCTL** register is set, the contents of the **PWMnFLTSTAT0** register are read / write 1 to clear (RW1C) and provide a latched version of the MnFAULTn inputs. In this mode, the register bits are cleared by writing a 1 to a set bit. The MnFAULTn inputs are recorded after their sense is adjusted in the generator.

The contents of this register can only be written if the fault source extensions are enabled (the FLTSRC bit in the **PWMnCTL** register is set).

**Note:** The fault status registers, **PWMnFLTSTAT0** and **PWMnFLTSTAT1**, reflect the status of all fault sources, regardless of what fault sources are enabled for that particular generator.

PWMn Fault Status 0 (PWMnFLTSTAT0)

PWM0 base: 0x4002.8000
Offset 0x804
Type -, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | FAULT3 | FAULT2 | FAULT1 | FAULT0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | - | - | - | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | 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. |
| 3 | FAULT3 | - | 0 | Fault Input 3<br>If the **PWMnCTL** register LATCH bit is clear, this bit is RO and represents the current state of the MnFAULT3 input signal after the logic sense adjustment.<br>If the **PWMnCTL** register LATCH bit is set, this bit is RW1C and represents a sticky version of the MnFAULT3 input signal after the logic sense adjustment.<br>■ If FAULT3 is set, the input transitioned to the active state previously.<br>■ If FAULT3 is clear, the input has not transitioned to the active state since the last time it was cleared.<br>■ The FAULT3 bit is cleared by writing it with the value 1. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | FAULT2 | - | 0 | Fault Input 2<br>If the **PWMnCTL** register LATCH bit is clear, this bit is RO and represents the current state of the MnFAULT2 input signal after the logic sense adjustment.<br>If the **PWMnCTL** register LATCH bit is set, this bit is RW1C and represents a sticky version of the MnFAULT2 input signal after the logic sense adjustment.<br>■ If FAULT2 is set, the input transitioned to the active state previously.<br>■ If FAULT2 is clear, the input has not transitioned to the active state since the last time it was cleared.<br>■ The FAULT2 bit is cleared by writing it with the value 1. |
| 1 | FAULT1 | - | 0 | Fault Input 1<br>If the **PWMnCTL** register LATCH bit is clear, this bit is RO and represents the current state of the MnFAULT1 input signal after the logic sense adjustment.<br>If the **PWMnCTL** register LATCH bit is set, this bit is RW1C and represents a sticky version of the MnFAULT1 input signal after the logic sense adjustment.<br>■ If FAULT1 is set, the input transitioned to the active state previously.<br>■ If FAULT1 is clear, the input has not transitioned to the active state since the last time it was cleared.<br>■ The FAULT1 bit is cleared by writing it with the value 1. |
| 0 | FAULT0 | - | 0 | Fault Input 0<br>If the **PWMnCTL** register LATCH bit is clear, this bit is RO and represents the current state of the input signal after the logic sense adjustment.<br>If the **PWMnCTL** register LATCH bit is set, this bit is RW1C and represents a sticky version of the input signal after the logic sense adjustment.<br>■ If FAULT0 is set, the input transitioned to the active state previously.<br>■ If FAULT0 is clear, the input has not transitioned to the active state since the last time it was cleared.<br>■ The FAULT0 bit is cleared by writing it with the value 1. |

## Register 84: PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808

## Register 85: PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888

## Register 86: PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908

## Register 87: PWM3 Fault Status 1 (PWM3FLTSTAT1), offset 0x988

Along with the **PWMnFLTSTAT0** register, this register provides status regarding the fault condition inputs.

If the LATCH bit in the **PWMnCTL** register is clear, the contents of the **PWMnFLTSTAT1** register are read-only (RO) and provide the current state of the digital comparator triggers.

If the LATCH bit in the **PWMnCTL** register is set, the contents of the **PWMnFLTSTAT1** register are read / write 1 to clear (RW1C) and provide a latched version of the digital comparator triggers. In this mode, the register bits are cleared by writing a 1 to a set bit. The contents of this register can only be written if the fault source extensions are enabled (the FLTSRC bit in the **PWMnCTL** register is set).

**Note:** The fault status registers, **PWMnFLTSTAT0** and **PWMnFLTSTAT1**, reflect the status of all fault sources, regardless of what fault sources are enabled for that particular generator.

PWMn Fault Status 1 (PWMnFLTSTAT1)

PWM0 base: 0x4002.8000
Offset 0x808
Type -, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | DCMP7 | DCMP6 | DCMP5 | DCMP4 | DCMP3 | DCMP2 | DCMP1 | DCMP0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | - | - | - | - | - | - | - | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | 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. |
| 7 | DCMP7 | - | 0 | Digital Comparator 7 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 7 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP7 is set, the trigger transitioned to the active state previously.<br>■ If DCMP7 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP7 bit is cleared by writing it with the value 1. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | DCMP6 | - | 0 | Digital Comparator 6 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 6 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP6 is set, the trigger transitioned to the active state previously.<br>■ If DCMP6 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP6 bit is cleared by writing it with the value 1. |
| 5 | DCMP5 | - | 0 | Digital Comparator 5 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 5 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP5 is set, the trigger transitioned to the active state previously.<br>■ If DCMP5 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP5 bit is cleared by writing it with the value 1. |
| 4 | DCMP4 | - | 0 | Digital Comparator 4 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 4 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP4 is set, the trigger transitioned to the active state previously.<br>■ If DCMP4 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP4 bit is cleared by writing it with the value 1. |
| 3 | DCMP3 | - | 0 | Digital Comparator 3 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 3 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP3 is set, the trigger transitioned to the active state previously.<br>■ If DCMP3 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP3 bit is cleared by writing it with the value 1. |

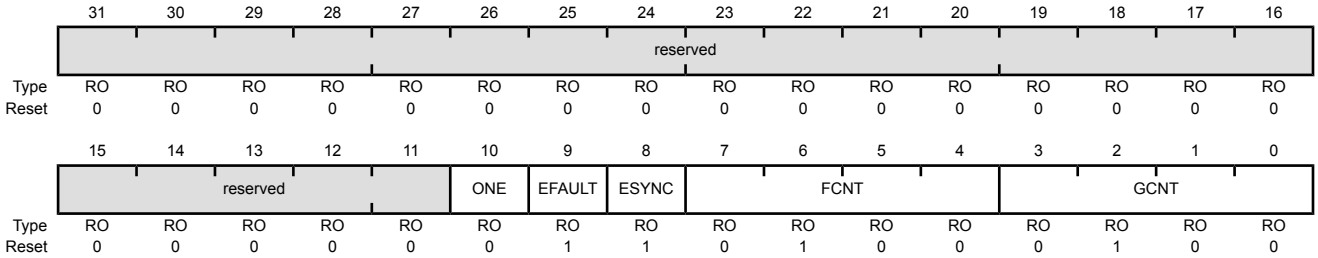
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | DCMP2 | - | 0 | Digital Comparator 2 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 2 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP2 is set, the trigger transitioned to the active state previously.<br>■ If DCMP2 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP2 bit is cleared by writing it with the value 1. |
| 1 | DCMP1 | - | 0 | Digital Comparator 1 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 1 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP1 is set, the trigger transitioned to the active state previously.<br>■ If DCMP1 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP1 bit is cleared by writing it with the value 1. |
| 0 | DCMP0 | - | 0 | Digital Comparator 0 Trigger<br>If the **PWMnCTL** register LATCH bit is clear, this bit represents the current state of the Digital Comparator 0 trigger input.<br>If the **PWMnCTL** register LATCH bit is set, this bit represents a sticky version of the trigger.<br>■ If DCMP0 is set, the trigger transitioned to the active state previously.<br>■ If DCMP0 is clear, the trigger has not transitioned to the active state since the last time it was cleared.<br>■ The DCMP0 bit is cleared by writing it with the value 1. |

## Register 88: PWM Peripheral Properties (PWMPP), offset 0xFC0

The **PWMPP** register provides information regarding the properties of the PWM module.

PWM Peripheral Properties (PWMPP)

PWM0 base: 0x4002.8000
Offset 0xFC0
Type RO, reset 0x0000.0344




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:11 | reserved | RO | 0x0 | 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. |
| 10 | ONE | RO | 0x0 | One-Shot Mode<br>Value Description<br>0 One-shot modes are not available.<br>1 One-shot modes are available. |
| 9 | EFAULT | RO | 0x1 | Extended Fault<br>Value Description<br>0 Extended fault capabilities are not available.<br>1 Extended fault capabilities are available. |
| 8 | ESYNC | RO | 0x1 | Extended Synchronization<br>Value Description<br>0 Extended synchronization is not available.<br>1 Extended synchronization is available. |
| 7:4 | FCNT | RO | 0x4 | Fault Inputs<br>Value Description<br>0x0 No fault inputs.<br>0x1 1 fault input.<br>0x2 2 fault input.<br>0x3 3 fault input.<br>0x4 4 fault input.<br>0x5 - 0xF reserved |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | GCNT | RO | 0x4 | Generators |

| Value | Description |
|---|---|
| 0x0 | No generators. |
| 0x1 | 1 generator |
| 0x2 | 2 generators |
| 0x3 | 3 generators |
| 0x4 | 4 generators |
| 0x5 - 0xF | reserved |

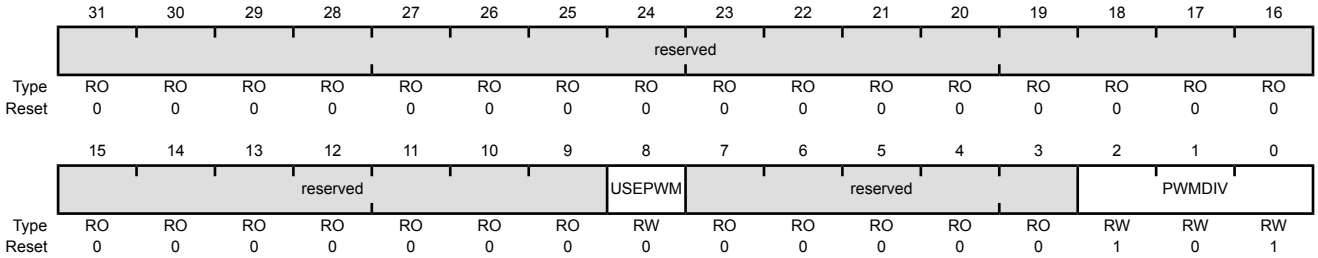
The number of PWM outputs is 2 times the number of PWM generators.

## Register 89: PWM Clock Configuration (PWMCC), offset 0xFC8

The **PWMCC** register controls the clock source for the PWM module.

PWM Clock Configuration (PWMCC)

PWM0 base: 0x4002.8000
Offset 0xFC8
Type RW, reset 0x0000.0005




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:9 | reserved | RO | 0x0000.0000 | 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. |
| 8 | USEPWM | RW | 0x0 | Use PWM Clock Divisor<br><br>Value Description<br>0 The system clock is the source of PWM unit clock.<br>1 The PWM clock divider is the source of PWM unit clock. |
| 7:3 | reserved | RO | 0 | 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. |
| 2:0 | PWMDIV | RW | 0x5 | PWM Clock Divider<br>This field specifies the PWM clock frequency as a division of the system clock.<br><br>Value Description<br>0x0 /2<br>0x1 /4<br>0x2 /8<br>0x3 /16<br>0x4 /32<br>0x5 /64<br>0x6 - 0x7 reserved |

# 29 Quadrature Encoder Interface (QEI)

A quadrature encoder, also known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring both the number of pulses and the relative phase of the two signals, you can track the position, direction of rotation, and speed. In addition, a third channel, or index signal, can be used to reset the position counter.

The TM4C129XKCZAD quadrature encoder interface (QEI) module interprets the code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

The TM4C129XKCZAD microcontroller includes one QEI module with the following features:

- Position integrator that tracks the encoder position
- Programmable noise filter on the inputs
- Velocity capture using built-in timer
- The input frequency of the QEI inputs may be as high as 1/4 of the processor frequency (for example, 12.5 MHz for a 50-MHz system)
- Interrupt generation on:
  - Index pulse
  - Velocity-timer expiration
  - Direction change
  - Quadrature error detection

## 29.1 Block Diagram

Figure 29-1 on page 2005 provides an internal block diagram of a TM4C129XKCZAD QEI module. The PhA and PhB inputs shown in this diagram are the internal signals that enter the Quadrature Encoder after the external signals, PhAn and PhBn, have passed through inversion and swapping logic shown in Figure 29-2 on page 2006. The QEI module has the option of inverting and/or swapping the incoming signals.

**Note:** Any references in this chapter to PhA and PhB refer to the internal PhA and PhB inputs that enter the Quadrature Encoder after the external signals, PhAn and PhBn, have passed through inversion and swapping logic that is enabled through the **QEI Control (QEICTL)** register.

**Figure 29-1. QEI Block Diagram**

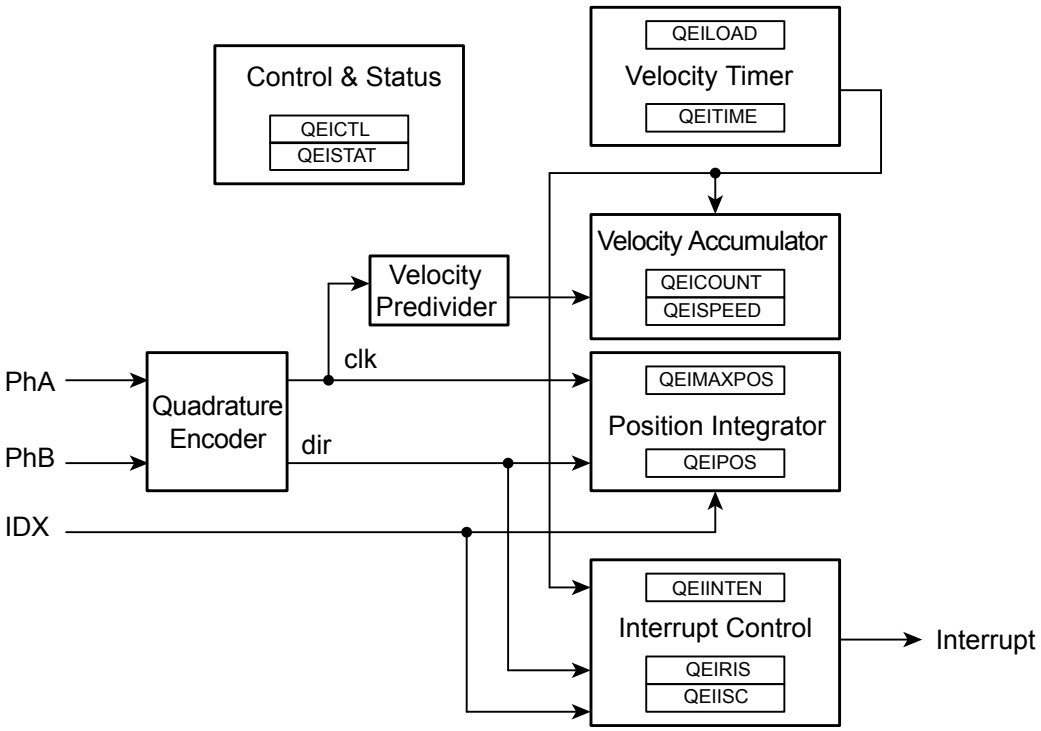
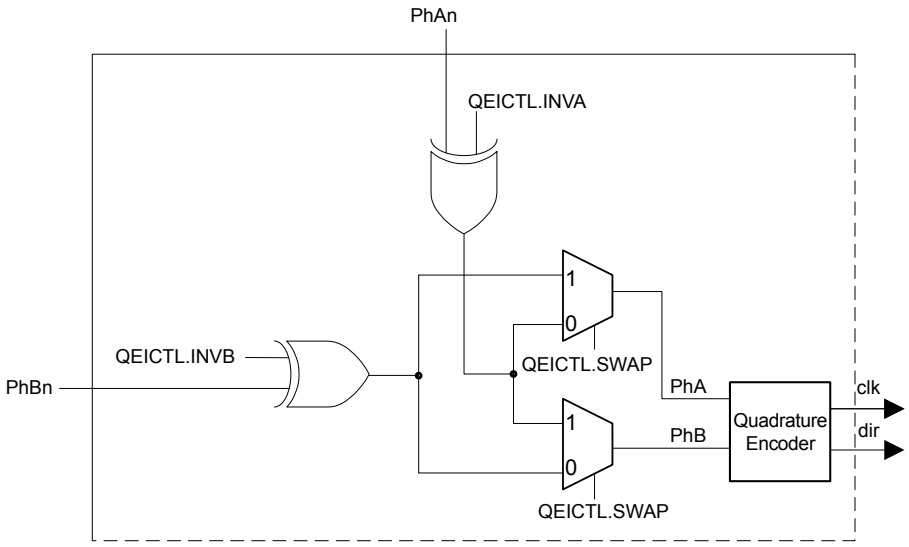



Figure 29-2 on page 2006 shows the logic that is provided to allow the PhAn and PhBn signals to be inverted and/or swapped.

**Figure 29-2. QEI Input Signal Logic**




## 29.2 Signal Description

The following table lists the external signals of the QEI module and describes the function of each. The QEI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these QEI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 803) should be set to choose the QEI function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 820) to assign the QEI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 772.

**Table 29-1. QEI Signals (212BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| IDX0 | J18<br>U10 | PL3 (6)<br>PS6 (6) | I | TTL | QEI module 0 index. |
| PhA0 | H19<br>V9 | PL1 (6)<br>PS4 (6) | I | TTL | QEI module 0 phase A. |
| PhB0 | G18<br>T13 | PL2 (6)<br>PS5 (6) | I | TTL | QEI module 0 phase B. |

## 29.3 Functional Description

The QEI module interprets the two-bit gray code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

The position integrator and velocity capture can be independently enabled, though the position integrator must be enabled before the velocity capture can be enabled. The two phase signals,

PhAn and PhBn, can be swapped before being interpreted by the QEI module to change the meaning of forward and backward and to correct for miswiring of the system. Alternatively, the phase signals can be interpreted as a clock and direction signal as output by some encoders.

The QEI module input signals have a digital noise filter on them that can be enabled to prevent spurious operation. The noise filter requires that the inputs be stable for a specified number of consecutive clock cycles before updating the edge detector. The filter is enabled by the FILTEN bit in the **QEI Control (QEICTL)** register. The frequency of the input update is programmable using the FILTCNT bit field in the **QEICTL** register.

The QEI module supports two modes of signal operation: quadrature phase mode and clock/direction mode. In quadrature phase mode, the encoder produces two clocks that are 90 degrees out of phase; the edge relationship is used to determine the direction of rotation. In clock/direction mode, the encoder produces a clock signal to indicate steps and a direction signal to indicate the direction of rotation. This mode is determined by the SIGMODE bit of the **QEICTL** register (see page 2011).

When the QEI module is set to use the quadrature phase mode (SIGMODE bit is clear), the capture mode for the position integrator can be set to update the position counter on every edge of the PhA signal or to update on every edge of both PhA and PhB. Updating the position counter on every PhA and PhB edge provides more positional resolution at the cost of less range in the positional counter.

When edges on PhA lead edges on PhB, the position counter is incremented. When edges on PhB lead edges on PhA, the position counter is decremented. When a rising and falling edge pair is seen on one of the phases without any edges on the other, the direction of rotation has changed.

The positional counter is automatically reset on one of two conditions: sensing the index pulse or reaching the maximum position value. The reset mode is determined by the RESMODE bit of the **QEICTL** register.

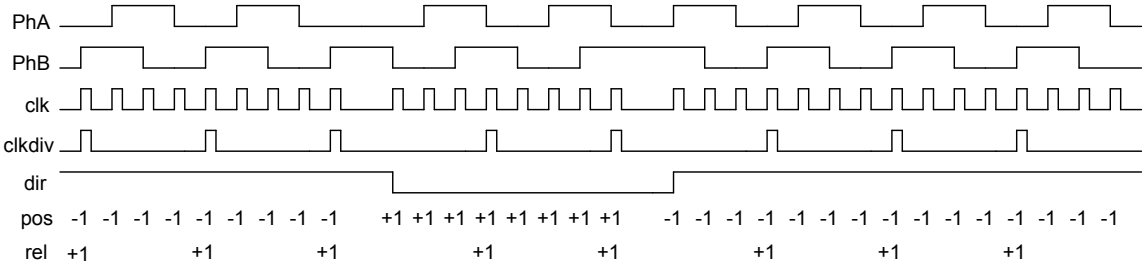
When RESMODE is set, the positional counter is reset when the index pulse is sensed. This mode limits the positional counter to the values [0:N-1], where N is the number of phase edges in a full revolution of the encoder wheel. The **QEI Maximum Position (QEIMAXPOS)** register must be programmed with N-1 so that the reverse direction from position 0 can move the position counter to N-1. In this mode, the position register contains the absolute position of the encoder relative to the index (or home) position once an index pulse has been seen.

When RESMODE is clear, the positional counter is constrained to the range [0:M], where M is the programmable maximum value. The index pulse is ignored by the positional counter in this mode.

Velocity capture uses a configurable timer and a count register. The timer counts the number of phase edges (using the same configuration as for the position integrator) in a given time period. The edge count from the previous time period is available to the controller via the **QEI Velocity (QEISPEED)** register, while the edge count for the current time period is being accumulated in the **QEI Velocity Counter (QEICOUNT)** register. As soon as the current time period is complete, the total number of edges counted in that time period is made available in the **QEISPEED** register (overwriting the previous value), the **QEICOUNT** register is cleared, and counting commences on a new time period. The number of edges counted in a given time period is directly proportional to the velocity of the encoder.

Figure 29-3 on page 2008 shows how the TM4C129XKCZAD quadrature encoder converts the phase input signals into clock pulses, the direction signal, and how the velocity predivider operates (in Divide by 4 mode).

**Figure 29-3. Quadrature Encoder and Velocity Predivider Operation**




The period of the timer is configurable by specifying the load value for the timer in the **QEI Timer Load (QEILOAD)** register. When the timer reaches zero, an interrupt can be triggered, and the hardware reloads the timer with the **QEILOAD** value and continues to count down. At lower encoder speeds, a longer timer period is required to be able to capture enough edges to have a meaningful result. At higher encoder speeds, both a shorter timer period and/or the velocity predivider can be used.

The following equation converts the velocity counter value into an rpm value:

```
rpm = (clock * (2 ^ VELDIV) * SPEED * 60) ÷ (LOAD * ppr * edges)
```

where:

`clock` is the controller clock rate

`ppr` is the number of pulses per revolution of the physical encoder

`edges` is 2 or 4, based on the capture mode set in the **QEICTL** register (2 for `CAPMODE` clear and 4 for `CAPMODE` set)

For example, consider a motor running at 600 rpm. A 2048 pulse per revolution quadrature encoder is attached to the motor, producing 8192 phase edges per revolution. With a velocity predivider of ÷1 (`VELDIV` is clear) and clocking on both `PhA` and `PhB` edges, this results in 81,920 pulses per second (the motor turns 10 times per second). If the timer were clocked at 10,000 Hz, and the load value was 2,500 (¼ of a second), it would count 20,480 pulses per update. Using the above equation:

```
rpm = (10000 * 1 * 20480 * 60) ÷ (2500 * 2048 * 4) = 600 rpm
```

Now, consider that the motor is sped up to 3000 rpm. This results in 409,600 pulses per second, or 102,400 every ¼ of a second. Again, the above equation gives:

```
rpm = (10000 * 1 * 102400 * 60) ÷ (2500 * 2048 * 4) = 3000 rpm
```

Care must be taken when evaluating this equation because intermediate values may exceed the capacity of a 32-bit integer. In the above examples, the clock is 10,000 and the divider is 2,500; both could be predivided by 100 (at compile time if they are constants) and therefore be 100 and 25. In fact, if they were compile-time constants, they could also be reduced to a simple multiply by 4, cancelled by the ÷4 for the edge-count factor.

**Important:** Reducing constant factors at compile time is the best way to control the intermediate values of this equation and reduce the processing requirement of computing this equation.

The division can be avoided by selecting a timer load value such that the divisor is a power of 2; a simple shift can therefore be done in place of the division. For encoders with a power of 2 pulses per revolution, the load value can be a power of 2. For other encoders, a load value must be selected such that the product is very close to a power of 2. For example, a 100 pulse-per-revolution encoder

could use a load value of 82, resulting in 32,800 as the divisor, which is 0.09% above $2^{14}$. In this case a shift by 15 would be an adequate approximation of the divide in most cases. If absolute accuracy were required, the microcontroller's divide instruction could be used.

The QEI module can produce a controller interrupt on several events: phase error, direction change, reception of the index pulse, and expiration of the velocity timer. Standard masking, raw interrupt status, interrupt status, and interrupt clear capabilities are provided.

## 29.4 Initialization and Configuration

The following example shows how to configure the Quadrature Encoder module to read back an absolute position:

1. Enable the QEI clock using the **RCGCQEI** register in the System Control module (see page 416).
2. Enable the clock to the appropriate GPIO module via the **RCGCGPIO** register in the System Control module (see page 399).
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. To determine which GPIOs to configure, see Table 31-4 on page 2064.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the QEI signals to the appropriate pins (see page 820 and Table 31-5 on page 2081).
5. Configure the quadrature encoder to capture edges on both signals and maintain an absolute position by resetting on index pulses. A 1000-line encoder with four edges per line, results in 4000 pulses per revolution; therefore, set the maximum position to 3999 (0xF9F) as the count is zero-based.
   - Write the **QEICTL** register with the value of 0x0000.0018.
   - Write the **QEIMAXPOS** register with the value of 0x0000.0F9F.
6. Enable the quadrature encoder by setting bit 0 of the **QEICTL** register.

   **Note:** Once the QEI module has been enabled by setting the ENABLE bit in the **QEICTL** register, it cannot be disabled. The only way to clear the ENABLE bit is to reset the module using the **Quadrature Encoder Interface Software Reset (SRQEI)** register.
7. Delay until the encoder position is required.
8. Read the encoder position by reading the **QEI Position (QEIPOS)** register value.

**Note:** If the application requires the quadrature encoder to have a specific initial position, this value must be programmed in the **QEIPOS** register after the quadrature encoder has been enabled by setting the ENABLE bit in the **QEICTL** register.

## 29.5 Register Map

Table 29-2 on page 2010 lists the QEI registers. The offset listed is a hexadecimal increment to the register's address, relative to the module's base address:

- QEI0: 0x4002.C000

Note that the QEI module clock must be enabled before the registers can be programmed (see page 416). There must be a delay of 3 system clocks after the QEI module clock is enabled before any QEI module registers are accessed.

**Table 29-2. QEI Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | QEICTL | RW | 0x0000.0000 | QEI Control | 2011 |
| 0x004 | QEISTAT | RO | 0x0000.0000 | QEI Status | 2014 |
| 0x008 | QEIPOS | RW | 0x0000.0000 | QEI Position | 2015 |
| 0x00C | QEIMAXPOS | RW | 0x0000.0000 | QEI Maximum Position | 2016 |
| 0x010 | QEILOAD | RW | 0x0000.0000 | QEI Timer Load | 2017 |
| 0x014 | QEITIME | RO | 0x0000.0000 | QEI Timer | 2018 |
| 0x018 | QEICOUNT | RO | 0x0000.0000 | QEI Velocity Counter | 2019 |
| 0x01C | QEISPEED | RO | 0x0000.0000 | QEI Velocity | 2020 |
| 0x020 | QEIINTEN | RW | 0x0000.0000 | QEI Interrupt Enable | 2021 |
| 0x024 | QEIRIS | RO | 0x0000.0000 | QEI Raw Interrupt Status | 2023 |
| 0x028 | QEIISC | RW1C | 0x0000.0000 | QEI Interrupt Status and Clear | 2025 |

## 29.6 Register Descriptions

The remainder of this section lists and describes the QEI registers, in numerical order by address offset.

## Register 1: QEI Control (QEICTL), offset 0x000

This register contains the configuration of the QEI module. Separate enables are provided for the quadrature encoder and the velocity capture blocks; the quadrature encoder must be enabled in order to capture the velocity, but the velocity does not need to be captured in applications that do not need it. The phase signal interpretation, phase swap, Position Update mode, Position Reset mode, and velocity predivider are all set via this register.

QEI Control (QEICTL)

QEI0 base: 0x4002.C000
Offset 0x000
Type RW, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | FILTCNT | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | FILTEN | STALLEN | INVI | INVB | INVA | | VELDIV | | VELEN | RESMODE | CAPMODE | SIGMODE | SWAP | ENABLE |
| Type | RO | RO | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:20 | reserved | RO | 0x000 | 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. |
| 19:16 | FILTCNT | RW | 0x0 | Input Filter Prescale Count<br>This field controls the frequency of the input update.<br>When this field is clear, the input is sampled after 2 system clocks. When this field ix 0x1, the input is sampled after 3 system clocks. Similarly, when this field is 0xF, the input is sampled after 17 clocks. |
| 15:14 | reserved | RO | 0x0 | 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. |
| 13 | FILTEN | RW | 0 | Enable Input Filter<br>Value Description<br>0 The QEI inputs are not filtered.<br>1 Enables the digital noise filter on the QEI input signals. Inputs must be stable for 3 consecutive clock edges before the edge detector is updated. |
| 12 | STALLEN | RW | 0 | Stall QEI<br>Value Description<br>0 The QEI module does not stall when the microcontroller is stopped by a debugger.<br>1 The QEI module stalls when the microcontroller is stopped by a debugger. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11 | INVI | RW | 0 | Invert Index Pulse<br>Value Description<br>0 No effect.<br>1 Inverts the IDX input. |
| 10 | INVB | RW | 0 | Invert PhB<br>Value Description<br>0 No effect.<br>1 Inverts the PhBn input. |
| 9 | INVA | RW | 0 | Invert PhA<br>Value Description<br>0 No effect.<br>1 Inverts the PhAn input. |
| 8:6 | VELDIV | RW | 0x0 | Predivide Velocity<br>This field defines the predivider of the input quadrature pulses before being applied to the **QEICOUNT** accumulator.<br>Value Predivider<br>0x0 ÷1<br>0x1 ÷2<br>0x2 ÷4<br>0x3 ÷8<br>0x4 ÷16<br>0x5 ÷32<br>0x6 ÷64<br>0x7 ÷128 |
| 5 | VELEN | RW | 0 | Capture Velocity<br>Value Description<br>0 No effect.<br>1 Enables capture of the velocity of the quadrature encoder. |
| 4 | RESMODE | RW | 0 | Reset Mode<br>Value Description<br>0 The position counter is reset when it reaches the maximum as defined by the MAXPOS field in the **QEIMAXPOS** register.<br>1 The position counter is reset when the index pulse is captured. |

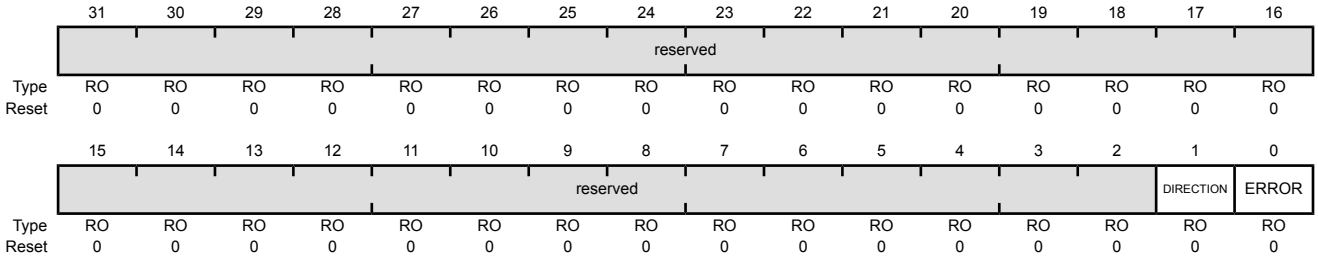
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | CAPMODE | RW | 0 | Capture Mode<br>**Note:** When SIGMODE=1, the CAPMODE setting is not applicable and is reserved.<br>Value Description<br>0 Only the PhA edges are counted.<br>1 The PhA and PhB edges are counted, providing twice the positional resolution but half the range. |
| 2 | SIGMODE | RW | 0 | Signal Mode<br>Value Description<br>0 The internal PhA and PhB signals operate as quadrature phase signals.<br>1 The internal PhA input operates as the clock (CLK) signal and the internal PhB input operates as the direction (DIR) signal. |
| 1 | SWAP | RW | 0 | Swap Signals<br>Note if the INVA or INVB bit are set, the inversion of the signals occur prior to the swap.<br>Value Description<br>0 No effect.<br>1 Swaps the PhAn and PhBn signals. |
| 0 | ENABLE | RW | 0 | Enable QEI<br>Value Description<br>0 No effect.<br>1 Enables the quadrature encoder module.<br>**Note:** Once the QEI module has been enabled by setting the ENABLE bit, it cannot be disabled. The only way to clear the ENABLE bit is to reset the module using the **Quadrature Encoder Interface Software Reset (SRQEI)** register. |

## Register 2: QEI Status (QEISTAT), offset 0x004

This register provides status about the operation of the QEI module.

QEI Status (QEISTAT)

QEI0 base: 0x4002.C000
Offset 0x004
Type RO, reset 0x0000.0000



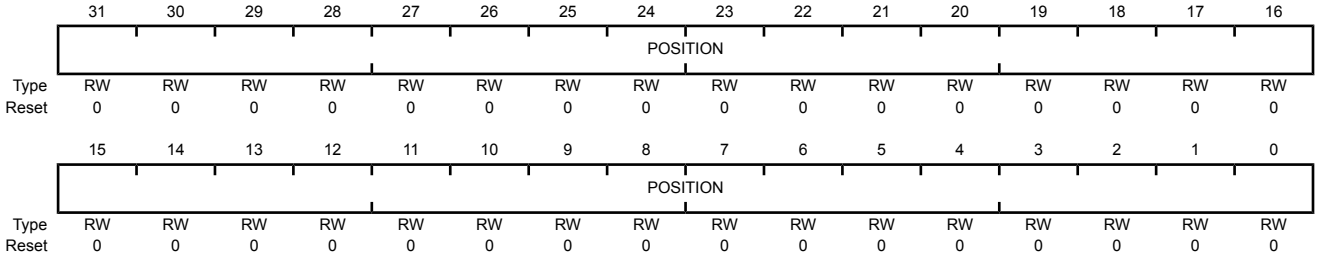

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | DIRECTION | RO | 0 | Direction of Rotation<br>Indicates the direction the encoder is rotating.<br><br>Value Description<br>0 The encoder is rotating forward.<br>1 The encoder is rotating in reverse. |
| 0 | ERROR | RO | 0 | Error Detected<br><br>Value Description<br>0 No error.<br>1 An error was detected in the gray code sequence (that is, both signals changing at the same time). |

## Register 3: QEI Position (QEIPOS), offset 0x008

This register contains the current value of the position integrator. The value is updated by the status of the QEI phase inputs and can be set to a specific value by writing to it.

QEI Position (QEIPOS)

QEI0 base: 0x4002.C000
Offset 0x008
Type RW, reset 0x0000.0000



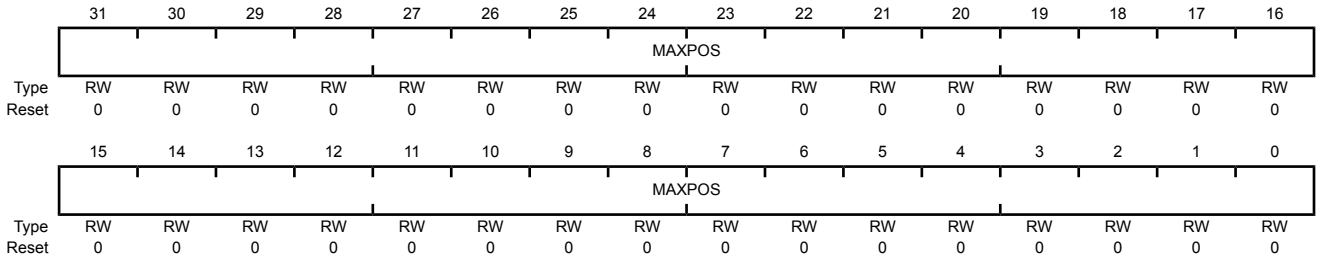

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | POSITION | RW | 0x0000.0000 | Current Position Integrator Value<br>The current value of the position integrator. |

## Register 4: QEI Maximum Position (QEIMAXPOS), offset 0x00C

This register contains the maximum value of the position integrator. When moving forward, the position register resets to zero when it increments past this value. When moving in reverse, the position register resets to this value when it decrements from zero.

QEI Maximum Position (QEIMAXPOS)

QEI0 base: 0x4002.C000
Offset 0x00C
Type RW, reset 0x0000.0000



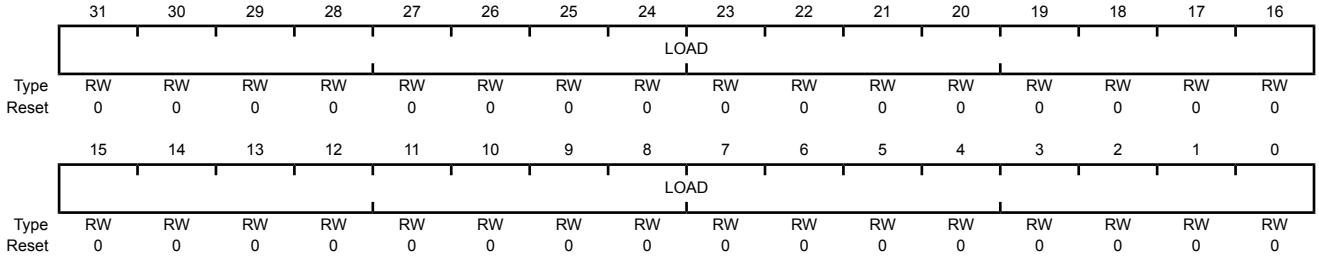

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | MAXPOS | RW | 0x0000.0000 | Maximum Position Integrator Value<br>The maximum value of the position integrator. |

## Register 5: QEI Timer Load (QEILOAD), offset 0x010

This register contains the load value for the velocity timer. Because this value is loaded into the timer on the clock cycle after the timer is zero, this value should be one less than the number of clocks in the desired period. So, for example, to have 2000 decimal clocks per timer period, this register should contain 1999 decimal.

QEI Timer Load (QEILOAD)

QEI0 base: 0x4002.C000
Offset 0x010
Type RW, reset 0x0000.0000



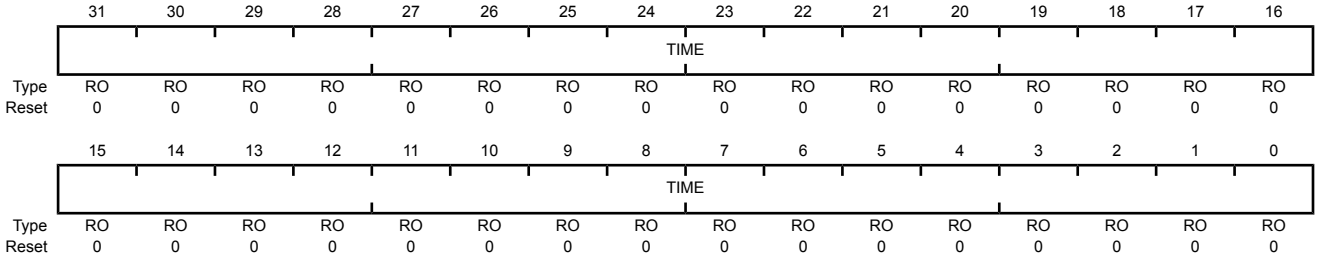

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LOAD | RW | 0x0000.0000 | Velocity Timer Load Value<br>The load value for the velocity timer. |

## Register 6: QEI Timer (QEITIME), offset 0x014

This register contains the current value of the velocity timer. This counter does not increment when the VELEN bit in the **QEICTL** register is clear.

QEI Timer (QEITIME)

QEI0 base: 0x4002.C000
Offset 0x014
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TIME | RO | 0x0000.0000 | Velocity Timer Current Value<br>The current value of the velocity timer. |

## Register 7: QEI Velocity Counter (QEICOUNT), offset 0x018

This register contains the running count of velocity pulses for the current time period. Because this count is a running total, the time period to which it applies cannot be known with precision (that is, a read of this register does not necessarily correspond to the time returned by the **QEITIME** register because there is a small window of time between the two reads, during which either value may have changed). The **QEISPEED** register should be used to determine the actual encoder velocity; this register is provided for information purposes only. This counter does not increment when the VELEN bit in the **QEICTL** register is clear.

QEI Velocity Counter (QEICOUNT)

QEI0 base: 0x4002.C000
Offset 0x018
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COUNT | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | COUNT | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

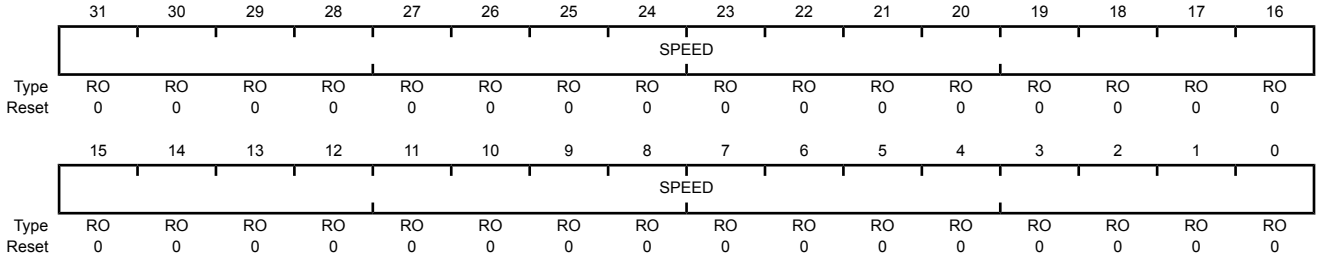
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | COUNT | RO | 0x0000.0000 | Velocity Pulse Count<br>The running total of encoder pulses during this velocity timer period. |

## Register 8: QEI Velocity (QEISPEED), offset 0x01C

This register contains the most recently measured velocity of the quadrature encoder. This value corresponds to the number of velocity pulses counted in the previous velocity timer period. This register does not update when the VELEN bit in the **QEICTL** register is clear.

QEI Velocity (QEISPEED)

QEI0 base: 0x4002.C000
Offset 0x01C
Type RO, reset 0x0000.0000



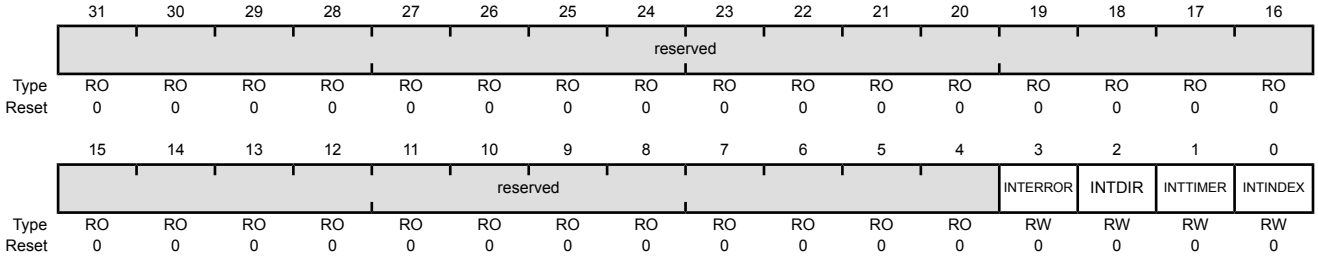

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SPEED | RO | 0x0000.0000 | Velocity<br>The measured speed of the quadrature encoder in pulses per period. |

## Register 9: QEI Interrupt Enable (QEIINTEN), offset 0x020

This register contains enables for each of the QEI module interrupts. An interrupt is asserted to the interrupt controller if the corresponding bit in this register is set.

QEI Interrupt Enable (QEIINTEN)

QEI0 base: 0x4002.C000
Offset 0x020
Type RW, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | INTERROR | RW | 0 | Phase Error Interrupt Enable<br>**Note:** The INTERROR bit is only applicable when the QEI is operating in quadrature phase mode (SIGMODE=0) and should be masked when SIGMODE =1.<br>Value Description<br>0 The INTERROR interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the INTERROR bit in the **QEIRIS** register is set. |
| 2 | INTDIR | RW | 0 | Direction Change Interrupt Enable<br>Value Description<br>0 The INTDIR interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the INTDIR bit in the **QEIRIS** register is set. |
| 1 | INTTIMER | RW | 0 | Timer Expires Interrupt Enable<br>Value Description<br>0 The INTTIMER interrupt is suppressed and not sent to the interrupt controller.<br>1 An interrupt is sent to the interrupt controller when the INTTIMER bit in the **QEIRIS** register is set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTINDEX | RW | 0 | Index Pulse Detected Interrupt Enable |

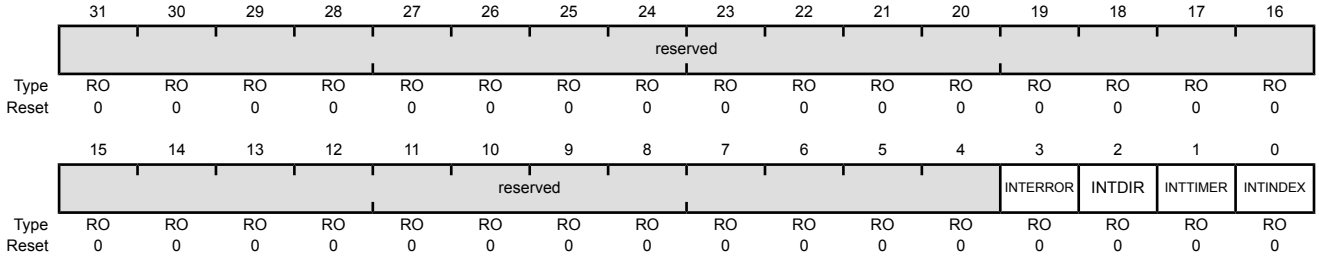
| Value | Description |
|---|---|
| 0 | The INTINDEX interrupt is suppressed and not sent to the interrupt controller. |
| 1 | An interrupt is sent to the interrupt controller when the INTINDEX bit in the **QEIRIS** register is set. |

## Register 10: QEI Raw Interrupt Status (QEIRIS), offset 0x024

This register provides the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (configured through the **QEIINTEN** register). If a bit is set, the latched event has occurred; if a bit is clear, the event in question has not occurred.

QEI Raw Interrupt Status (QEIRIS)

QEI0 base: 0x4002.C000
Offset 0x024
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | INTERROR | RO | 0 | Phase Error Detected<br>**Note:** The INTERROR bit is only applicable when the QEI is operating in quadrature phase mode (SIGMODE=0).<br>Value Description<br>0 An interrupt has not occurred.<br>1 A phase error has been detected.<br>This bit is cleared by writing a 1 to the INTERROR bit in the **QEIISC** register. |
| 2 | INTDIR | RO | 0 | Direction Change Detected<br>Value Description<br>0 An interrupt has not occurred.<br>1 The rotation direction has changed<br>This bit is cleared by writing a 1 to the INTDIR bit in the **QEIISC** register. |
| 1 | INTTIMER | RO | 0 | Velocity Timer Expired<br>Value Description<br>0 An interrupt has not occurred.<br>1 The velocity timer has expired.<br>This bit is cleared by writing a 1 to the INTTIMER bit in the **QEIISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTINDEX | RO | 0 | Index Pulse Asserted |

| Value | Description |
|---|---|
| 0 | An interrupt has not occurred. |
| 1 | The index pulse has occurred. |

This bit is cleared by writing a 1 to the `INTINDEX` bit in the **QEIISC** register.

## Register 11: QEI Interrupt Status and Clear (QEIISC), offset 0x028

This register provides the current set of interrupt sources that are asserted to the controller. If a bit is set, the latched event has occurred and is enabled to generate an interrupt; if a bit is clear the event in question has not occurred or is not enabled to generate an interrupt. This register is RW1C; writing a 1 to a bit position clears the bit and the corresponding interrupt reason.

QEI Interrupt Status and Clear (QEIISC)

QEI0 base: 0x4002.C000
Offset 0x028
Type RW1C, reset 0x0000.0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | | | | | | | | | INTERROR | INTDIR | INTTIMER | INTINDEX |
| RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW1C | RW1C | RW1C | RW1C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | 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. |
| 3 | INTERROR | RW1C | 0 | Phase Error Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The `INTERROR` bits in the **QEIRIS** register and the **QEIINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the `INTERROR` bit in the **QEIRIS** register. |
| 2 | INTDIR | RW1C | 0 | Direction Change Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The `INTDIR` bits in the **QEIRIS** register and the **QEIINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the `INTDIR` bit in the **QEIRIS** register. |
| 1 | INTTIMER | RW1C | 0 | Velocity Timer Expired Interrupt<br><br>Value Description<br>0 No interrupt has occurred or the interrupt is masked.<br>1 The `INTTIMER` bits in the **QEIRIS** register and the **QEIINTEN** registers are set, providing an interrupt to the interrupt controller.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the `INTTIMER` bit in the **QEIRIS** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INTINDEX | RW1C | 0 | Index Pulse Interrupt |

| Value | Description |
|---|---|
| 0 | No interrupt has occurred or the interrupt is masked. |
| 1 | The INTINDEX bits in the **QEIRIS** register and the **QEIINTEN** registers are set, providing an interrupt to the interrupt controller. |

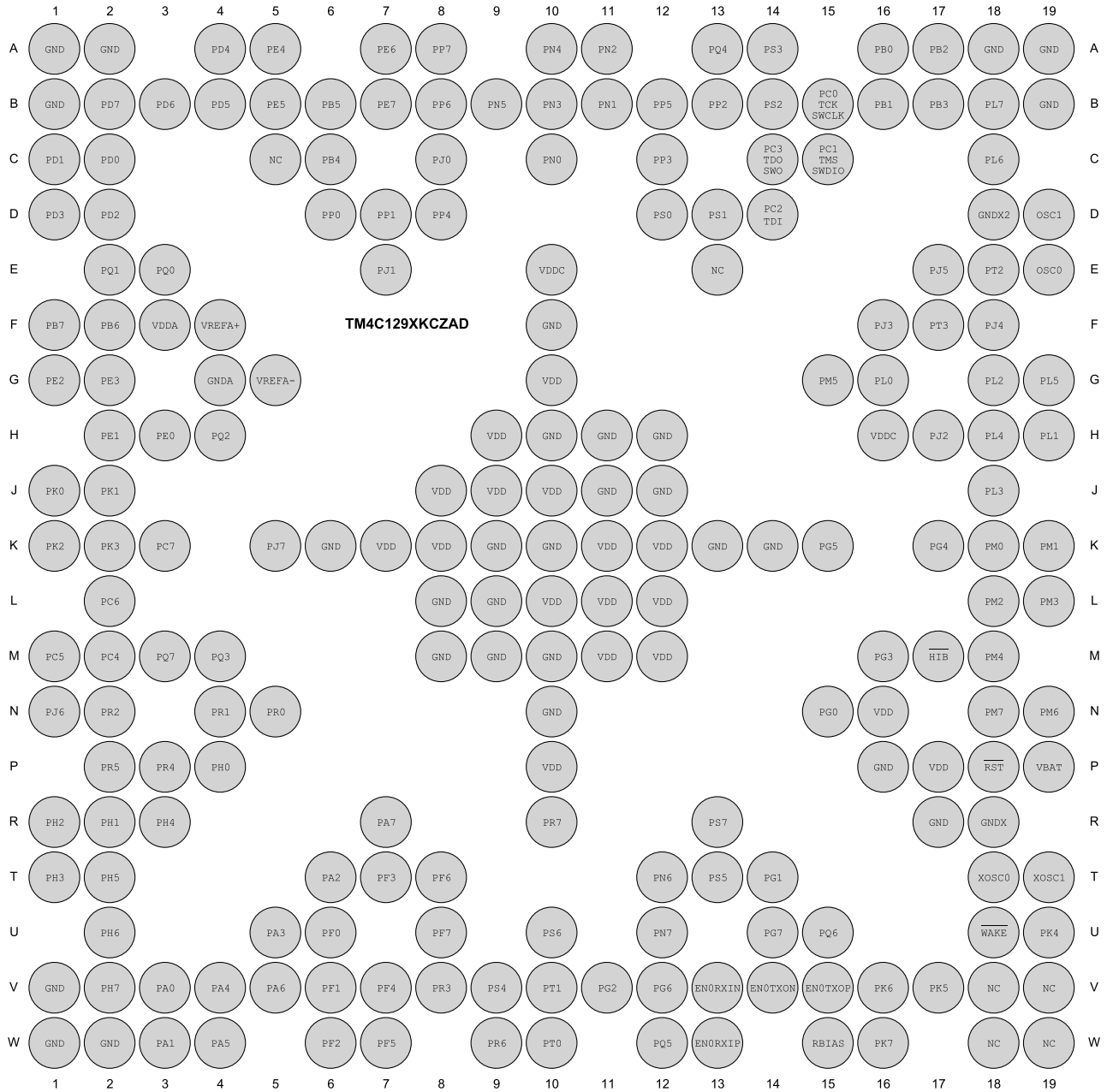
This bit is cleared by writing a 1. Clearing this bit also clears the INTINDEX bit in the **QEIRIS** register.

# 30 Pin Diagram

The TM4C129XKCZAD microcontroller pin diagram is shown below.

Each GPIO signal is identified by its GPIO port unless it defaults to an alternate function on reset. In this case, the GPIO port name is followed by the default alternate function. To see a complete list of possible functions for each pin, see Table 31-5 on page 2081.

**Figure 30-1. 212-Ball BGA Package Pin Diagram (Top View)**



# 31 Signal Tables

The following tables list the signals available for each pin. Signals are configured as GPIOs on reset, except for those noted below. Use the **GPIOAMSEL** register (see page 819) to select analog mode. For a GPIO pin to be used for an alternate digital function, the corresponding bit in the **GPIOAFSEL** register (see page 803) must be set. Further pin muxing options are provided through the PMCx bit field in the **GPIOPCTL** register (see page 820), which selects one of several available peripheral functions for that GPIO.

**Important:** Table 10-1 on page 773 shows special consideration GPIO pins. Most GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0). Special consideration pins may be programed to a non-GPIO function or may have special commit controls out of reset. In addition, a Power-On-Reset ($\overline{\text{POR}}$) returns these GPIO to their original special consideration state.

**Table 31-1. GPIO Pins With Special Considerations**

| GPIO Pins | Default Reset State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL | GPIOCR |
|---|---|---|---|---|---|---|---|
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 | 0 |
| PD[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |
| PE[7] | GPIO[a] | 0 | 0 | 0 | 0 | 0x0 | 0 |

a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

Table 31-2 on page 2029 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Each possible alternate analog and digital function is listed for each pin.

Table 31-3 on page 2048 lists the signals in alphabetical order by signal name. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed. The "Pin Mux" column indicates the GPIO and the encoding needed in the PMCx bit field in the **GPIOPCTL** register.

Table 31-4 on page 2064 groups the signals by functionality, except for GPIOs. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed.

Table 31-5 on page 2081 lists the GPIO pins and their analog and digital alternate functions. The AINx analog signals go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding AMSEL bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 3.3-V tolerant and are connected directly to their circuitry (C0-, C0+, C1-, C1+, C2-, C2+, USB0VBUS, USB0ID). These signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. The digital signals are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the PMCx bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric enoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Table 31-6 on page 2086 lists the signals based on number of possible pin assignments. This table can be used to plan how to configure the pins for a particular functionality. Application Note AN01274 Configuring Tiva™ C Series Microcontrollers with Pin Multiplexing provides an overview of the pin muxing implementation, an explanation of how a system designer defines a pin configuration, and examples of the pin configuration process.

**Note:** All digital inputs are Schmitt triggered.

## 31.1 Signals by Pin Number

**Table 31-2. Signals by Pin Number**

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| A1 | GND | - | Power | Ground reference for logic and I/O pins. |
| A2 | GND | - | Power | Ground reference for logic and I/O pins. |
| A4 | PD4 | I/O | TTL | GPIO port D bit 4. |
| | AIN7 | I | Analog | Analog-to-digital converter input 7. |
| | SSI1XDAT2 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 2. |
| | T3CCP0 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 0. |
| | U2Rx | I | TTL | UART module 2 receive. |
| A5 | PE4 | I/O | TTL | GPIO port E bit 4. |
| | AIN9 | I | Analog | Analog-to-digital converter input 9. |
| | SSI1XDAT0 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 0 (SSI1TX in Legacy SSI Mode). |
| | U1RI | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| A7 | PE6 | I/O | TTL | GPIO port E bit 6. |
| | AIN20 | I | Analog | Analog-to-digital converter input 20. |
| | I2C9SCL | I/O | OD | $I^2C$ module 9 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | U0CTS | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| A8 | PP7 | I/O | TTL | GPIO port P bit 7. |
| | AIN22 | I | Analog | Analog-to-digital converter input 22. |
| | OWIRE | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |
| A10 | PN4 | I/O | TTL | GPIO port N bit 4. |
| | EPI0S34 | I/O | TTL | EPI module 0 signal 34. |
| | I2C2SDA | I/O | OD | $I^2C$ module 2 data. |
| | U1DTR | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| | U3RTS | O | TTL | UART module 3 Request to Send modem flow control output line. |
| A11 | PN2 | I/O | TTL | GPIO port N bit 2. |
| | EPI0S29 | I/O | TTL | EPI module 0 signal 29. |
| | U1DCD | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| | U2RTS | O | TTL | UART module 2 Request to Send modem flow control output line. |
| A13 | PQ4 | I/O | TTL | GPIO port Q bit 4. |
| | DIVSCLK | O | TTL | An optionally divided reference clock output based on a selected clock source. Note that this signal is not synchronized to the System Clock. |
| | U1Rx | I | TTL | UART module 1 receive. |
| A14 | PS3 | I/O | TTL | GPIO port S bit 3. |
| | LCDDATA23 | O | TTL | LCD Data Pin 23 output. |
| | M0FAULT3 | I | TTL | Motion Control Module 0 PWM Fault 3. |
| | T3CCP1 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 1. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| A16 | PB0 | I/O | TTL | GPIO port B bit 0. |
| | CAN1Rx | I | TTL | CAN module 1 receive. |
| | I2C5SCL | I/O | OD | $I^2C$ module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | T4CCP0 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 0. |
| | U1Rx | I | TTL | UART module 1 receive. |
| | USB0ID | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side). |
| A17 | PB2 | I/O | TTL | GPIO port B bit 2. |
| | EN0MDC | O | TTL | Ethernet 0 Management Data Clock. |
| | EPI0S27 | I/O | TTL | EPI module 0 signal 27. |
| | I2C0SCL | I/O | OD | $I^2C$ module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | T5CCP0 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 0. |
| | USB0STP | O | TTL | Asserted by the USB controller to signal the end of a USB transmit packet or register write operation. |
| A18 | GND | - | Power | Ground reference for logic and I/O pins. |
| A19 | GND | - | Power | Ground reference for logic and I/O pins. |
| B1 | GND | - | Power | Ground reference for logic and I/O pins. |
| B2 | PD7 | I/O | TTL | GPIO port D bit 7. |
| | AIN4 | I | Analog | Analog-to-digital converter input 4. |
| | NMI | I | TTL | Non-maskable interrupt. |
| | SSI2XDAT2 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 2. |
| | T4CCP1 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 1. |
| | U2CTS | I | TTL | UART module 2 Clear To Send modem flow control input signal. |
| | USB0PFLT | I | TTL | Optionally used in Host mode by an external power source to indicate an error state by that power source. |
| B3 | PD6 | I/O | TTL | GPIO port D bit 6. |
| | AIN5 | I | Analog | Analog-to-digital converter input 5. |
| | SSI2XDAT3 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 3. |
| | T4CCP0 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 0. |
| | U2RTS | O | TTL | UART module 2 Request to Send modem flow control output line. |
| | USB0EPEN | O | TTL | Optionally used in Host mode to control an external power source to supply power to the USB bus. |
| B4 | PD5 | I/O | TTL | GPIO port D bit 5. |
| | AIN6 | I | Analog | Analog-to-digital converter input 6. |
| | SSI1XDAT3 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 3. |
| | T3CCP1 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 1. |
| | U2Tx | O | TTL | UART module 2 transmit. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| B5 | PE5 | I/O | TTL | GPIO port E bit 5. |
| | AIN8 | I | Analog | Analog-to-digital converter input 8. |
| | SSI1XDAT1 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 1 (SSI1RX in Legacy SSI Mode). |
| B6 | PB5 | I/O | TTL | GPIO port B bit 5. |
| | AIN11 | I | Analog | Analog-to-digital converter input 11. |
| | I2C5SDA | I/O | OD | $I^2C$ module 5 data. |
| | SSI1Clk | I/O | TTL | SSI module 1 clock. |
| | U0RTS | O | TTL | UART module 0 Request to Send modem flow control output signal. |
| B7 | PE7 | I/O | TTL | GPIO port E bit 7. |
| | AIN21 | I | Analog | Analog-to-digital converter input 21. |
| | I2C9SDA | I/O | OD | $I^2C$ module 9 data. |
| | NMI | I | TTL | Non-maskable interrupt. |
| | U0RTS | O | TTL | UART module 0 Request to Send modem flow control output signal. |
| B8 | PP6 | I/O | TTL | GPIO port P bit 6. |
| | AIN23 | I | Analog | Analog-to-digital converter input 23. |
| | I2C2SDA | I/O | OD | $I^2C$ module 2 data. |
| | U1DCD | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| B9 | PN5 | I/O | TTL | GPIO port N bit 5. |
| | EPI0S35 | I/O | TTL | EPI module 0 signal 35. |
| | I2C2SCL | I/O | OD | $I^2C$ module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | U1RI | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| | U3CTS | I | TTL | UART module 3 Clear To Send modem flow control input signal. |
| B10 | PN3 | I/O | TTL | GPIO port N bit 3. |
| | EPI0S30 | I/O | TTL | EPI module 0 signal 30. |
| | U1DSR | I | TTL | UART module 1 Data Set Ready modem output control line. |
| | U2CTS | I | TTL | UART module 2 Clear To Send modem flow control input signal. |
| B11 | PN1 | I/O | TTL | GPIO port N bit 1. |
| | U1CTS | I | TTL | UART module 1 Clear To Send modem flow control input signal. |
| B12 | PP5 | I/O | TTL | GPIO port P bit 5. |
| | I2C2SCL | I/O | OD | $I^2C$ module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | OWALT | I/O | TTL | 1-Wire Optional 2nd signal to be used as output. |
| | U3CTS | I | TTL | UART module 3 Clear To Send modem flow control input signal. |
| | USB0D6 | I/O | TTL | USB data 6. |
| B13 | PP2 | I/O | TTL | GPIO port P bit 2. |
| | EPI0S29 | I/O | TTL | EPI module 0 signal 29. |
| | U0DTR | O | TTL | UART module 0 Data Terminal Ready modem status input signal. |
| | USB0NXT | O | TTL | Asserted by the external PHY to throttle all data types. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| B14 | PS2 | I/O | TTL | GPIO port S bit 2. |
| | LCDDATA22 | O | TTL | LCD Data Pin 22 output. |
| | M0FAULT2 | I | TTL | Motion Control Module 0 PWM Fault 2. |
| | T3CCP0 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 0. |
| | U1DSR | I | TTL | UART module 1 Data Set Ready modem output control line. |
| B15 | PC0 | I/O | TTL | GPIO port C bit 0. |
| | SWCLK | I | TTL | JTAG/SWD CLK. |
| | TCK | I | TTL | JTAG/SWD CLK. |
| B16 | PB1 | I/O | TTL | GPIO port B bit 1. |
| | CAN1Tx | O | TTL | CAN module 1 transmit. |
| | I2C5SDA | I/O | OD | $I^2C$ module 5 data. |
| | T4CCP1 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 1. |
| | U1Tx | O | TTL | UART module 1 transmit. |
| | USB0VBUS | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |
| B17 | PB3 | I/O | TTL | GPIO port B bit 3. |
| | EN0MDIO | I/O | TTL | Ethernet 0 Management Data Input/Output signal. |
| | EPI0S28 | I/O | TTL | EPI module 0 signal 28. |
| | I2C0SDA | I/O | OD | $I^2C$ module 0 data. |
| | T5CCP1 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 1. |
| | USB0CLK | O | TTL | 60-MHz clock to the external PHY. |
| B18 | PL7 | I/O | TTL | GPIO port L bit 7. |
| | T1CCP1 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 1. |
| | USB0DM | I/O | Analog | Bidirectional differential data pin (D- per USB specification) for USB0. |
| B19 | GND | - | Power | Ground reference for logic and I/O pins. |
| C1 | PD1 | I/O | TTL | GPIO port D bit 1. |
| | AIN14 | I | Analog | Analog-to-digital converter input 14. |
| | C1o | O | TTL | Analog comparator 1 output. |
| | I2C7SDA | I/O | OD | $I^2C$ module 7 data. |
| | SSI2XDAT0 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 0 (SSI2TX in Legacy SSI Mode). |
| | T0CCP1 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 1. |
| C2 | PD0 | I/O | TTL | GPIO port D bit 0. |
| | AIN15 | I | Analog | Analog-to-digital converter input 15. |
| | C0o | O | TTL | Analog comparator 0 output. |
| | I2C7SCL | I/O | OD | $I^2C$ module 7 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | SSI2XDAT1 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 1 (SSI2RX in Legacy SSI Mode). |
| | T0CCP0 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 0. |
| C5 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| C6 | PB4 | I/O | TTL | GPIO port B bit 4. |
| | AIN10 | I | Analog | Analog-to-digital converter input 10. |
| | I2C5SCL | I/O | OD | $I^2C$ module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame signal. |
| | U0CTS | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| C8 | PJ0 | I/O | TTL | GPIO port J bit 0. |
| | EN0PPS | O | TTL | Ethernet 0 Pulse-Per-Second (PPS) Output. |
| | U3Rx | I | TTL | UART module 3 receive. |
| C10 | PN0 | I/O | TTL | GPIO port N bit 0. |
| | U1RTS | O | TTL | UART module 1 Request to Send modem flow control output line. |
| C12 | PP3 | I/O | TTL | GPIO port P bit 3. |
| | EPI0S30 | I/O | TTL | EPI module 0 signal 30. |
| | RTCCLK | O | TTL | Buffered version of the Hibernation module's 32.768-kHz clock. This signal is not output when the part is in Hibernate mode and before being configured after power-on reset. |
| | U0DCD | I | TTL | UART module 0 Data Carrier Detect modem status input signal. |
| | U1CTS | I | TTL | UART module 1 Clear To Send modem flow control input signal. |
| | USB0DIR | O | TTL | Indicates that the external PHY is able to accept data from the USB controller. |
| C14 | PC3 | I/O | TTL | GPIO port C bit 3. |
| | SWO | O | TTL | JTAG TDO and SWO. |
| | TDO | O | TTL | JTAG TDO and SWO. |
| C15 | PC1 | I/O | TTL | GPIO port C bit 1. |
| | SWDIO | I/O | TTL | JTAG TMS and SWDIO. |
| | TMS | I | TTL | JTAG TMS and SWDIO. |
| C18 | PL6 | I/O | TTL | GPIO port L bit 6. |
| | T1CCP0 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 0. |
| | USB0DP | I/O | Analog | Bidirectional differential data pin (D+ per USB specification) for USB0. |
| D1 | PD3 | I/O | TTL | GPIO port D bit 3. |
| | AIN12 | I | Analog | Analog-to-digital converter input 12. |
| | I2C8SDA | I/O | OD | $I^2C$ module 8 data. |
| | SSI2Clk | I/O | TTL | SSI module 2 clock. |
| | T1CCP1 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 1. |
| D2 | PD2 | I/O | TTL | GPIO port D bit 2. |
| | AIN13 | I | Analog | Analog-to-digital converter input 13. |
| | C2o | O | TTL | Analog comparator 2 output. |
| | I2C8SCL | I/O | OD | $I^2C$ module 8 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | SSI2Fss | I/O | TTL | SSI module 2 frame signal. |
| | T1CCP0 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 0. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| D6 | PP0 | I/O | TTL | GPIO port P bit 0. |
| | C2+ | I | Analog | Analog comparator 2 positive input. |
| | EN0INTRN | I | TTL | Ethernet 0 Interrupt from the Ethernet PHY. |
| | SSI3XDAT2 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 2. |
| | T6CCP0 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 0. |
| | U6Rx | I | TTL | UART module 6 receive. |
| D7 | PP1 | I/O | TTL | GPIO port P bit 1. |
| | C2- | I | Analog | Analog comparator 2 negative input. |
| | SSI3XDAT3 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 3. |
| | T6CCP1 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 1. |
| | U6Tx | O | TTL | UART module 6 transmit. |
| D8 | PP4 | I/O | TTL | GPIO port P bit 4. |
| | OWIRE | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |
| | U0DSR | I | TTL | UART module 0 Data Set Ready modem output control line. |
| | U3RTS | O | TTL | UART module 3 Request to Send modem flow control output line. |
| | USB0D7 | I/O | TTL | USB data 7. |
| D12 | PS0 | I/O | TTL | GPIO port S bit 0. |
| | LCDDATA20 | O | TTL | LCD Data Pin 20 output. |
| | M0FAULT0 | I | TTL | Motion Control Module 0 PWM Fault 0. |
| | T2CCP0 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 0. |
| D13 | PS1 | I/O | TTL | GPIO port S bit 1. |
| | LCDDATA21 | O | TTL | LCD Data Pin 21 output. |
| | M0FAULT1 | I | TTL | Motion Control Module 0 PWM Fault 1. |
| | T2CCP1 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 1. |
| D14 | PC2 | I/O | TTL | GPIO port C bit 2. |
| | TDI | I | TTL | JTAG TDI. |
| D18 | GNDX2 | - | Power | GND for the MOSC. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| D19 | OSC1 | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| E2 | PQ1 | I/O | TTL | GPIO port Q bit 1. |
| | EPI0S21 | I/O | TTL | EPI module 0 signal 21. |
| | SSI3Fss | I/O | TTL | SSI module 3 frame signal. |
| | T6CCP1 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 1. |
| E3 | PQ0 | I/O | TTL | GPIO port Q bit 0. |
| | EPI0S20 | I/O | TTL | EPI module 0 signal 20. |
| | SSI3Clk | I/O | TTL | SSI module 3 clock. |
| | T6CCP0 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 0. |
| E7 | PJ1 | I/O | TTL | GPIO port J bit 1. |
| | U3Tx | O | TTL | UART module 3 transmit. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| E10 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 32-15 on page 2111 . |
| E13 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| E17 | PJ5 | I/O | TTL | GPIO port J bit 5. |
| | LCDDATA17 | O | TTL | LCD Data Pin 17 output. |
| | U3CTS | I | TTL | UART module 3 Clear To Send modem flow control input signal. |
| E18 | PT2 | I/O | TTL | GPIO port T bit 2. |
| | CAN1Rx | I | TTL | CAN module 1 receive. |
| | LCDDATA18 | O | TTL | LCD Data Pin 18 output. |
| | T7CCP0 | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 0. |
| E19 | OSC0 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| F1 | PB7 | I/O | TTL | GPIO port B bit 7. |
| | I2C6SDA | I/O | OD | $I^2C$ module 6 data. |
| | T6CCP1 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 1. |
| F2 | PB6 | I/O | TTL | GPIO port B bit 6. |
| | I2C6SCL | I/O | OD | $I^2C$ module 6 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | T6CCP0 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 0. |
| F3 | VDDA | - | Power | The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in , regardless of system implementation. |
| F4 | VREFA+ | - | Analog | A reference voltage used to specify the voltage at which the ADC converts to a maximum value. This pin is used in conjunction with VREFA-, which specifies the minimum value . The voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA+ voltage is limited to the range specified in Table 32-44 on page 2138. |
| F10 | GND | - | Power | Ground reference for logic and I/O pins. |
| F16 | PJ3 | I/O | TTL | GPIO port J bit 3. |
| | LCDDATA15 | I/O | TTL | LCD Data Pin 15 input/output. |
| | U2CTS | I | TTL | UART module 2 Clear To Send modem flow control input signal. |
| F17 | PT3 | I/O | TTL | GPIO port T bit 3. |
| | CAN1Tx | O | TTL | CAN module 1 transmit. |
| | LCDDATA19 | O | TTL | LCD Data Pin 19 output. |
| | T7CCP1 | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 1. |
| F18 | PJ4 | I/O | TTL | GPIO port J bit 4. |
| | LCDDATA16 | O | TTL | LCD Data Pin 16 output. |
| | U3RTS | O | TTL | UART module 3 Request to Send modem flow control output line. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| G1 | PE2 | I/O | TTL | GPIO port E bit 2. |
| | AIN1 | I | Analog | Analog-to-digital converter input 1. |
| | U1DCD | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| G2 | PE3 | I/O | TTL | GPIO port E bit 3. |
| | AIN0 | I | Analog | Analog-to-digital converter input 0. |
| | OWIRE | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |
| | U1DTR | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| G4 | GNDA | - | Power | The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| G5 | VREFA- | - | Analog | A reference voltage used to specify the input voltage at which the ADC converts to a minimum value. This pin is used in conjunction with VREFA+, which specifies the maximum value. In other words, the voltage that is applied to VREFA- is the voltage with which an AINn signal is converted to 0, while the voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA- voltage is limited to the range specified in Table 32-44 on page 2138. |
| G10 | VDD | - | Power | Positive supply for I/O and some logic. |
| G15 | PM5 | I/O | TTL | GPIO port M bit 5. |
| | T4CCP1 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 1. |
| | TMPR2 | I/O | TTL | Tamper signal 2. |
| | U0DCD | I | TTL | UART module 0 Data Carrier Detect modem status input signal. |
| G16 | PL0 | I/O | TTL | GPIO port L bit 0. |
| | EPI0S16 | I/O | TTL | EPI module 0 signal 16. |
| | I2C2SDA | I/O | OD | $I^2C$ module 2 data. |
| | M0FAULT3 | I | TTL | Motion Control Module 0 PWM Fault 3. |
| | USB0D0 | I/O | TTL | USB data 0. |
| G18 | PL2 | I/O | TTL | GPIO port L bit 2. |
| | C0o | O | TTL | Analog comparator 0 output. |
| | EPI0S18 | I/O | TTL | EPI module 0 signal 18. |
| | PhB0 | I | TTL | QEI module 0 phase B. |
| | USB0D2 | I/O | TTL | USB data 2. |
| G19 | PL5 | I/O | TTL | GPIO port L bit 5. |
| | EPI0S33 | I/O | TTL | EPI module 0 signal 33. |
| | T0CCP1 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 1. |
| | USB0D5 | I/O | TTL | USB data 5. |
| H2 | PE1 | I/O | TTL | GPIO port E bit 1. |
| | AIN2 | I | Analog | Analog-to-digital converter input 2. |
| | U1DSR | I | TTL | UART module 1 Data Set Ready modem output control line. |
| H3 | PE0 | I/O | TTL | GPIO port E bit 0. |
| | AIN3 | I | Analog | Analog-to-digital converter input 3. |
| | U1RTS | O | TTL | UART module 1 Request to Send modem flow control output line. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| H4 | PQ2 | I/O | TTL | GPIO port Q bit 2. |
| | EPI0S22 | I/O | TTL | EPI module 0 signal 22. |
| | SSI3XDAT0 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 0 (SSI3TX in Legacy SSI Mode). |
| | T7CCP0 | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 0. |
| H9 | VDD | - | Power | Positive supply for I/O and some logic. |
| H10 | GND | - | Power | Ground reference for logic and I/O pins. |
| H11 | GND | - | Power | Ground reference for logic and I/O pins. |
| H12 | GND | - | Power | Ground reference for logic and I/O pins. |
| H16 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 32-15 on page 2111 . |
| H17 | PJ2 | I/O | TTL | GPIO port J bit 2. |
| | LCDDATA14 | I/O | TTL | LCD Data Pin 14 input/output. |
| | U2RTS | O | TTL | UART module 2 Request to Send modem flow control output line. |
| H18 | PL4 | I/O | TTL | GPIO port L bit 4. |
| | EPI0S26 | I/O | TTL | EPI module 0 signal 26. |
| | T0CCP0 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 0. |
| | USB0D4 | I/O | TTL | USB data 4. |
| H19 | PL1 | I/O | TTL | GPIO port L bit 1. |
| | EPI0S17 | I/O | TTL | EPI module 0 signal 17. |
| | I2C2SCL | I/O | OD | $I^2C$ module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | PhA0 | I | TTL | QEI module 0 phase A. |
| | USB0D1 | I/O | TTL | USB data 1. |
| J1 | PK0 | I/O | TTL | GPIO port K bit 0. |
| | AIN16 | I | Analog | Analog-to-digital converter input 16. |
| | EPI0S0 | I/O | TTL | EPI module 0 signal 0. |
| | U4Rx | I | TTL | UART module 4 receive. |
| J2 | PK1 | I/O | TTL | GPIO port K bit 1. |
| | AIN17 | I | Analog | Analog-to-digital converter input 17. |
| | EPI0S1 | I/O | TTL | EPI module 0 signal 1. |
| | U4Tx | O | TTL | UART module 4 transmit. |
| J8 | VDD | - | Power | Positive supply for I/O and some logic. |
| J9 | VDD | - | Power | Positive supply for I/O and some logic. |
| J10 | VDD | - | Power | Positive supply for I/O and some logic. |
| J11 | GND | - | Power | Ground reference for logic and I/O pins. |
| J12 | GND | - | Power | Ground reference for logic and I/O pins. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| J18 | PL3 | I/O | TTL | GPIO port L bit 3. |
| | C1o | O | TTL | Analog comparator 1 output. |
| | EPI0S19 | I/O | TTL | EPI module 0 signal 19. |
| | IDX0 | I | TTL | QEI module 0 index. |
| | USB0D3 | I/O | TTL | USB data 3. |
| K1 | PK2 | I/O | TTL | GPIO port K bit 2. |
| | AIN18 | I | Analog | Analog-to-digital converter input 18. |
| | EPI0S2 | I/O | TTL | EPI module 0 signal 2. |
| | U4RTS | O | TTL | UART module 4 Request to Send modem flow control output line. |
| K2 | PK3 | I/O | TTL | GPIO port K bit 3. |
| | AIN19 | I | Analog | Analog-to-digital converter input 19. |
| | EPI0S3 | I/O | TTL | EPI module 0 signal 3. |
| | U4CTS | I | TTL | UART module 4 Clear To Send modem flow control input signal. |
| K3 | PC7 | I/O | TTL | GPIO port C bit 7. |
| | C0- | I | Analog | Analog comparator 0 negative input. |
| | EPI0S4 | I/O | TTL | EPI module 0 signal 4. |
| | U5Tx | O | TTL | UART module 5 transmit. |
| K5 | PJ7 | I/O | TTL | GPIO port J bit 7. |
| | U4CTS | I | TTL | UART module 4 Clear To Send modem flow control input signal. |
| K6 | GND | - | Power | Ground reference for logic and I/O pins. |
| K7 | VDD | - | Power | Positive supply for I/O and some logic. |
| K8 | VDD | - | Power | Positive supply for I/O and some logic. |
| K9 | GND | - | Power | Ground reference for logic and I/O pins. |
| K10 | GND | - | Power | Ground reference for logic and I/O pins. |
| K11 | VDD | - | Power | Positive supply for I/O and some logic. |
| K12 | VDD | - | Power | Positive supply for I/O and some logic. |
| K13 | GND | - | Power | Ground reference for logic and I/O pins. |
| K14 | GND | - | Power | Ground reference for logic and I/O pins. |
| K15 | PG5 | I/O | TTL | GPIO port G bit 5. |
| | EN0TXD1 | O | TTL | Ethernet 0 Transmit Data 1. |
| | I2C3SDA | I/O | OD | $I^2C$ module 3 data. |
| | OWALT | I/O | TTL | 1-Wire Optional 2nd signal to be used as output. |
| | SSI2XDAT0 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 0 (SSI2TX in Legacy SSI Mode). |
| | U0RTS | O | TTL | UART module 0 Request to Send modem flow control output signal. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| K17 | PG4 | I/O | TTL | GPIO port G bit 4. |
| | EN0TXD0 | O | TTL | Ethernet 0 Transmit Data 0. |
| | I2C3SCL | I/O | OD | $I^2C$ module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | OWIRE | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |
| | SSI2XDAT1 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 1 (SSI2RX in Legacy SSI Mode). |
| | U0CTS | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| K18 | PM0 | I/O | TTL | GPIO port M bit 0. |
| | EPI0S15 | I/O | TTL | EPI module 0 signal 15. |
| | T2CCP0 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 0. |
| K19 | PM1 | I/O | TTL | GPIO port M bit 1. |
| | EPI0S14 | I/O | TTL | EPI module 0 signal 14. |
| | T2CCP1 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 1. |
| L2 | PC6 | I/O | TTL | GPIO port C bit 6. |
| | C0+ | I | Analog | Analog comparator 0 positive input. |
| | EPI0S5 | I/O | TTL | EPI module 0 signal 5. |
| | U5Rx | I | TTL | UART module 5 receive. |
| L8 | GND | - | Power | Ground reference for logic and I/O pins. |
| L9 | GND | - | Power | Ground reference for logic and I/O pins. |
| L10 | VDD | - | Power | Positive supply for I/O and some logic. |
| L11 | VDD | - | Power | Positive supply for I/O and some logic. |
| L12 | VDD | - | Power | Positive supply for I/O and some logic. |
| L18 | PM2 | I/O | TTL | GPIO port M bit 2. |
| | EPI0S13 | I/O | TTL | EPI module 0 signal 13. |
| | T3CCP0 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 0. |
| L19 | PM3 | I/O | TTL | GPIO port M bit 3. |
| | EPI0S12 | I/O | TTL | EPI module 0 signal 12. |
| | T3CCP1 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 1. |
| M1 | PC5 | I/O | TTL | GPIO port C bit 5. |
| | C1+ | I | Analog | Analog comparator 1 positive input. |
| | EPI0S6 | I/O | TTL | EPI module 0 signal 6. |
| | RTCCLK | O | TTL | Buffered version of the Hibernation module's 32.768-kHz clock. This signal is not output when the part is in Hibernate mode and before being configured after power-on reset. |
| | T7CCP1 | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 1. |
| | U7Tx | O | TTL | UART module 7 transmit. |
| M2 | PC4 | I/O | TTL | GPIO port C bit 4. |
| | C1- | I | Analog | Analog comparator 1 negative input. |
| | EPI0S7 | I/O | TTL | EPI module 0 signal 7. |
| | T7CCP0 | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 0. |
| | U7Rx | I | TTL | UART module 7 receive. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| M3 | PQ7 | I/O | TTL | GPIO port Q bit 7. |
| | U1RI | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| M4 | PQ3 | I/O | TTL | GPIO port Q bit 3. |
| | EPI0S23 | I/O | TTL | EPI module 0 signal 23. |
| | SSI3XDAT1 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 1 (SSI3RX in Legacy SSI Mode). |
| | T7CCP1 | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 1. |
| M8 | GND | - | Power | Ground reference for logic and I/O pins. |
| M9 | GND | - | Power | Ground reference for logic and I/O pins. |
| M10 | GND | - | Power | Ground reference for logic and I/O pins. |
| M11 | VDD | - | Power | Positive supply for I/O and some logic. |
| M12 | VDD | - | Power | Positive supply for I/O and some logic. |
| M16 | PG3 | I/O | TTL | GPIO port G bit 3. |
| | EN0TXEN | O | TTL | Ethernet 0 Transmit Enable. |
| | I2C2SDA | I/O | OD | $I^2C$ module 2 data. |
| | SSI2XDAT2 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 2. |
| M17 | $\overline{\text{HIB}}$ | O | TTL | An output that indicates the processor is in Hibernate mode. |
| M18 | PM4 | I/O | TTL | GPIO port M bit 4. |
| | EN0RREF_CLK | I/O | TTL | Ethernet 0 Reference Clock. |
| | T4CCP0 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 0. |
| | TMPR3 | I/O | TTL | Tamper signal 3. |
| | U0CTS | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| N1 | PJ6 | I/O | TTL | GPIO port J bit 6. |
| | LCDAC | O | TTL | LCD AC bias or latch enable in Raster mode; Primary chip select (CS0)/Primary Enable (E0) in LIDD MPU/Hitachi mode |
| | U4RTS | O | TTL | UART module 4 Request to Send modem flow control output line. |
| N2 | PR2 | I/O | TTL | GPIO port R bit 2. |
| | I2C2SCL | I/O | OD | $I^2C$ module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | LCDLP | O | TTL | LCD Line Clock or HSYNC in Raster mode; Write Strobe or Direction bit in LIDD mode |
| | M0PWM2 | O | TTL | Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1. |
| N4 | PR1 | I/O | TTL | GPIO port R bit 1. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| | LCDFP | O | TTL | LCD Frame Clock or VSYNC in Raster mode; Address Latch Enable in LIDD mode |
| | M0PWM1 | O | TTL | Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0. |
| | U4Rx | I | TTL | UART module 4 receive. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| N5 | PR0 | I/O | TTL | GPIO port R bit 0. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | LCDCP | O | TTL | LCD Pixel Clock in Raster mode; read strobe or read/write strobe in LIDD mode |
| | M0PWM0 | O | TTL | Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0. |
| | U4Tx | O | TTL | UART module 4 transmit. |
| N10 | GND | - | Power | Ground reference for logic and I/O pins. |
| N15 | PG0 | I/O | TTL | GPIO port G bit 0. |
| | EN0PPS | O | TTL | Ethernet 0 Pulse-Per-Second (PPS) Output. |
| | EPI0S11 | I/O | TTL | EPI module 0 signal 11. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | M0PWM4 | O | TTL | Motion Control Module 0 PWM 4. This signal is controlled by Module 0 PWM Generator 2. |
| N16 | VDD | - | Power | Positive supply for I/O and some logic. |
| N18 | PM7 | I/O | TTL | GPIO port M bit 7. |
| | EN0COL | I | TTL | Ethernet 0 Collision Detect. |
| | T5CCP1 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 1. |
| | TMPR0 | I/O | TTL | Tamper signal 0. |
| | U0RI | I | TTL | UART module 0 Ring Indicator modem status input signal. |
| N19 | PM6 | I/O | TTL | GPIO port M bit 6. |
| | EN0CRS | I | TTL | Ethernet 0 Carrier Sense. |
| | T5CCP0 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 0. |
| | TMPR1 | I/O | TTL | Tamper signal 1. |
| | U0DSR | I | TTL | UART module 0 Data Set Ready modem output control line. |
| P2 | PR5 | I/O | TTL | GPIO port R bit 5. |
| | I2C3SDA | I/O | OD | $I^2C$ module 3 data. |
| | LCDDATA01 | I/O | TTL | LCD Data Pin 1 input/output. |
| | M0PWM5 | O | TTL | Motion Control Module 0 PWM 5. This signal is controlled by Module 0 PWM Generator 2. |
| | T0CCP1 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 1. |
| | U1Rx | I | TTL | UART module 1 receive. |
| P3 | PR4 | I/O | TTL | GPIO port R bit 4. |
| | I2C3SCL | I/O | OD | $I^2C$ module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | LCDDATA00 | I/O | TTL | LCD Data Pin 0 input/output. |
| | M0PWM4 | O | TTL | Motion Control Module 0 PWM 4. This signal is controlled by Module 0 PWM Generator 2. |
| | T0CCP0 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 0. |
| P4 | PH0 | I/O | TTL | GPIO port H bit 0. |
| | EPI0S0 | I/O | TTL | EPI module 0 signal 0. |
| | U0RTS | O | TTL | UART module 0 Request to Send modem flow control output signal. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| P10 | VDD | - | Power | Positive supply for I/O and some logic. |
| P16 | GND | - | Power | Ground reference for logic and I/O pins. |
| P17 | VDD | - | Power | Positive supply for I/O and some logic. |
| P18 | $\overline{\text{RST}}$ | I | TTL | System reset input. |
| P19 | VBAT | - | Power | Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply. |
| R1 | PH2 | I/O | TTL | GPIO port H bit 2. |
| | EPI0S2 | I/O | TTL | EPI module 0 signal 2. |
| | U0DCD | I | TTL | UART module 0 Data Carrier Detect modem status input signal. |
| R2 | PH1 | I/O | TTL | GPIO port H bit 1. |
| | EPI0S1 | I/O | TTL | EPI module 0 signal 1. |
| | U0CTS | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| R3 | PH4 | I/O | TTL | GPIO port H bit 4. |
| | U0DTR | O | TTL | UART module 0 Data Terminal Ready modem status input signal. |
| R7 | PA7 | I/O | TTL | GPIO port A bit 7. |
| | EPI0S9 | I/O | TTL | EPI module 0 signal 9. |
| | I2C6SDA | I/O | OD | $I^2C$ module 6 data. |
| | SSI0XDAT3 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 3. |
| | T3CCP1 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 1. |
| | U2Tx | O | TTL | UART module 2 transmit. |
| | USB0EPEN | O | TTL | Optionally used in Host mode to control an external power source to supply power to the USB bus. |
| | USB0PFLT | I | TTL | Optionally used in Host mode by an external power source to indicate an error state by that power source. |
| R10 | PR7 | I/O | TTL | GPIO port R bit 7. |
| | EN0TXEN | O | TTL | Ethernet 0 Transmit Enable. |
| | I2C4SDA | I/O | OD | $I^2C$ module 4 data. |
| | LCDDATA05 | I/O | TTL | LCD Data Pin 5 input/output. |
| | M0PWM7 | O | TTL | Motion Control Module 0 PWM 7. This signal is controlled by Module 0 PWM Generator 3. |
| | T1CCP1 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 1. |
| R13 | PS7 | I/O | TTL | GPIO port S bit 7. |
| | EN0RXDV | I | TTL | Ethernet 0 Receive Data Valid. |
| | LCDDATA09 | I/O | TTL | LCD Data Pin 9 input/output. |
| | T5CCP1 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 1. |
| R17 | GND | - | Power | Ground reference for logic and I/O pins. |
| R18 | GNDX | - | Power | GND for the Hibernation oscillator. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| T1 | PH3 | I/O | TTL | GPIO port H bit 3. |
| | EPI0S3 | I/O | TTL | EPI module 0 signal 3. |
| | U0DSR | I | TTL | UART module 0 Data Set Ready modem output control line. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| T2 | PH5 | I/O | TTL | GPIO port H bit 5. |
| | EN0PPS | O | TTL | Ethernet 0 Pulse-Per-Second (PPS) Output. |
| | U0RI | I | TTL | UART module 0 Ring Indicator modem status input signal. |
| T6 | PA2 | I/O | TTL | GPIO port A bit 2. |
| | I2C8SCL | I/O | OD | $I^2C$ module 8 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | SSI0Clk | I/O | TTL | SSI module 0 clock |
| | T1CCP0 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 0. |
| | U4Rx | I | TTL | UART module 4 receive. |
| T7 | PF3 | I/O | TTL | GPIO port F bit 3. |
| | EN0MDIO | I/O | TTL | Ethernet 0 Management Data Input/Output signal. |
| | M0PWM3 | O | TTL | Motion Control Module 0 PWM 3. This signal is controlled by Module 0 PWM Generator 1. |
| | SSI3Clk | I/O | TTL | SSI module 3 clock. |
| | TRCLK | O | TTL | Trace clock. |
| T8 | PF6 | I/O | TTL | GPIO port F bit 6. |
| | LCDMCLK | O | TTL | LCD Memory Clock/Secondary chip Select (CS1)/Secondary Enable (E1) in LIDD Synchronous/Async MPU/Hitachi mode |
| T12 | PN6 | I/O | TTL | GPIO port N bit 6. |
| | EN0TXER | O | TTL | Ethernet 0 transmit error. |
| | LCDDATA13 | I/O | TTL | LCD Data Pin 13 input/output. |
| | U4RTS | O | TTL | UART module 4 Request to Send modem flow control output line. |
| T13 | PS5 | I/O | TTL | GPIO port S bit 5. |
| | EN0TXD1 | O | TTL | Ethernet 0 Transmit Data 1. |
| | LCDDATA07 | I/O | TTL | LCD Data Pin 7 input/output. |
| | PhB0 | I | TTL | QEI module 0 phase B. |
| | T4CCP1 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 1. |
| T14 | PG1 | I/O | TTL | GPIO port G bit 1. |
| | EPI0S10 | I/O | TTL | EPI module 0 signal 10. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| | M0PWM5 | O | TTL | Motion Control Module 0 PWM 5. This signal is controlled by Module 0 PWM Generator 2. |
| T18 | XOSC0 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a crystal or a 32.768-kHz oscillator for the Hibernation module RTC. |
| T19 | XOSC1 | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| U2 | PH6 | I/O | TTL | GPIO port H bit 6. |
| | U5Rx | I | TTL | UART module 5 receive. |
| | U7Rx | I | TTL | UART module 7 receive. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| U5 | PA3 | I/O | TTL | GPIO port A bit 3. |
| | I2C8SDA | I/O | OD | $I^2C$ module 8 data. |
| | SSI0Fss | I/O | TTL | SSI module 0 frame signal |
| | T1CCP1 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 1. |
| | U4Tx | O | TTL | UART module 4 transmit. |
| U6 | PF0 | I/O | TTL | GPIO port F bit 0. |
| | EN0LED0 | O | TTL | Ethernet 0 LED 0. |
| | M0PWM0 | O | TTL | Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0. |
| | SSI3XDAT1 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 1 (SSI3RX in Legacy SSI Mode). |
| | TRD2 | O | TTL | Trace data 2. |
| U8 | PF7 | I/O | TTL | GPIO port F bit 7. |
| | LCDDATA02 | I/O | TTL | LCD Data Pin 2 input/output. |
| U10 | PS6 | I/O | TTL | GPIO port S bit 6. |
| | EN0RXER | I | TTL | Ethernet 0 Receive Error. |
| | IDX0 | I | TTL | QEI module 0 index. |
| | LCDDATA08 | I/O | TTL | LCD Data Pin 8 input/output. |
| | T5CCP0 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 0. |
| U12 | PN7 | I/O | TTL | GPIO port N bit 7. |
| | LCDDATA12 | I/O | TTL | LCD Data Pin 12 input/output. |
| | U1RTS | O | TTL | UART module 1 Request to Send modem flow control output line. |
| | U4CTS | I | TTL | UART module 4 Clear To Send modem flow control input signal. |
| U14 | PG7 | I/O | TTL | GPIO port G bit 7. |
| | EN0RXDV | I | TTL | Ethernet 0 Receive Data Valid. |
| | I2C4SDA | I/O | OD | $I^2C$ module 4 data. |
| | OWIRE | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |
| | SSI2Clk | I/O | TTL | SSI module 2 clock. |
| U15 | PQ6 | I/O | TTL | GPIO port Q bit 6. |
| | EN0RXD1 | I | TTL | Ethernet 0 Receive Data 1. |
| | U1DTR | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| U18 | $\overline{\text{WAKE}}$ | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| U19 | PK4 | I/O | TTL | GPIO port K bit 4. |
| | EN0INTRN | I | TTL | Ethernet 0 Interrupt from the Ethernet PHY. |
| | EN0LED0 | O | TTL | Ethernet 0 LED 0. |
| | EN0RXD3 | I | TTL | Ethernet 0 Receive Data 3. |
| | EPI0S32 | I/O | TTL | EPI module 0 signal 32. |
| | I2C3SCL | I/O | OD | $I^2C$ module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | M0PWM6 | O | TTL | Motion Control Module 0 PWM 6. This signal is controlled by Module 0 PWM Generator 3. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| V1 | GND | - | Power | Ground reference for logic and I/O pins. |
| V2 | PH7 | I/O | TTL | GPIO port H bit 7. |
| | U5Tx | O | TTL | UART module 5 transmit. |
| | U7Tx | O | TTL | UART module 7 transmit. |
| V3 | PA0 | I/O | TTL | GPIO port A bit 0. |
| | CAN0Rx | I | TTL | CAN module 0 receive. |
| | I2C9SCL | I/O | OD | $I^2C$ module 9 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | T0CCP0 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 0. |
| | U0Rx | I | TTL | UART module 0 receive. |
| V4 | PA4 | I/O | TTL | GPIO port A bit 4. |
| | I2C7SCL | I/O | OD | $I^2C$ module 7 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | SSI0XDAT0 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 0 (SSI0TX in Legacy SSI Mode). |
| | T2CCP0 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 0. |
| | U3Rx | I | TTL | UART module 3 receive. |
| V5 | PA6 | I/O | TTL | GPIO port A bit 6. |
| | EN0RXCK | I | TTL | Ethernet 0 Receive Clock. |
| | EPI0S8 | I/O | TTL | EPI module 0 signal 8. |
| | I2C6SCL | I/O | OD | $I^2C$ module 6 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | SSI0XDAT2 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 2. |
| | T3CCP0 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 0. |
| | U2Rx | I | TTL | UART module 2 receive. |
| | USB0EPEN | O | TTL | Optionally used in Host mode to control an external power source to supply power to the USB bus. |
| V6 | PF1 | I/O | TTL | GPIO port F bit 1. |
| | EN0LED2 | O | TTL | Ethernet 0 LED 2. |
| | M0PWM1 | O | TTL | Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0. |
| | SSI3XDAT0 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 0 (SSI3TX in Legacy SSI Mode). |
| | TRD1 | O | TTL | Trace data 1. |
| V7 | PF4 | I/O | TTL | GPIO port F bit 4. |
| | EN0LED1 | O | TTL | Ethernet 0 LED 1. |
| | M0FAULT0 | I | TTL | Motion Control Module 0 PWM Fault 0. |
| | SSI3XDAT2 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 2. |
| | TRD3 | O | TTL | Trace data 3. |
| V8 | PR3 | I/O | TTL | GPIO port R bit 3. |
| | I2C2SDA | I/O | OD | $I^2C$ module 2 data. |
| | LCDDATA03 | I/O | TTL | LCD Data Pin 3 input/output. |
| | M0PWM3 | O | TTL | Motion Control Module 0 PWM 3. This signal is controlled by Module 0 PWM Generator 1. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| V9 | PS4 | I/O | TTL | GPIO port S bit 4. |
| | EN0TXD0 | O | TTL | Ethernet 0 Transmit Data 0. |
| | LCDDATA06 | I/O | TTL | LCD Data Pin 6 input/output. |
| | PhA0 | I | TTL | QEI module 0 phase A. |
| | T4CCP0 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 0. |
| V10 | PT1 | I/O | TTL | GPIO port T bit 1. |
| | CAN0Tx | O | TTL | CAN module 0 transmit. |
| | EN0RXD1 | I | TTL | Ethernet 0 Receive Data 1. |
| | LCDDATA11 | I/O | TTL | LCD Data Pin 11 input/output. |
| | T6CCP1 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 1. |
| V11 | PG2 | I/O | TTL | GPIO port G bit 2. |
| | EN0TXCK | I | TTL | Ethernet 0 Transmit Clock. |
| | I2C2SCL | I/O | OD | $I^2C$ module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | SSI2XDAT3 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 3. |
| V12 | PG6 | I/O | TTL | GPIO port G bit 6. |
| | EN0RXER | I | TTL | Ethernet 0 Receive Error. |
| | I2C4SCL | I/O | OD | $I^2C$ module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | OWIRE | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |
| | SSI2Fss | I/O | TTL | SSI module 2 frame signal. |
| V13 | EN0RXIN | I/O | TTL | Ethernet PHY negative receive differential input. |
| V14 | EN0TXON | I/O | TTL | Ethernet PHY negative transmit differential output. |
| V15 | EN0TXOP | I/O | TTL | Ethernet PHY positive transmit differential output. |
| V16 | PK6 | I/O | TTL | GPIO port K bit 6. |
| | EN0LED1 | O | TTL | Ethernet 0 LED 1. |
| | EN0TXD2 | O | TTL | Ethernet 0 Transmit Data 2. |
| | EPI0S25 | I/O | TTL | EPI module 0 signal 25. |
| | I2C4SCL | I/O | OD | $I^2C$ module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | M0FAULT1 | I | TTL | Motion Control Module 0 PWM Fault 1. |
| V17 | PK5 | I/O | TTL | GPIO port K bit 5. |
| | EN0LED2 | O | TTL | Ethernet 0 LED 2. |
| | EN0RXD2 | I | TTL | Ethernet 0 Receive Data 2. |
| | EPI0S31 | I/O | TTL | EPI module 0 signal 31. |
| | I2C3SDA | I/O | OD | $I^2C$ module 3 data. |
| | M0PWM7 | O | TTL | Motion Control Module 0 PWM 7. This signal is controlled by Module 0 PWM Generator 3. |
| V18 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| V19 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| W1 | GND | - | Power | Ground reference for logic and I/O pins. |
| W2 | GND | - | Power | Ground reference for logic and I/O pins. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| W3 | PA1 | I/O | TTL | GPIO port A bit 1. |
| | CAN0Tx | O | TTL | CAN module 0 transmit. |
| | I2C9SDA | I/O | OD | $I^2C$ module 9 data. |
| | T0CCP1 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 1. |
| | U0Tx | O | TTL | UART module 0 transmit. |
| W4 | PA5 | I/O | TTL | GPIO port A bit 5. |
| | I2C7SDA | I/O | OD | $I^2C$ module 7 data. |
| | SSI0XDAT1 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 1 (SSI0RX in Legacy SSI Mode). |
| | T2CCP1 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 1. |
| | U3Tx | O | TTL | UART module 3 transmit. |
| W6 | PF2 | I/O | TTL | GPIO port F bit 2. |
| | EN0MDC | O | TTL | Ethernet 0 Management Data Clock. |
| | M0PWM2 | O | TTL | Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1. |
| | SSI3Fss | I/O | TTL | SSI module 3 frame signal. |
| | TRD0 | O | TTL | Trace data 0. |
| W7 | PF5 | I/O | TTL | GPIO port F bit 5. |
| | SSI3XDAT3 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 3. |
| W9 | PR6 | I/O | TTL | GPIO port R bit 6. |
| | I2C4SCL | I/O | OD | $I^2C$ module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | LCDDATA04 | I/O | TTL | LCD Data Pin 4 input/output. |
| | M0PWM6 | O | TTL | Motion Control Module 0 PWM 6. This signal is controlled by Module 0 PWM Generator 3. |
| | T1CCP0 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 0. |
| | U1Tx | O | TTL | UART module 1 transmit. |
| W10 | PT0 | I/O | TTL | GPIO port T bit 0. |
| | CAN0Rx | I | TTL | CAN module 0 receive. |
| | EN0RXD0 | I | TTL | Ethernet 0 Receive Data 0. |
| | LCDDATA10 | I/O | TTL | LCD Data Pin 10 input/output. |
| | T6CCP0 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 0. |
| W12 | PQ5 | I/O | TTL | GPIO port Q bit 5. |
| | EN0RXD0 | I | TTL | Ethernet 0 Receive Data 0. |
| | U1Tx | O | TTL | UART module 1 transmit. |
| W13 | EN0RXIP | I/O | TTL | Ethernet PHY positive receive differential input. |
| W15 | RBIAS | O | Analog | 4.87-kΩ resistor (1% precision) for Ethernet PHY. |

**Table 31-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| W16 | PK7 | I/O | TTL | GPIO port K bit 7. |
| | EN0TXD3 | O | TTL | Ethernet 0 Transmit Data 3. |
| | EPI0S24 | I/O | TTL | EPI module 0 signal 24. |
| | I2C4SDA | I/O | OD | $I^2C$ module 4 data. |
| | M0FAULT2 | I | TTL | Motion Control Module 0 PWM Fault 2. |
| | RTCCLK | O | TTL | Buffered version of the Hibernation module's 32.768-kHz clock. This signal is not output when the part is in Hibernate mode and before being configured after power-on reset. |
| | U0RI | I | TTL | UART module 0 Ring Indicator modem status input signal. |
| W18 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| W19 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |

## 31.2 Signals by Signal Name

**Table 31-3. Signals by Signal Name**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| AIN0 | G2 | PE3 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | G1 | PE2 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | H2 | PE1 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | H3 | PE0 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | B2 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | B3 | PD6 | I | Analog | Analog-to-digital converter input 5. |
| AIN6 | B4 | PD5 | I | Analog | Analog-to-digital converter input 6. |
| AIN7 | A4 | PD4 | I | Analog | Analog-to-digital converter input 7. |
| AIN8 | B5 | PE5 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | A5 | PE4 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | C6 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | B6 | PB5 | I | Analog | Analog-to-digital converter input 11. |
| AIN12 | D1 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | D2 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | C1 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | C2 | PD0 | I | Analog | Analog-to-digital converter input 15. |
| AIN16 | J1 | PK0 | I | Analog | Analog-to-digital converter input 16. |
| AIN17 | J2 | PK1 | I | Analog | Analog-to-digital converter input 17. |
| AIN18 | K1 | PK2 | I | Analog | Analog-to-digital converter input 18. |
| AIN19 | K2 | PK3 | I | Analog | Analog-to-digital converter input 19. |
| AIN20 | A7 | PE6 | I | Analog | Analog-to-digital converter input 20. |
| AIN21 | B7 | PE7 | I | Analog | Analog-to-digital converter input 21. |
| AIN22 | A8 | PP7 | I | Analog | Analog-to-digital converter input 22. |
| AIN23 | B8 | PP6 | I | Analog | Analog-to-digital converter input 23. |
| C0+ | L2 | PC6 | I | Analog | Analog comparator 0 positive input. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| C0- | K3 | PC7 | I | Analog | Analog comparator 0 negative input. |
| C0o | C2<br>G18 | PD0 (5)<br>PL2 (5) | O | TTL | Analog comparator 0 output. |
| C1+ | M1 | PC5 | I | Analog | Analog comparator 1 positive input. |
| C1- | M2 | PC4 | I | Analog | Analog comparator 1 negative input. |
| C1o | C1<br>J18 | PD1 (5)<br>PL3 (5) | O | TTL | Analog comparator 1 output. |
| C2+ | D6 | PP0 | I | Analog | Analog comparator 2 positive input. |
| C2- | D7 | PP1 | I | Analog | Analog comparator 2 negative input. |
| C2o | D2 | PD2 (5) | O | TTL | Analog comparator 2 output. |
| CAN0Rx | V3<br>W10 | PA0 (7)<br>PT0 (7) | I | TTL | CAN module 0 receive. |
| CAN0Tx | W3<br>V10 | PA1 (7)<br>PT1 (7) | O | TTL | CAN module 0 transmit. |
| CAN1Rx | A16<br>E18 | PB0 (7)<br>PT2 (7) | I | TTL | CAN module 1 receive. |
| CAN1Tx | B16<br>F17 | PB1 (7)<br>PT3 (7) | O | TTL | CAN module 1 transmit. |
| DIVSCLK | A13 | PQ4 (7) | O | TTL | An optionally divided reference clock output based on a selected clock source. Note that this signal is not synchronized to the System Clock. |
| EN0COL | N18 | PM7 (14) | I | TTL | Ethernet 0 Collision Detect. |
| EN0CRS | N19 | PM6 (14) | I | TTL | Ethernet 0 Carrier Sense. |
| EN0INTRN | U19<br>D6 | PK4 (7)<br>PP0 (7) | I | TTL | Ethernet 0 Interrupt from the Ethernet PHY. |
| EN0LED0 | U6<br>U19 | PF0 (5)<br>PK4 (5) | O | TTL | Ethernet 0 LED 0. |
| EN0LED1 | V7<br>V16 | PF4 (5)<br>PK6 (5) | O | TTL | Ethernet 0 LED 1. |
| EN0LED2 | V6<br>V17 | PF1 (5)<br>PK5 (5) | O | TTL | Ethernet 0 LED 2. |
| EN0MDC | A17<br>W6 | PB2 (5)<br>PF2 (5) | O | TTL | Ethernet 0 Management Data Clock. |
| EN0MDIO | B17<br>T7 | PB3 (5)<br>PF3 (5) | I/O | TTL | Ethernet 0 Management Data Input/Output signal. |
| EN0PPS | N15<br>T2<br>C8 | PG0 (5)<br>PH5 (5)<br>PJ0 (5) | O | TTL | Ethernet 0 Pulse-Per-Second (PPS) Output. |
| EN0RREF_CLK | M18 | PM4 (14) | I/O | TTL | Ethernet 0 Reference Clock. |
| EN0RXCK | V5 | PA6 (14) | I | TTL | Ethernet 0 Receive Clock. |
| EN0RXD0 | W12<br>W10 | PQ5 (14)<br>PT0 (14) | I | TTL | Ethernet 0 Receive Data 0. |
| EN0RXD1 | U15<br>V10 | PQ6 (14)<br>PT1 (14) | I | TTL | Ethernet 0 Receive Data 1. |
| EN0RXD2 | V17 | PK5 (14) | I | TTL | Ethernet 0 Receive Data 2. |
| EN0RXD3 | U19 | PK4 (14) | I | TTL | Ethernet 0 Receive Data 3. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| EN0RXDV | U14<br>R13 | PG7 (14)<br>PS7 (14) | I | TTL | Ethernet 0 Receive Data Valid. |
| EN0RXER | V12<br>U10 | PG6 (14)<br>PS6 (14) | I | TTL | Ethernet 0 Receive Error. |
| EN0RXIN | V13 | fixed | I/O | TTL | Ethernet PHY negative receive differential input. |
| EN0RXIP | W13 | fixed | I/O | TTL | Ethernet PHY positive receive differential input. |
| EN0TXCK | V11 | PG2 (14) | I | TTL | Ethernet 0 Transmit Clock. |
| EN0TXD0 | K17<br>V9 | PG4 (14)<br>PS4 (14) | O | TTL | Ethernet 0 Transmit Data 0. |
| EN0TXD1 | K15<br>T13 | PG5 (14)<br>PS5 (14) | O | TTL | Ethernet 0 Transmit Data 1. |
| EN0TXD2 | V16 | PK6 (14) | O | TTL | Ethernet 0 Transmit Data 2. |
| EN0TXD3 | W16 | PK7 (14) | O | TTL | Ethernet 0 Transmit Data 3. |
| EN0TXEN | M16<br>R10 | PG3 (14)<br>PR7 (14) | O | TTL | Ethernet 0 Transmit Enable. |
| EN0TXER | T12 | PN6 (14) | O | TTL | Ethernet 0 transmit error. |
| EN0TXON | V14 | fixed | I/O | TTL | Ethernet PHY negative transmit differential output. |
| EN0TXOP | V15 | fixed | I/O | TTL | Ethernet PHY positive transmit differential output. |
| EPI0S0 | P4<br>J1 | PH0 (15)<br>PK0 (15) | I/O | TTL | EPI module 0 signal 0. |
| EPI0S1 | R2<br>J2 | PH1 (15)<br>PK1 (15) | I/O | TTL | EPI module 0 signal 1. |
| EPI0S2 | R1<br>K1 | PH2 (15)<br>PK2 (15) | I/O | TTL | EPI module 0 signal 2. |
| EPI0S3 | T1<br>K2 | PH3 (15)<br>PK3 (15) | I/O | TTL | EPI module 0 signal 3. |
| EPI0S4 | K3 | PC7 (15) | I/O | TTL | EPI module 0 signal 4. |
| EPI0S5 | L2 | PC6 (15) | I/O | TTL | EPI module 0 signal 5. |
| EPI0S6 | M1 | PC5 (15) | I/O | TTL | EPI module 0 signal 6. |
| EPI0S7 | M2 | PC4 (15) | I/O | TTL | EPI module 0 signal 7. |
| EPI0S8 | V5 | PA6 (15) | I/O | TTL | EPI module 0 signal 8. |
| EPI0S9 | R7 | PA7 (15) | I/O | TTL | EPI module 0 signal 9. |
| EPI0S10 | T14 | PG1 (15) | I/O | TTL | EPI module 0 signal 10. |
| EPI0S11 | N15 | PG0 (15) | I/O | TTL | EPI module 0 signal 11. |
| EPI0S12 | L19 | PM3 (15) | I/O | TTL | EPI module 0 signal 12. |
| EPI0S13 | L18 | PM2 (15) | I/O | TTL | EPI module 0 signal 13. |
| EPI0S14 | K19 | PM1 (15) | I/O | TTL | EPI module 0 signal 14. |
| EPI0S15 | K18 | PM0 (15) | I/O | TTL | EPI module 0 signal 15. |
| EPI0S16 | G16 | PL0 (15) | I/O | TTL | EPI module 0 signal 16. |
| EPI0S17 | H19 | PL1 (15) | I/O | TTL | EPI module 0 signal 17. |
| EPI0S18 | G18 | PL2 (15) | I/O | TTL | EPI module 0 signal 18. |
| EPI0S19 | J18 | PL3 (15) | I/O | TTL | EPI module 0 signal 19. |
| EPI0S20 | E3 | PQ0 (15) | I/O | TTL | EPI module 0 signal 20. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| EPI0S21 | E2 | PQ1 (15) | I/O | TTL | EPI module 0 signal 21. |
| EPI0S22 | H4 | PQ2 (15) | I/O | TTL | EPI module 0 signal 22. |
| EPI0S23 | M4 | PQ3 (15) | I/O | TTL | EPI module 0 signal 23. |
| EPI0S24 | W16 | PK7 (15) | I/O | TTL | EPI module 0 signal 24. |
| EPI0S25 | V16 | PK6 (15) | I/O | TTL | EPI module 0 signal 25. |
| EPI0S26 | H18 | PL4 (15) | I/O | TTL | EPI module 0 signal 26. |
| EPI0S27 | A17 | PB2 (15) | I/O | TTL | EPI module 0 signal 27. |
| EPI0S28 | B17 | PB3 (15) | I/O | TTL | EPI module 0 signal 28. |
| EPI0S29 | A11<br>B13 | PN2 (15)<br>PP2 (15) | I/O | TTL | EPI module 0 signal 29. |
| EPI0S30 | B10<br>C12 | PN3 (15)<br>PP3 (15) | I/O | TTL | EPI module 0 signal 30. |
| EPI0S31 | V17 | PK5 (15) | I/O | TTL | EPI module 0 signal 31. |
| EPI0S32 | U19 | PK4 (15) | I/O | TTL | EPI module 0 signal 32. |
| EPI0S33 | G19 | PL5 (15) | I/O | TTL | EPI module 0 signal 33. |
| EPI0S34 | A10 | PN4 (15) | I/O | TTL | EPI module 0 signal 34. |
| EPI0S35 | B9 | PN5 (15) | I/O | TTL | EPI module 0 signal 35. |
| GND | F10<br>H10<br>H11<br>H12<br>J11<br>J12<br>K6<br>K9<br>P16<br>K10<br>R17<br>K13<br>K14<br>L8<br>L9<br>M8<br>M9<br>M10<br>N10<br>A1<br>A2<br>B1<br>V1<br>W1<br>W2<br>A18<br>A19<br>B19 | fixed | - | Power | Ground reference for logic and I/O pins. |
| GNDA | G4 | fixed | - | Power | The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| GNDX | R18 | fixed | - | Power | GND for the Hibernation oscillator. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| GNDX2 | D18 | fixed | - | Power | GND for the MOSC. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| HIB | M17 | fixed | O | TTL | An output that indicates the processor is in Hibernate mode. |
| I2C0SCL | A17 | PB2 (2) | I/O | OD | $I^2C$ module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C0SDA | B17 | PB3 (2) | I/O | OD | $I^2C$ module 0 data. |
| I2C1SCL | N15<br>N5 | PG0 (2)<br>PR0 (2) | I/O | OD | $I^2C$ module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C1SDA | T14<br>N4 | PG1 (2)<br>PR1 (2) | I/O | OD | $I^2C$ module 1 data. |
| I2C2SCL | V11<br>H19<br>B9<br>B12<br>N2 | PG2 (2)<br>PL1 (2)<br>PN5 (3)<br>PP5 (2)<br>PR2 (2) | I/O | OD | $I^2C$ module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C2SDA | M16<br>G16<br>A10<br>B8<br>V8 | PG3 (2)<br>PL0 (2)<br>PN4 (3)<br>PP6 (2)<br>PR3 (2) | I/O | OD | $I^2C$ module 2 data. |
| I2C3SCL | K17<br>U19<br>P3 | PG4 (2)<br>PK4 (2)<br>PR4 (2) | I/O | OD | $I^2C$ module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C3SDA | K15<br>V17<br>P2 | PG5 (2)<br>PK5 (2)<br>PR5 (2) | I/O | OD | $I^2C$ module 3 data. |
| I2C4SCL | V12<br>V16<br>W9 | PG6 (2)<br>PK6 (2)<br>PR6 (2) | I/O | OD | $I^2C$ module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C4SDA | U14<br>W16<br>R10 | PG7 (2)<br>PK7 (2)<br>PR7 (2) | I/O | OD | $I^2C$ module 4 data. |
| I2C5SCL | A16<br>C6 | PB0 (2)<br>PB4 (2) | I/O | OD | $I^2C$ module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C5SDA | B16<br>B6 | PB1 (2)<br>PB5 (2) | I/O | OD | $I^2C$ module 5 data. |
| I2C6SCL | V5<br>F2 | PA6 (2)<br>PB6 (2) | I/O | OD | $I^2C$ module 6 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| I2C6SDA | R7<br>F1 | PA7 (2)<br>PB7 (2) | I/O | OD | $I^2C$ module 6 data. |
| I2C7SCL | V4<br>C2 | PA4 (2)<br>PD0 (2) | I/O | OD | $I^2C$ module 7 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C7SDA | W4<br>C1 | PA5 (2)<br>PD1 (2) | I/O | OD | $I^2C$ module 7 data. |
| I2C8SCL | T6<br>D2 | PA2 (2)<br>PD2 (2) | I/O | OD | $I^2C$ module 8 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C8SDA | U5<br>D1 | PA3 (2)<br>PD3 (2) | I/O | OD | $I^2C$ module 8 data. |
| I2C9SCL | V3<br>A7 | PA0 (2)<br>PE6 (2) | I/O | OD | $I^2C$ module 9 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| I2C9SDA | W3<br>B7 | PA1 (2)<br>PE7 (2) | I/O | OD | $I^2C$ module 9 data. |
| IDX0 | J18<br>U10 | PL3 (6)<br>PS6 (6) | I | TTL | QEI module 0 index. |
| LCDAC | N1 | PJ6 (15) | O | TTL | LCD AC bias or latch enable in Raster mode; Primary chip select (CS0)/Primary Enable (E0) in LIDD MPU/Hitachi mode |
| LCDCP | N5 | PR0 (15) | O | TTL | LCD Pixel Clock in Raster mode; read strobe or read/write strobe in LIDD mode |
| LCDDATA00 | P3 | PR4 (15) | I/O | TTL | LCD Data Pin 0 input/output. |
| LCDDATA01 | P2 | PR5 (15) | I/O | TTL | LCD Data Pin 1 input/output. |
| LCDDATA02 | U8 | PF7 (15) | I/O | TTL | LCD Data Pin 2 input/output. |
| LCDDATA03 | V8 | PR3 (15) | I/O | TTL | LCD Data Pin 3 input/output. |
| LCDDATA04 | W9 | PR6 (15) | I/O | TTL | LCD Data Pin 4 input/output. |
| LCDDATA05 | R10 | PR7 (15) | I/O | TTL | LCD Data Pin 5 input/output. |
| LCDDATA06 | V9 | PS4 (15) | I/O | TTL | LCD Data Pin 6 input/output. |
| LCDDATA07 | T13 | PS5 (15) | I/O | TTL | LCD Data Pin 7 input/output. |
| LCDDATA08 | U10 | PS6 (15) | I/O | TTL | LCD Data Pin 8 input/output. |
| LCDDATA09 | R13 | PS7 (15) | I/O | TTL | LCD Data Pin 9 input/output. |
| LCDDATA10 | W10 | PT0 (15) | I/O | TTL | LCD Data Pin 10 input/output. |
| LCDDATA11 | V10 | PT1 (15) | I/O | TTL | LCD Data Pin 11 input/output. |
| LCDDATA12 | U12 | PN7 (15) | I/O | TTL | LCD Data Pin 12 input/output. |
| LCDDATA13 | T12 | PN6 (15) | I/O | TTL | LCD Data Pin 13 input/output. |
| LCDDATA14 | H17 | PJ2 (15) | I/O | TTL | LCD Data Pin 14 input/output. |
| LCDDATA15 | F16 | PJ3 (15) | I/O | TTL | LCD Data Pin 15 input/output. |
| LCDDATA16 | F18 | PJ4 (15) | O | TTL | LCD Data Pin 16 output. |
| LCDDATA17 | E17 | PJ5 (15) | O | TTL | LCD Data Pin 17 output. |
| LCDDATA18 | E18 | PT2 (15) | O | TTL | LCD Data Pin 18 output. |
| LCDDATA19 | F17 | PT3 (15) | O | TTL | LCD Data Pin 19 output. |
| LCDDATA20 | D12 | PS0 (15) | O | TTL | LCD Data Pin 20 output. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| LCDDATA21 | D13 | PS1 (15) | O | TTL | LCD Data Pin 21 output. |
| LCDDATA22 | B14 | PS2 (15) | O | TTL | LCD Data Pin 22 output. |
| LCDDATA23 | A14 | PS3 (15) | O | TTL | LCD Data Pin 23 output. |
| LCDFP | N4 | PR1 (15) | O | TTL | LCD Frame Clock or VSYNC in Raster mode; Address Latch Enable in LIDD mode |
| LCDLP | N2 | PR2 (15) | O | TTL | LCD Line Clock or HSYNC in Raster mode; Write Strobe or Direction bit in LIDD mode |
| LCDMCLK | T8 | PF6 (15) | O | TTL | LCD Memory Clock/Secondary chip Select (CS1)/Secondary Enable (E1) in LIDD Synchronous/Async MPU/Hitachi mode |
| M0FAULT0 | V7<br>D12 | PF4 (6)<br>PS0 (6) | I | TTL | Motion Control Module 0 PWM Fault 0. |
| M0FAULT1 | V16<br>D13 | PK6 (6)<br>PS1 (6) | I | TTL | Motion Control Module 0 PWM Fault 1. |
| M0FAULT2 | W16<br>B14 | PK7 (6)<br>PS2 (6) | I | TTL | Motion Control Module 0 PWM Fault 2. |
| M0FAULT3 | G16<br>A14 | PL0 (6)<br>PS3 (6) | I | TTL | Motion Control Module 0 PWM Fault 3. |
| M0PWM0 | U6<br>N5 | PF0 (6)<br>PR0 (6) | O | TTL | Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0. |
| M0PWM1 | V6<br>N4 | PF1 (6)<br>PR1 (6) | O | TTL | Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0. |
| M0PWM2 | W6<br>N2 | PF2 (6)<br>PR2 (6) | O | TTL | Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1. |
| M0PWM3 | T7<br>V8 | PF3 (6)<br>PR3 (6) | O | TTL | Motion Control Module 0 PWM 3. This signal is controlled by Module 0 PWM Generator 1. |
| M0PWM4 | N15<br>P3 | PG0 (6)<br>PR4 (6) | O | TTL | Motion Control Module 0 PWM 4. This signal is controlled by Module 0 PWM Generator 2. |
| M0PWM5 | T14<br>P2 | PG1 (6)<br>PR5 (6) | O | TTL | Motion Control Module 0 PWM 5. This signal is controlled by Module 0 PWM Generator 2. |
| M0PWM6 | U19<br>W9 | PK4 (6)<br>PR6 (6) | O | TTL | Motion Control Module 0 PWM 6. This signal is controlled by Module 0 PWM Generator 3. |
| M0PWM7 | V17<br>R10 | PK5 (6)<br>PR7 (6) | O | TTL | Motion Control Module 0 PWM 7. This signal is controlled by Module 0 PWM Generator 3. |
| NC | C5<br>V18<br>V19<br>W18<br>W19<br>E13 | fixed | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| NMI | B2<br>B7 | PD7 (8)<br>PE7 (8) | I | TTL | Non-maskable interrupt. |
| OSC0 | E19 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | D19 | fixed | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| OWALT | K15<br>B12 | PG5 (5)<br>PP5 (4) | I/O | TTL | 1-Wire Optional 2nd signal to be used as output. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| OWIRE | G2<br>K17<br>V12<br>U14<br>D8<br>A8 | PE3 (5)<br>PG4 (5)<br>PG6 (5)<br>PG7 (5)<br>PP4 (4)<br>PP7 (5) | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |
| PA0 | V3 | - | I/O | TTL | GPIO port A bit 0. |
| PA1 | W3 | - | I/O | TTL | GPIO port A bit 1. |
| PA2 | T6 | - | I/O | TTL | GPIO port A bit 2. |
| PA3 | U5 | - | I/O | TTL | GPIO port A bit 3. |
| PA4 | V4 | - | I/O | TTL | GPIO port A bit 4. |
| PA5 | W4 | - | I/O | TTL | GPIO port A bit 5. |
| PA6 | V5 | - | I/O | TTL | GPIO port A bit 6. |
| PA7 | R7 | - | I/O | TTL | GPIO port A bit 7. |
| PB0 | A16 | - | I/O | TTL | GPIO port B bit 0. |
| PB1 | B16 | - | I/O | TTL | GPIO port B bit 1. |
| PB2 | A17 | - | I/O | TTL | GPIO port B bit 2. |
| PB3 | B17 | - | I/O | TTL | GPIO port B bit 3. |
| PB4 | C6 | - | I/O | TTL | GPIO port B bit 4. |
| PB5 | B6 | - | I/O | TTL | GPIO port B bit 5. |
| PB6 | F2 | - | I/O | TTL | GPIO port B bit 6. |
| PB7 | F1 | - | I/O | TTL | GPIO port B bit 7. |
| PC0 | B15 | - | I/O | TTL | GPIO port C bit 0. |
| PC1 | C15 | - | I/O | TTL | GPIO port C bit 1. |
| PC2 | D14 | - | I/O | TTL | GPIO port C bit 2. |
| PC3 | C14 | - | I/O | TTL | GPIO port C bit 3. |
| PC4 | M2 | - | I/O | TTL | GPIO port C bit 4. |
| PC5 | M1 | - | I/O | TTL | GPIO port C bit 5. |
| PC6 | L2 | - | I/O | TTL | GPIO port C bit 6. |
| PC7 | K3 | - | I/O | TTL | GPIO port C bit 7. |
| PD0 | C2 | - | I/O | TTL | GPIO port D bit 0. |
| PD1 | C1 | - | I/O | TTL | GPIO port D bit 1. |
| PD2 | D2 | - | I/O | TTL | GPIO port D bit 2. |
| PD3 | D1 | - | I/O | TTL | GPIO port D bit 3. |
| PD4 | A4 | - | I/O | TTL | GPIO port D bit 4. |
| PD5 | B4 | - | I/O | TTL | GPIO port D bit 5. |
| PD6 | B3 | - | I/O | TTL | GPIO port D bit 6. |
| PD7 | B2 | - | I/O | TTL | GPIO port D bit 7. |
| PE0 | H3 | - | I/O | TTL | GPIO port E bit 0. |
| PE1 | H2 | - | I/O | TTL | GPIO port E bit 1. |
| PE2 | G1 | - | I/O | TTL | GPIO port E bit 2. |
| PE3 | G2 | - | I/O | TTL | GPIO port E bit 3. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| PE4 | A5 | - | I/O | TTL | GPIO port E bit 4. |
| PE5 | B5 | - | I/O | TTL | GPIO port E bit 5. |
| PE6 | A7 | - | I/O | TTL | GPIO port E bit 6. |
| PE7 | B7 | - | I/O | TTL | GPIO port E bit 7. |
| PF0 | U6 | - | I/O | TTL | GPIO port F bit 0. |
| PF1 | V6 | - | I/O | TTL | GPIO port F bit 1. |
| PF2 | W6 | - | I/O | TTL | GPIO port F bit 2. |
| PF3 | T7 | - | I/O | TTL | GPIO port F bit 3. |
| PF4 | V7 | - | I/O | TTL | GPIO port F bit 4. |
| PF5 | W7 | - | I/O | TTL | GPIO port F bit 5. |
| PF6 | T8 | - | I/O | TTL | GPIO port F bit 6. |
| PF7 | U8 | - | I/O | TTL | GPIO port F bit 7. |
| PG0 | N15 | - | I/O | TTL | GPIO port G bit 0. |
| PG1 | T14 | - | I/O | TTL | GPIO port G bit 1. |
| PG2 | V11 | - | I/O | TTL | GPIO port G bit 2. |
| PG3 | M16 | - | I/O | TTL | GPIO port G bit 3. |
| PG4 | K17 | - | I/O | TTL | GPIO port G bit 4. |
| PG5 | K15 | - | I/O | TTL | GPIO port G bit 5. |
| PG6 | V12 | - | I/O | TTL | GPIO port G bit 6. |
| PG7 | U14 | - | I/O | TTL | GPIO port G bit 7. |
| PH0 | P4 | - | I/O | TTL | GPIO port H bit 0. |
| PH1 | R2 | - | I/O | TTL | GPIO port H bit 1. |
| PH2 | R1 | - | I/O | TTL | GPIO port H bit 2. |
| PH3 | T1 | - | I/O | TTL | GPIO port H bit 3. |
| PH4 | R3 | - | I/O | TTL | GPIO port H bit 4. |
| PH5 | T2 | - | I/O | TTL | GPIO port H bit 5. |
| PH6 | U2 | - | I/O | TTL | GPIO port H bit 6. |
| PH7 | V2 | - | I/O | TTL | GPIO port H bit 7. |
| PhA0 | H19<br>V9 | PL1 (6)<br>PS4 (6) | I | TTL | QEI module 0 phase A. |
| PhB0 | G18<br>T13 | PL2 (6)<br>PS5 (6) | I | TTL | QEI module 0 phase B. |
| PJ0 | C8 | - | I/O | TTL | GPIO port J bit 0. |
| PJ1 | E7 | - | I/O | TTL | GPIO port J bit 1. |
| PJ2 | H17 | - | I/O | TTL | GPIO port J bit 2. |
| PJ3 | F16 | - | I/O | TTL | GPIO port J bit 3. |
| PJ4 | F18 | - | I/O | TTL | GPIO port J bit 4. |
| PJ5 | E17 | - | I/O | TTL | GPIO port J bit 5. |
| PJ6 | N1 | - | I/O | TTL | GPIO port J bit 6. |
| PJ7 | K5 | - | I/O | TTL | GPIO port J bit 7. |
| PK0 | J1 | - | I/O | TTL | GPIO port K bit 0. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| PK1 | J2 | - | I/O | TTL | GPIO port K bit 1. |
| PK2 | K1 | - | I/O | TTL | GPIO port K bit 2. |
| PK3 | K2 | - | I/O | TTL | GPIO port K bit 3. |
| PK4 | U19 | - | I/O | TTL | GPIO port K bit 4. |
| PK5 | V17 | - | I/O | TTL | GPIO port K bit 5. |
| PK6 | V16 | - | I/O | TTL | GPIO port K bit 6. |
| PK7 | W16 | - | I/O | TTL | GPIO port K bit 7. |
| PL0 | G16 | - | I/O | TTL | GPIO port L bit 0. |
| PL1 | H19 | - | I/O | TTL | GPIO port L bit 1. |
| PL2 | G18 | - | I/O | TTL | GPIO port L bit 2. |
| PL3 | J18 | - | I/O | TTL | GPIO port L bit 3. |
| PL4 | H18 | - | I/O | TTL | GPIO port L bit 4. |
| PL5 | G19 | - | I/O | TTL | GPIO port L bit 5. |
| PL6 | C18 | - | I/O | TTL | GPIO port L bit 6. |
| PL7 | B18 | - | I/O | TTL | GPIO port L bit 7. |
| PM0 | K18 | - | I/O | TTL | GPIO port M bit 0. |
| PM1 | K19 | - | I/O | TTL | GPIO port M bit 1. |
| PM2 | L18 | - | I/O | TTL | GPIO port M bit 2. |
| PM3 | L19 | - | I/O | TTL | GPIO port M bit 3. |
| PM4 | M18 | - | I/O | TTL | GPIO port M bit 4. |
| PM5 | G15 | - | I/O | TTL | GPIO port M bit 5. |
| PM6 | N19 | - | I/O | TTL | GPIO port M bit 6. |
| PM7 | N18 | - | I/O | TTL | GPIO port M bit 7. |
| PN0 | C10 | - | I/O | TTL | GPIO port N bit 0. |
| PN1 | B11 | - | I/O | TTL | GPIO port N bit 1. |
| PN2 | A11 | - | I/O | TTL | GPIO port N bit 2. |
| PN3 | B10 | - | I/O | TTL | GPIO port N bit 3. |
| PN4 | A10 | - | I/O | TTL | GPIO port N bit 4. |
| PN5 | B9 | - | I/O | TTL | GPIO port N bit 5. |
| PN6 | T12 | - | I/O | TTL | GPIO port N bit 6. |
| PN7 | U12 | - | I/O | TTL | GPIO port N bit 7. |
| PP0 | D6 | - | I/O | TTL | GPIO port P bit 0. |
| PP1 | D7 | - | I/O | TTL | GPIO port P bit 1. |
| PP2 | B13 | - | I/O | TTL | GPIO port P bit 2. |
| PP3 | C12 | - | I/O | TTL | GPIO port P bit 3. |
| PP4 | D8 | - | I/O | TTL | GPIO port P bit 4. |
| PP5 | B12 | - | I/O | TTL | GPIO port P bit 5. |
| PP6 | B8 | - | I/O | TTL | GPIO port P bit 6. |
| PP7 | A8 | - | I/O | TTL | GPIO port P bit 7. |
| PQ0 | E3 | - | I/O | TTL | GPIO port Q bit 0. |
| PQ1 | E2 | - | I/O | TTL | GPIO port Q bit 1. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| PQ2 | H4 | - | I/O | TTL | GPIO port Q bit 2. |
| PQ3 | M4 | - | I/O | TTL | GPIO port Q bit 3. |
| PQ4 | A13 | - | I/O | TTL | GPIO port Q bit 4. |
| PQ5 | W12 | - | I/O | TTL | GPIO port Q bit 5. |
| PQ6 | U15 | - | I/O | TTL | GPIO port Q bit 6. |
| PQ7 | M3 | - | I/O | TTL | GPIO port Q bit 7. |
| PR0 | N5 | - | I/O | TTL | GPIO port R bit 0. |
| PR1 | N4 | - | I/O | TTL | GPIO port R bit 1. |
| PR2 | N2 | - | I/O | TTL | GPIO port R bit 2. |
| PR3 | V8 | - | I/O | TTL | GPIO port R bit 3. |
| PR4 | P3 | - | I/O | TTL | GPIO port R bit 4. |
| PR5 | P2 | - | I/O | TTL | GPIO port R bit 5. |
| PR6 | W9 | - | I/O | TTL | GPIO port R bit 6. |
| PR7 | R10 | - | I/O | TTL | GPIO port R bit 7. |
| PS0 | D12 | - | I/O | TTL | GPIO port S bit 0. |
| PS1 | D13 | - | I/O | TTL | GPIO port S bit 1. |
| PS2 | B14 | - | I/O | TTL | GPIO port S bit 2. |
| PS3 | A14 | - | I/O | TTL | GPIO port S bit 3. |
| PS4 | V9 | - | I/O | TTL | GPIO port S bit 4. |
| PS5 | T13 | - | I/O | TTL | GPIO port S bit 5. |
| PS6 | U10 | - | I/O | TTL | GPIO port S bit 6. |
| PS7 | R13 | - | I/O | TTL | GPIO port S bit 7. |
| PT0 | W10 | - | I/O | TTL | GPIO port T bit 0. |
| PT1 | V10 | - | I/O | TTL | GPIO port T bit 1. |
| PT2 | E18 | - | I/O | TTL | GPIO port T bit 2. |
| PT3 | F17 | - | I/O | TTL | GPIO port T bit 3. |
| RBIAS | W15 | fixed | O | Analog | 4.87-kΩ resistor (1% precision) for Ethernet PHY. |
| $\overline{\text{RST}}$ | P18 | fixed | I | TTL | System reset input. |
| RTCCLK | M1<br>W16<br>C12 | PC5 (7)<br>PK7 (5)<br>PP3 (7) | O | TTL | Buffered version of the Hibernation module's 32.768-kHz clock. This signal is not output when the part is in Hibernate mode and before being configured after power-on reset. |
| SSI0Clk | T6 | PA2 (15) | I/O | TTL | SSI module 0 clock |
| SSI0Fss | U5 | PA3 (15) | I/O | TTL | SSI module 0 frame signal |
| SSI0XDAT0 | V4 | PA4 (15) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 0 (SSI0TX in Legacy SSI Mode). |
| SSI0XDAT1 | W4 | PA5 (15) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 1 (SSI0RX in Legacy SSI Mode). |
| SSI0XDAT2 | V5 | PA6 (13) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 2. |
| SSI0XDAT3 | R7 | PA7 (13) | I/O | TTL | SSI Module 0 Bi-directional Data Pin 3. |
| SSI1Clk | B6 | PB5 (15) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | C6 | PB4 (15) | I/O | TTL | SSI module 1 frame signal. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| SSI1XDAT0 | A5 | PE4 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 0 (SSI1TX in Legacy SSI Mode). |
| SSI1XDAT1 | B5 | PE5 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 1 (SSI1RX in Legacy SSI Mode). |
| SSI1XDAT2 | A4 | PD4 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 2. |
| SSI1XDAT3 | B4 | PD5 (15) | I/O | TTL | SSI Module 1 Bi-directional Data Pin 3. |
| SSI2Clk | D1<br>U14 | PD3 (15)<br>PG7 (15) | I/O | TTL | SSI module 2 clock. |
| SSI2Fss | D2<br>V12 | PD2 (15)<br>PG6 (15) | I/O | TTL | SSI module 2 frame signal. |
| SSI2XDAT0 | C1<br>K15 | PD1 (15)<br>PG5 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 0 (SSI2TX in Legacy SSI Mode). |
| SSI2XDAT1 | C2<br>K17 | PD0 (15)<br>PG4 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 1 (SSI2RX in Legacy SSI Mode). |
| SSI2XDAT2 | B2<br>M16 | PD7 (15)<br>PG3 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 2. |
| SSI2XDAT3 | B3<br>V11 | PD6 (15)<br>PG2 (15) | I/O | TTL | SSI Module 2 Bi-directional Data Pin 3. |
| SSI3Clk | T7<br>E3 | PF3 (14)<br>PQ0 (14) | I/O | TTL | SSI module 3 clock. |
| SSI3Fss | W6<br>E2 | PF2 (14)<br>PQ1 (14) | I/O | TTL | SSI module 3 frame signal. |
| SSI3XDAT0 | V6<br>H4 | PF1 (14)<br>PQ2 (14) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 0 (SSI3TX in Legacy SSI Mode). |
| SSI3XDAT1 | U6<br>M4 | PF0 (14)<br>PQ3 (14) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 1 (SSI3RX in Legacy SSI Mode). |
| SSI3XDAT2 | V7<br>D6 | PF4 (14)<br>PP0 (15) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 2. |
| SSI3XDAT3 | W7<br>D7 | PF5 (14)<br>PP1 (15) | I/O | TTL | SSI Module 3 Bi-directional Data Pin 3. |
| SWCLK | B15 | PC0 (1) | I | TTL | JTAG/SWD CLK. |
| SWDIO | C15 | PC1 (1) | I/O | TTL | JTAG TMS and SWDIO. |
| SWO | C14 | PC3 (1) | O | TTL | JTAG TDO and SWO. |
| T0CCP0 | V3<br>C2<br>H18<br>P3 | PA0 (3)<br>PD0 (3)<br>PL4 (3)<br>PR4 (3) | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 0. |
| T0CCP1 | W3<br>C1<br>G19<br>P2 | PA1 (3)<br>PD1 (3)<br>PL5 (3)<br>PR5 (3) | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 1. |
| T1CCP0 | T6<br>D2<br>C18<br>W9 | PA2 (3)<br>PD2 (3)<br>PL6 (3)<br>PR6 (3) | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 0. |
| T1CCP1 | U5<br>D1<br>B18<br>R10 | PA3 (3)<br>PD3 (3)<br>PL7 (3)<br>PR7 (3) | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 1. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| T2CCP0 | V4<br>K18<br>D12 | PA4 (3)<br>PM0 (3)<br>PS0 (3) | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 0. |
| T2CCP1 | W4<br>K19<br>D13 | PA5 (3)<br>PM1 (3)<br>PS1 (3) | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 1. |
| T3CCP0 | V5<br>A4<br>L18<br>B14 | PA6 (3)<br>PD4 (3)<br>PM2 (3)<br>PS2 (3) | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 0. |
| T3CCP1 | R7<br>B4<br>L19<br>A14 | PA7 (3)<br>PD5 (3)<br>PM3 (3)<br>PS3 (3) | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 1. |
| T4CCP0 | A16<br>B3<br>M18<br>V9 | PB0 (3)<br>PD6 (3)<br>PM4 (3)<br>PS4 (3) | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 0. |
| T4CCP1 | B16<br>B2<br>G15<br>T13 | PB1 (3)<br>PD7 (3)<br>PM5 (3)<br>PS5 (3) | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 1. |
| T5CCP0 | A17<br>N19<br>U10 | PB2 (3)<br>PM6 (3)<br>PS6 (3) | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 0. |
| T5CCP1 | B17<br>N18<br>R13 | PB3 (3)<br>PM7 (3)<br>PS7 (3) | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 1. |
| T6CCP0 | F2<br>D6<br>E3<br>W10 | PB6 (3)<br>PP0 (5)<br>PQ0 (3)<br>PT0 (3) | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 0. |
| T6CCP1 | F1<br>D7<br>E2<br>V10 | PB7 (3)<br>PP1 (5)<br>PQ1 (3)<br>PT1 (3) | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 1. |
| T7CCP0 | M2<br>H4<br>E18 | PC4 (3)<br>PQ2 (3)<br>PT2 (3) | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 0. |
| T7CCP1 | M1<br>M4<br>F17 | PC5 (3)<br>PQ3 (3)<br>PT3 (3) | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 1. |
| TCK | B15 | PC0 (1) | I | TTL | JTAG/SWD CLK. |
| TDI | D14 | PC2 (1) | I | TTL | JTAG TDI. |
| TDO | C14 | PC3 (1) | O | TTL | JTAG TDO and SWO. |
| TMPR0 | N18 | PM7 | I/O | TTL | Tamper signal 0. |
| TMPR1 | N19 | PM6 | I/O | TTL | Tamper signal 1. |
| TMPR2 | G15 | PM5 | I/O | TTL | Tamper signal 2. |
| TMPR3 | M18 | PM4 | I/O | TTL | Tamper signal 3. |
| TMS | C15 | PC1 (1) | I | TTL | JTAG TMS and SWDIO. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| TRCLK | T7 | PF3 (15) | O | TTL | Trace clock. |
| TRD0 | W6 | PF2 (15) | O | TTL | Trace data 0. |
| TRD1 | V6 | PF1 (15) | O | TTL | Trace data 1. |
| TRD2 | U6 | PF0 (15) | O | TTL | Trace data 2. |
| TRD3 | V7 | PF4 (15) | O | TTL | Trace data 3. |
| U0CTS | C6<br>A7<br>K17<br>R2<br>M18 | PB4 (1)<br>PE6 (1)<br>PG4 (1)<br>PH1 (1)<br>PM4 (1) | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| U0DCD | R1<br>G15<br>C12 | PH2 (1)<br>PM5 (1)<br>PP3 (2) | I | TTL | UART module 0 Data Carrier Detect modem status input signal. |
| U0DSR | T1<br>N19<br>D8 | PH3 (1)<br>PM6 (1)<br>PP4 (2) | I | TTL | UART module 0 Data Set Ready modem output control line. |
| U0DTR | R3<br>B13 | PH4 (1)<br>PP2 (1) | O | TTL | UART module 0 Data Terminal Ready modem status input signal. |
| U0RI | T2<br>W16<br>N18 | PH5 (1)<br>PK7 (1)<br>PM7 (1) | I | TTL | UART module 0 Ring Indicator modem status input signal. |
| U0RTS | B6<br>B7<br>K15<br>P4 | PB5 (1)<br>PE7 (1)<br>PG5 (1)<br>PH0 (1) | O | TTL | UART module 0 Request to Send modem flow control output signal. |
| U0Rx | V3 | PA0 (1) | I | TTL | UART module 0 receive. |
| U0Tx | W3 | PA1 (1) | O | TTL | UART module 0 transmit. |
| U1CTS | B11<br>C12 | PN1 (1)<br>PP3 (1) | I | TTL | UART module 1 Clear To Send modem flow control input signal. |
| U1DCD | G1<br>A11<br>B8 | PE2 (1)<br>PN2 (1)<br>PP6 (1) | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| U1DSR | H2<br>B10<br>B14 | PE1 (1)<br>PN3 (1)<br>PS2 (1) | I | TTL | UART module 1 Data Set Ready modem output control line. |
| U1DTR | G2<br>A10<br>U15 | PE3 (1)<br>PN4 (1)<br>PQ6 (1) | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| U1RI | A5<br>B9<br>M3 | PE4 (1)<br>PN5 (1)<br>PQ7 (1) | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| U1RTS | H3<br>C10<br>U12 | PE0 (1)<br>PN0 (1)<br>PN7 (1) | O | TTL | UART module 1 Request to Send modem flow control output line. |
| U1Rx | A16<br>A13<br>P2 | PB0 (1)<br>PQ4 (1)<br>PR5 (1) | I | TTL | UART module 1 receive. |
| U1Tx | B16<br>W12<br>W9 | PB1 (1)<br>PQ5 (1)<br>PR6 (1) | O | TTL | UART module 1 transmit. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| U2CTS | B2<br>F16<br>B10 | PD7 (1)<br>PJ3 (1)<br>PN3 (2) | I | TTL | UART module 2 Clear To Send modem flow control input signal. |
| U2RTS | B3<br>H17<br>A11 | PD6 (1)<br>PJ2 (1)<br>PN2 (2) | O | TTL | UART module 2 Request to Send modem flow control output line. |
| U2Rx | V5<br>A4 | PA6 (1)<br>PD4 (1) | I | TTL | UART module 2 receive. |
| U2Tx | R7<br>B4 | PA7 (1)<br>PD5 (1) | O | TTL | UART module 2 transmit. |
| U3CTS | E17<br>B9<br>B12 | PJ5 (1)<br>PN5 (2)<br>PP5 (1) | I | TTL | UART module 3 Clear To Send modem flow control input signal. |
| U3RTS | F18<br>A10<br>D8 | PJ4 (1)<br>PN4 (2)<br>PP4 (1) | O | TTL | UART module 3 Request to Send modem flow control output line. |
| U3Rx | V4<br>C8 | PA4 (1)<br>PJ0 (1) | I | TTL | UART module 3 receive. |
| U3Tx | W4<br>E7 | PA5 (1)<br>PJ1 (1) | O | TTL | UART module 3 transmit. |
| U4CTS | K5<br>K2<br>U12 | PJ7 (1)<br>PK3 (1)<br>PN7 (2) | I | TTL | UART module 4 Clear To Send modem flow control input signal. |
| U4RTS | N1<br>K1<br>T12 | PJ6 (1)<br>PK2 (1)<br>PN6 (2) | O | TTL | UART module 4 Request to Send modem flow control output line. |
| U4Rx | T6<br>J1<br>N4 | PA2 (1)<br>PK0 (1)<br>PR1 (1) | I | TTL | UART module 4 receive. |
| U4Tx | U5<br>J2<br>N5 | PA3 (1)<br>PK1 (1)<br>PR0 (1) | O | TTL | UART module 4 transmit. |
| U5Rx | L2<br>U2 | PC6 (1)<br>PH6 (1) | I | TTL | UART module 5 receive. |
| U5Tx | K3<br>V2 | PC7 (1)<br>PH7 (1) | O | TTL | UART module 5 transmit. |
| U6Rx | D6 | PP0 (1) | I | TTL | UART module 6 receive. |
| U6Tx | D7 | PP1 (1) | O | TTL | UART module 6 transmit. |
| U7Rx | M2<br>U2 | PC4 (1)<br>PH6 (2) | I | TTL | UART module 7 receive. |
| U7Tx | M1<br>V2 | PC5 (1)<br>PH7 (2) | O | TTL | UART module 7 transmit. |
| USB0CLK | B17 | PB3 (14) | O | TTL | 60-MHz clock to the external PHY. |
| USB0D0 | G16 | PL0 (14) | I/O | TTL | USB data 0. |
| USB0D1 | H19 | PL1 (14) | I/O | TTL | USB data 1. |
| USB0D2 | G18 | PL2 (14) | I/O | TTL | USB data 2. |
| USB0D3 | J18 | PL3 (14) | I/O | TTL | USB data 3. |
| USB0D4 | H18 | PL4 (14) | I/O | TTL | USB data 4. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| USB0D5 | G19 | PL5 (14) | I/O | TTL | USB data 5. |
| USB0D6 | B12 | PP5 (14) | I/O | TTL | USB data 6. |
| USB0D7 | D8 | PP4 (14) | I/O | TTL | USB data 7. |
| USB0DIR | C12 | PP3 (14) | O | TTL | Indicates that the external PHY is able to accept data from the USB controller. |
| USB0DM | B18 | PL7 | I/O | Analog | Bidirectional differential data pin (D- per USB specification) for USB0. |
| USB0DP | C18 | PL6 | I/O | Analog | Bidirectional differential data pin (D+ per USB specification) for USB0. |
| USB0EPEN | V5<br>R7<br>B3 | PA6 (5)<br>PA7 (11)<br>PD6 (5) | O | TTL | Optionally used in Host mode to control an external power source to supply power to the USB bus. |
| USB0ID | A16 | PB0 | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side). |
| USB0NXT | B13 | PP2 (14) | O | TTL | Asserted by the external PHY to throttle all data types. |
| USB0PFLT | R7<br>B2 | PA7 (5)<br>PD7 (5) | I | TTL | Optionally used in Host mode by an external power source to indicate an error state by that power source. |
| USB0STP | A17 | PB2 (14) | O | TTL | Asserted by the USB controller to signal the end of a USB transmit packet or register write operation. |
| USB0VBUS | B16 | PB1 | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |
| VBAT | P19 | fixed | - | Power | Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply. |
| VDD | G10<br>H9<br>J8<br>J9<br>J10<br>K7<br>K8<br>K11<br>N16<br>P17<br>K12<br>L10<br>L11<br>L12<br>M11<br>M12<br>P10 | fixed | - | Power | Positive supply for I/O and some logic. |

**Table 31-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| VDDA | F3 | fixed | - | Power | The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in , regardless of system implementation. |
| VDDC | H16<br>E10 | fixed | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 32-15 on page 2111 . |
| VREFA+ | F4 | fixed | - | Analog | A reference voltage used to specify the voltage at which the ADC converts to a maximum value. This pin is used in conjunction with VREFA-, which specifies the minimum value . The voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA+ voltage is limited to the range specified in Table 32-44 on page 2138. |
| VREFA- | G5 | fixed | - | Analog | A reference voltage used to specify the input voltage at which the ADC converts to a minimum value. This pin is used in conjunction with VREFA+, which specifies the maximum value. In other words, the voltage that is applied to VREFA- is the voltage with which an AINn signal is converted to 0, while the voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA- voltage is limited to the range specified in Table 32-44 on page 2138. |
| WAKE | U18 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| XOSC0 | T18 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a crystal or a 32.768-kHz oscillator for the Hibernation module RTC. |
| XOSC1 | T19 | fixed | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |

# 31.3 Signals by Function, Except for GPIO

**Table 31-4. Signals by Function, Except for GPIO**

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| 1-Wire | 0WALT | K15<br>B12 | I/O | TTL | 1-Wire Optional 2nd signal to be used as output. |
| | 0WIRE | G2<br>K17<br>V12<br>U14<br>D8<br>A8 | I/O | TTL | 1-Wire Single Bus Pin. This signal is input only if 1-Wire Alternate Output is enabled. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| ADC | AIN0 | G2 | I | Analog | Analog-to-digital converter input 0. |
| | AIN1 | G1 | I | Analog | Analog-to-digital converter input 1. |
| | AIN2 | H2 | I | Analog | Analog-to-digital converter input 2. |
| | AIN3 | H3 | I | Analog | Analog-to-digital converter input 3. |
| | AIN4 | B2 | I | Analog | Analog-to-digital converter input 4. |
| | AIN5 | B3 | I | Analog | Analog-to-digital converter input 5. |
| | AIN6 | B4 | I | Analog | Analog-to-digital converter input 6. |
| | AIN7 | A4 | I | Analog | Analog-to-digital converter input 7. |
| | AIN8 | B5 | I | Analog | Analog-to-digital converter input 8. |
| | AIN9 | A5 | I | Analog | Analog-to-digital converter input 9. |
| | AIN10 | C6 | I | Analog | Analog-to-digital converter input 10. |
| | AIN11 | B6 | I | Analog | Analog-to-digital converter input 11. |
| | AIN12 | D1 | I | Analog | Analog-to-digital converter input 12. |
| | AIN13 | D2 | I | Analog | Analog-to-digital converter input 13. |
| | AIN14 | C1 | I | Analog | Analog-to-digital converter input 14. |
| | AIN15 | C2 | I | Analog | Analog-to-digital converter input 15. |
| | AIN16 | J1 | I | Analog | Analog-to-digital converter input 16. |
| | AIN17 | J2 | I | Analog | Analog-to-digital converter input 17. |
| | AIN18 | K1 | I | Analog | Analog-to-digital converter input 18. |
| | AIN19 | K2 | I | Analog | Analog-to-digital converter input 19. |
| | AIN20 | A7 | I | Analog | Analog-to-digital converter input 20. |
| | AIN21 | B7 | I | Analog | Analog-to-digital converter input 21. |
| | AIN22 | A8 | I | Analog | Analog-to-digital converter input 22. |
| | AIN23 | B8 | I | Analog | Analog-to-digital converter input 23. |
| | VREFA+ | F4 | - | Analog | A reference voltage used to specify the voltage at which the ADC converts to a maximum value. This pin is used in conjunction with VREFA-, which specifies the minimum value . The voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA+ voltage is limited to the range specified in Table 32-44 on page 2138. |
| | VREFA- | G5 | - | Analog | A reference voltage used to specify the input voltage at which the ADC converts to a minimum value. This pin is used in conjunction with VREFA+, which specifies the maximum value. In other words, the voltage that is applied to VREFA- is the voltage with which an AINn signal is converted to 0, while the voltage that is applied to VREFA+ is the voltage with which an AINn signal is converted to 4095. The VREFA- voltage is limited to the range specified in Table 32-44 on page 2138. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| Analog Comparators | C0+ | L2 | I | Analog | Analog comparator 0 positive input. |
| | C0- | K3 | I | Analog | Analog comparator 0 negative input. |
| | C0o | C2<br>G18 | O | TTL | Analog comparator 0 output. |
| | C1+ | M1 | I | Analog | Analog comparator 1 positive input. |
| | C1- | M2 | I | Analog | Analog comparator 1 negative input. |
| | C1o | C1<br>J18 | O | TTL | Analog comparator 1 output. |
| | C2+ | D6 | I | Analog | Analog comparator 2 positive input. |
| | C2- | D7 | I | Analog | Analog comparator 2 negative input. |
| | C2o | D2 | O | TTL | Analog comparator 2 output. |
| Controller Area Network | CAN0Rx | V3<br>W10 | I | TTL | CAN module 0 receive. |
| | CAN0Tx | W3<br>V10 | O | TTL | CAN module 0 transmit. |
| | CAN1Rx | A16<br>E18 | I | TTL | CAN module 1 receive. |
| | CAN1Tx | B16<br>F17 | O | TTL | CAN module 1 transmit. |
| Core | TRCLK | T7 | O | TTL | Trace clock. |
| | TRD0 | W6 | O | TTL | Trace data 0. |
| | TRD1 | V6 | O | TTL | Trace data 1. |
| | TRD2 | U6 | O | TTL | Trace data 2. |
| | TRD3 | V7 | O | TTL | Trace data 3. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| Ethernet | EN0COL | N18 | I | TTL | Ethernet 0 Collision Detect. |
| | EN0CRS | N19 | I | TTL | Ethernet 0 Carrier Sense. |
| | EN0INTRN | U19<br>D6 | I | TTL | Ethernet 0 Interrupt from the Ethernet PHY. |
| | EN0LED0 | U6<br>U19 | O | TTL | Ethernet 0 LED 0. |
| | EN0LED1 | V7<br>V16 | O | TTL | Ethernet 0 LED 1. |
| | EN0LED2 | V6<br>V17 | O | TTL | Ethernet 0 LED 2. |
| | EN0MDC | A17<br>W6 | O | TTL | Ethernet 0 Management Data Clock. |
| | EN0MDIO | B17<br>T7 | I/O | TTL | Ethernet 0 Management Data Input/Output signal. |
| | EN0PPS | N15<br>T2<br>C8 | O | TTL | Ethernet 0 Pulse-Per-Second (PPS) Output. |
| | EN0RREF_CLK | M18 | I/O | TTL | Ethernet 0 Reference Clock. |
| | EN0RXCK | V5 | I | TTL | Ethernet 0 Receive Clock. |
| | EN0RXD0 | W12<br>W10 | I | TTL | Ethernet 0 Receive Data 0. |
| | EN0RXD1 | U15<br>V10 | I | TTL | Ethernet 0 Receive Data 1. |
| | EN0RXD2 | V17 | I | TTL | Ethernet 0 Receive Data 2. |
| | EN0RXD3 | U19 | I | TTL | Ethernet 0 Receive Data 3. |
| | EN0RXDV | U14<br>R13 | I | TTL | Ethernet 0 Receive Data Valid. |
| | EN0RXER | V12<br>U10 | I | TTL | Ethernet 0 Receive Error. |
| | EN0RXIN | V13 | I/O | TTL | Ethernet PHY negative receive differential input. |
| | EN0RXIP | W13 | I/O | TTL | Ethernet PHY positive receive differential input. |
| | EN0TXCK | V11 | I | TTL | Ethernet 0 Transmit Clock. |
| | EN0TXD0 | K17<br>V9 | O | TTL | Ethernet 0 Transmit Data 0. |
| | EN0TXD1 | K15<br>T13 | O | TTL | Ethernet 0 Transmit Data 1. |
| | EN0TXD2 | V16 | O | TTL | Ethernet 0 Transmit Data 2. |
| | EN0TXD3 | W16 | O | TTL | Ethernet 0 Transmit Data 3. |
| | EN0TXEN | M16<br>R10 | O | TTL | Ethernet 0 Transmit Enable. |
| | EN0TXER | T12 | O | TTL | Ethernet 0 transmit error. |
| | EN0TXON | V14 | I/O | TTL | Ethernet PHY negative transmit differential output. |
| | EN0TXOP | V15 | I/O | TTL | Ethernet PHY positive transmit differential output. |
| | RBIAS | W15 | O | Analog | 4.87-kΩ resistor (1% precision) for Ethernet PHY. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| External Peripheral Interface | EPI0S0 | P4<br>J1 | I/O | TTL | EPI module 0 signal 0. |
| | EPI0S1 | R2<br>J2 | I/O | TTL | EPI module 0 signal 1. |
| | EPI0S2 | R1<br>K1 | I/O | TTL | EPI module 0 signal 2. |
| | EPI0S3 | T1<br>K2 | I/O | TTL | EPI module 0 signal 3. |
| | EPI0S4 | K3 | I/O | TTL | EPI module 0 signal 4. |
| | EPI0S5 | L2 | I/O | TTL | EPI module 0 signal 5. |
| | EPI0S6 | M1 | I/O | TTL | EPI module 0 signal 6. |
| | EPI0S7 | M2 | I/O | TTL | EPI module 0 signal 7. |
| | EPI0S8 | V5 | I/O | TTL | EPI module 0 signal 8. |
| | EPI0S9 | R7 | I/O | TTL | EPI module 0 signal 9. |
| | EPI0S10 | T14 | I/O | TTL | EPI module 0 signal 10. |
| | EPI0S11 | N15 | I/O | TTL | EPI module 0 signal 11. |
| | EPI0S12 | L19 | I/O | TTL | EPI module 0 signal 12. |
| | EPI0S13 | L18 | I/O | TTL | EPI module 0 signal 13. |
| | EPI0S14 | K19 | I/O | TTL | EPI module 0 signal 14. |
| | EPI0S15 | K18 | I/O | TTL | EPI module 0 signal 15. |
| | EPI0S16 | G16 | I/O | TTL | EPI module 0 signal 16. |
| | EPI0S17 | H19 | I/O | TTL | EPI module 0 signal 17. |
| | EPI0S18 | G18 | I/O | TTL | EPI module 0 signal 18. |
| | EPI0S19 | J18 | I/O | TTL | EPI module 0 signal 19. |
| | EPI0S20 | E3 | I/O | TTL | EPI module 0 signal 20. |
| | EPI0S21 | E2 | I/O | TTL | EPI module 0 signal 21. |
| | EPI0S22 | H4 | I/O | TTL | EPI module 0 signal 22. |
| | EPI0S23 | M4 | I/O | TTL | EPI module 0 signal 23. |
| | EPI0S24 | W16 | I/O | TTL | EPI module 0 signal 24. |
| | EPI0S25 | V16 | I/O | TTL | EPI module 0 signal 25. |
| | EPI0S26 | H18 | I/O | TTL | EPI module 0 signal 26. |
| | EPI0S27 | A17 | I/O | TTL | EPI module 0 signal 27. |
| | EPI0S28 | B17 | I/O | TTL | EPI module 0 signal 28. |
| | EPI0S29 | A11<br>B13 | I/O | TTL | EPI module 0 signal 29. |
| | EPI0S30 | B10<br>C12 | I/O | TTL | EPI module 0 signal 30. |
| | EPI0S31 | V17 | I/O | TTL | EPI module 0 signal 31. |
| | EPI0S32 | U19 | I/O | TTL | EPI module 0 signal 32. |
| | EPI0S33 | G19 | I/O | TTL | EPI module 0 signal 33. |
| | EPI0S34 | A10 | I/O | TTL | EPI module 0 signal 34. |
| | EPI0S35 | B9 | I/O | TTL | EPI module 0 signal 35. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| General-Purpose Timers | T0CCP0 | V3<br>C2<br>H18<br>P3 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 0. |
| | T0CCP1 | W3<br>C1<br>G19<br>P2 | I/O | TTL | 16/32-Bit Timer 0 Capture/Compare/PWM 1. |
| | T1CCP0 | T6<br>D2<br>C18<br>W9 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 0. |
| | T1CCP1 | U5<br>D1<br>B18<br>R10 | I/O | TTL | 16/32-Bit Timer 1 Capture/Compare/PWM 1. |
| | T2CCP0 | V4<br>K18<br>D12 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 0. |
| | T2CCP1 | W4<br>K19<br>D13 | I/O | TTL | 16/32-Bit Timer 2 Capture/Compare/PWM 1. |
| | T3CCP0 | V5<br>A4<br>L18<br>B14 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 0. |
| | T3CCP1 | R7<br>B4<br>L19<br>A14 | I/O | TTL | 16/32-Bit Timer 3 Capture/Compare/PWM 1. |
| | T4CCP0 | A16<br>B3<br>M18<br>V9 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 0. |
| | T4CCP1 | B16<br>B2<br>G15<br>T13 | I/O | TTL | 16/32-Bit Timer 4 Capture/Compare/PWM 1. |
| | T5CCP0 | A17<br>N19<br>U10 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 0. |
| | T5CCP1 | B17<br>N18<br>R13 | I/O | TTL | 16/32-Bit Timer 5 Capture/Compare/PWM 1. |
| | T6CCP0 | F2<br>D6<br>E3<br>W10 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 0. |
| | T6CCP1 | F1<br>D7<br>E2<br>V10 | I/O | TTL | 16/32-Bit Timer 6 Capture/Compare/PWM 1. |
| | T7CCP0 | | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 0. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | | M2<br>H4<br>E18 | | | |
| | T7CCP1 | M1<br>M4<br>F17 | I/O | TTL | 16/32-Bit Timer 7 Capture/Compare/PWM 1. |
| Hibernate | GNDX | R18 | - | Power | GND for the Hibernation oscillator. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| | HIB | M17 | O | TTL | An output that indicates the processor is in Hibernate mode. |
| | RTCCLK | M1<br>W16<br>C12 | O | TTL | Buffered version of the Hibernation module's 32.768-kHz clock. This signal is not output when the part is in Hibernate mode and before being configured after power-on reset. |
| | TMPR0 | N18 | I/O | TTL | Tamper signal 0. |
| | TMPR1 | N19 | I/O | TTL | Tamper signal 1. |
| | TMPR2 | G15 | I/O | TTL | Tamper signal 2. |
| | TMPR3 | M18 | I/O | TTL | Tamper signal 3. |
| | VBAT | P19 | - | Power | Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply. |
| | WAKE | U18 | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| | XOSC0 | T18 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a crystal or a 32.768-kHz oscillator for the Hibernation module RTC. |
| | XOSC1 | T19 | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| I2C | I2C0SCL | A17 | I/O | OD | $I^2C$ module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C0SDA | B17 | I/O | OD | $I^2C$ module 0 data. |
| | I2C1SCL | N15<br>N5 | I/O | OD | $I^2C$ module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C1SDA | T14<br>N4 | I/O | OD | $I^2C$ module 1 data. |
| | I2C2SCL | V11<br>H19<br>B9<br>B12<br>N2 | I/O | OD | $I^2C$ module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C2SDA | M16<br>G16<br>A10<br>B8<br>V8 | I/O | OD | $I^2C$ module 2 data. |
| | I2C3SCL | K17<br>U19<br>P3 | I/O | OD | $I^2C$ module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C3SDA | K15<br>V17<br>P2 | I/O | OD | $I^2C$ module 3 data. |
| | I2C4SCL | V12<br>V16<br>W9 | I/O | OD | $I^2C$ module 4 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C4SDA | U14<br>W16<br>R10 | I/O | OD | $I^2C$ module 4 data. |
| | I2C5SCL | A16<br>C6 | I/O | OD | $I^2C$ module 5 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C5SDA | B16<br>B6 | I/O | OD | $I^2C$ module 5 data. |
| | I2C6SCL | V5<br>F2 | I/O | OD | $I^2C$ module 6 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C6SDA | R7<br>F1 | I/O | OD | $I^2C$ module 6 data. |
| | I2C7SCL | V4<br>C2 | I/O | OD | $I^2C$ module 7 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C7SDA | W4<br>C1 | I/O | OD | $I^2C$ module 7 data. |
| | I2C8SCL | T6<br>D2 | I/O | OD | $I^2C$ module 8 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C8SDA | U5<br>D1 | I/O | OD | $I^2C$ module 8 data. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | I2C9SCL | V3<br>A7 | I/O | OD | $I^2C$ module 9 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain. |
| | I2C9SDA | W3<br>B7 | I/O | OD | $I^2C$ module 9 data. |
| JTAG/SWD/SWO | SWCLK | B15 | I | TTL | JTAG/SWD CLK. |
| | SWDIO | C15 | I/O | TTL | JTAG TMS and SWDIO. |
| | SWO | C14 | O | TTL | JTAG TDO and SWO. |
| | TCK | B15 | I | TTL | JTAG/SWD CLK. |
| | TDI | D14 | I | TTL | JTAG TDI. |
| | TDO | C14 | O | TTL | JTAG TDO and SWO. |
| | TMS | C15 | I | TTL | JTAG TMS and SWDIO. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| LCD | LCDAC | N1 | O | TTL | LCD AC bias or latch enable in Raster mode; Primary chip select (CS0)/Primary Enable (E0) in LIDD MPU/Hitachi mode |
| | LCDCP | N5 | O | TTL | LCD Pixel Clock in Raster mode; read strobe or read/write strobe in LIDD mode |
| | LCDDATA00 | P3 | I/O | TTL | LCD Data Pin 0 input/output. |
| | LCDDATA01 | P2 | I/O | TTL | LCD Data Pin 1 input/output. |
| | LCDDATA02 | U8 | I/O | TTL | LCD Data Pin 2 input/output. |
| | LCDDATA03 | V8 | I/O | TTL | LCD Data Pin 3 input/output. |
| | LCDDATA04 | W9 | I/O | TTL | LCD Data Pin 4 input/output. |
| | LCDDATA05 | R10 | I/O | TTL | LCD Data Pin 5 input/output. |
| | LCDDATA06 | V9 | I/O | TTL | LCD Data Pin 6 input/output. |
| | LCDDATA07 | T13 | I/O | TTL | LCD Data Pin 7 input/output. |
| | LCDDATA08 | U10 | I/O | TTL | LCD Data Pin 8 input/output. |
| | LCDDATA09 | R13 | I/O | TTL | LCD Data Pin 9 input/output. |
| | LCDDATA10 | W10 | I/O | TTL | LCD Data Pin 10 input/output. |
| | LCDDATA11 | V10 | I/O | TTL | LCD Data Pin 11 input/output. |
| | LCDDATA12 | U12 | I/O | TTL | LCD Data Pin 12 input/output. |
| | LCDDATA13 | T12 | I/O | TTL | LCD Data Pin 13 input/output. |
| | LCDDATA14 | H17 | I/O | TTL | LCD Data Pin 14 input/output. |
| | LCDDATA15 | F16 | I/O | TTL | LCD Data Pin 15 input/output. |
| | LCDDATA16 | F18 | O | TTL | LCD Data Pin 16 output. |
| | LCDDATA17 | E17 | O | TTL | LCD Data Pin 17 output. |
| | LCDDATA18 | E18 | O | TTL | LCD Data Pin 18 output. |
| | LCDDATA19 | F17 | O | TTL | LCD Data Pin 19 output. |
| | LCDDATA20 | D12 | O | TTL | LCD Data Pin 20 output. |
| | LCDDATA21 | D13 | O | TTL | LCD Data Pin 21 output. |
| | LCDDATA22 | B14 | O | TTL | LCD Data Pin 22 output. |
| | LCDDATA23 | A14 | O | TTL | LCD Data Pin 23 output. |
| | LCDFP | N4 | O | TTL | LCD Frame Clock or VSYNC in Raster mode; Address Latch Enable in LIDD mode |
| | LCDLP | N2 | O | TTL | LCD Line Clock or HSYNC in Raster mode; Write Strobe or Direction bit in LIDD mode |
| | LCDMCLK | T8 | O | TTL | LCD Memory Clock/Secondary chip Select (CS1)/Secondary Enable (E1) in LIDD Synchronous/Async MPU/Hitachi mode |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| PWM | M0FAULT0 | V7<br>D12 | I | TTL | Motion Control Module 0 PWM Fault 0. |
| | M0FAULT1 | V16<br>D13 | I | TTL | Motion Control Module 0 PWM Fault 1. |
| | M0FAULT2 | W16<br>B14 | I | TTL | Motion Control Module 0 PWM Fault 2. |
| | M0FAULT3 | G16<br>A14 | I | TTL | Motion Control Module 0 PWM Fault 3. |
| | M0PWM0 | U6<br>N5 | O | TTL | Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0. |
| | M0PWM1 | V6<br>N4 | O | TTL | Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0. |
| | M0PWM2 | W6<br>N2 | O | TTL | Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1. |
| | M0PWM3 | T7<br>V8 | O | TTL | Motion Control Module 0 PWM 3. This signal is controlled by Module 0 PWM Generator 1. |
| | M0PWM4 | N15<br>P3 | O | TTL | Motion Control Module 0 PWM 4. This signal is controlled by Module 0 PWM Generator 2. |
| | M0PWM5 | T14<br>P2 | O | TTL | Motion Control Module 0 PWM 5. This signal is controlled by Module 0 PWM Generator 2. |
| | M0PWM6 | U19<br>W9 | O | TTL | Motion Control Module 0 PWM 6. This signal is controlled by Module 0 PWM Generator 3. |
| | M0PWM7 | V17<br>R10 | O | TTL | Motion Control Module 0 PWM 7. This signal is controlled by Module 0 PWM Generator 3. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| Power | GND | F10<br>H10<br>H11<br>H12<br>J11<br>J12<br>K6<br>K9<br>P16<br>K10<br>R17<br>K13<br>K14<br>L8<br>L9<br>M8<br>M9<br>M10<br>N10<br>A1<br>A2<br>B1<br>V1<br>W1<br>W2<br>A18<br>A19<br>B19 | - | Power | Ground reference for logic and I/O pins. |
| | GNDA | G4 | - | Power | The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| | VDD | G10<br>H9<br>J8<br>J9<br>J10<br>K7<br>K8<br>K11<br>N16<br>P17<br>K12<br>L10<br>L11<br>L12<br>M11<br>M12<br>P10 | - | Power | Positive supply for I/O and some logic. |
| | VDDA | F3 | - | Power | The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in , regardless of system implementation. |
| | VDDC | H16<br>E10 | - | Power | |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | | | | | Positive supply for most of the logic function, including the processor core and most peripherals. The voltage on this pin is 1.2 V and is supplied by the on-chip LDO. The VDDC pins should only be connected to each other and an external capacitor as specified in Table 32-15 on page 2111 . |
| QEI | IDX0 | J18<br>U10 | I | TTL | QEI module 0 index. |
| | PhA0 | H19<br>V9 | I | TTL | QEI module 0 phase A. |
| | PhB0 | G18<br>T13 | I | TTL | QEI module 0 phase B. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| SSI | SSI0Clk | T6 | I/O | TTL | SSI module 0 clock |
| | SSI0Fss | U5 | I/O | TTL | SSI module 0 frame signal |
| | SSI0XDAT0 | V4 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 0 (SSI0TX in Legacy SSI Mode). |
| | SSI0XDAT1 | W4 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 1 (SSI0RX in Legacy SSI Mode). |
| | SSI0XDAT2 | V5 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 2. |
| | SSI0XDAT3 | R7 | I/O | TTL | SSI Module 0 Bi-directional Data Pin 3. |
| | SSI1Clk | B6 | I/O | TTL | SSI module 1 clock. |
| | SSI1Fss | C6 | I/O | TTL | SSI module 1 frame signal. |
| | SSI1XDAT0 | A5 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 0 (SSI1TX in Legacy SSI Mode). |
| | SSI1XDAT1 | B5 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 1 (SSI1RX in Legacy SSI Mode). |
| | SSI1XDAT2 | A4 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 2. |
| | SSI1XDAT3 | B4 | I/O | TTL | SSI Module 1 Bi-directional Data Pin 3. |
| | SSI2Clk | D1<br>U14 | I/O | TTL | SSI module 2 clock. |
| | SSI2Fss | D2<br>V12 | I/O | TTL | SSI module 2 frame signal. |
| | SSI2XDAT0 | C1<br>K15 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 0 (SSI2TX in Legacy SSI Mode). |
| | SSI2XDAT1 | C2<br>K17 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 1 (SSI2RX in Legacy SSI Mode). |
| | SSI2XDAT2 | B2<br>M16 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 2. |
| | SSI2XDAT3 | B3<br>V11 | I/O | TTL | SSI Module 2 Bi-directional Data Pin 3. |
| | SSI3Clk | T7<br>E3 | I/O | TTL | SSI module 3 clock. |
| | SSI3Fss | W6<br>E2 | I/O | TTL | SSI module 3 frame signal. |
| | SSI3XDAT0 | V6<br>H4 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 0 (SSI3TX in Legacy SSI Mode). |
| | SSI3XDAT1 | U6<br>M4 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 1 (SSI3RX in Legacy SSI Mode). |
| | SSI3XDAT2 | V7<br>D6 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 2. |
| | SSI3XDAT3 | W7<br>D7 | I/O | TTL | SSI Module 3 Bi-directional Data Pin 3. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| System Control & Clocks | DIVSCLK | A13 | O | TTL | An optionally divided reference clock output based on a selected clock source. Note that this signal is not synchronized to the System Clock. |
| | GNDX2 | D18 | - | Power | GND for the MOSC. When using a crystal clock source, this pin should be connected to digital ground along with the crystal load capacitors. When using an external oscillator, this pin should be connected to digital ground. |
| | NMI | B2<br>B7 | I | TTL | Non-maskable interrupt. |
| | OSC0 | E19 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| | OSC1 | D19 | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| | RST | P18 | I | TTL | System reset input. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| UART | U0CTS | C6<br>A7<br>K17<br>R2<br>M18 | I | TTL | UART module 0 Clear To Send modem flow control input signal. |
| | U0DCD | R1<br>G15<br>C12 | I | TTL | UART module 0 Data Carrier Detect modem status input signal. |
| | U0DSR | T1<br>N19<br>D8 | I | TTL | UART module 0 Data Set Ready modem output control line. |
| | U0DTR | R3<br>B13 | O | TTL | UART module 0 Data Terminal Ready modem status input signal. |
| | U0RI | T2<br>W16<br>N18 | I | TTL | UART module 0 Ring Indicator modem status input signal. |
| | U0RTS | B6<br>B7<br>K15<br>P4 | O | TTL | UART module 0 Request to Send modem flow control output signal. |
| | U0Rx | V3 | I | TTL | UART module 0 receive. |
| | U0Tx | W3 | O | TTL | UART module 0 transmit. |
| | U1CTS | B11<br>C12 | I | TTL | UART module 1 Clear To Send modem flow control input signal. |
| | U1DCD | G1<br>A11<br>B8 | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| | U1DSR | H2<br>B10<br>B14 | I | TTL | UART module 1 Data Set Ready modem output control line. |
| | U1DTR | G2<br>A10<br>U15 | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| | U1RI | A5<br>B9<br>M3 | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| | U1RTS | H3<br>C10<br>U12 | O | TTL | UART module 1 Request to Send modem flow control output line. |
| | U1Rx | A16<br>A13<br>P2 | I | TTL | UART module 1 receive. |
| | U1Tx | B16<br>W12<br>W9 | O | TTL | UART module 1 transmit. |
| | U2CTS | B2<br>F16<br>B10 | I | TTL | UART module 2 Clear To Send modem flow control input signal. |
| | U2RTS | B3<br>H17<br>A11 | O | TTL | UART module 2 Request to Send modem flow control output line. |
| | U2Rx | | I | TTL | UART module 2 receive. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | | V5<br>A4 | | | |
| | U2TX | R7<br>B4 | O | TTL | UART module 2 transmit. |
| | U3CTS | E17<br>B9<br>B12 | I | TTL | UART module 3 Clear To Send modem flow control input signal. |
| | U3RTS | F18<br>A10<br>D8 | O | TTL | UART module 3 Request to Send modem flow control output line. |
| | U3RX | V4<br>C8 | I | TTL | UART module 3 receive. |
| | U3TX | W4<br>E7 | O | TTL | UART module 3 transmit. |
| | U4CTS | K5<br>K2<br>U12 | I | TTL | UART module 4 Clear To Send modem flow control input signal. |
| | U4RTS | N1<br>K1<br>T12 | O | TTL | UART module 4 Request to Send modem flow control output line. |
| | U4RX | T6<br>J1<br>N4 | I | TTL | UART module 4 receive. |
| | U4TX | U5<br>J2<br>N5 | O | TTL | UART module 4 transmit. |
| | U5RX | L2<br>U2 | I | TTL | UART module 5 receive. |
| | U5TX | K3<br>V2 | O | TTL | UART module 5 transmit. |
| | U6RX | D6 | I | TTL | UART module 6 receive. |
| | U6TX | D7 | O | TTL | UART module 6 transmit. |
| | U7RX | M2<br>U2 | I | TTL | UART module 7 receive. |
| | U7TX | M1<br>V2 | O | TTL | UART module 7 transmit. |

**Table 31-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| USB | USB0CLK | B17 | O | TTL | 60-MHz clock to the external PHY. |
| | USB0D0 | G16 | I/O | TTL | USB data 0. |
| | USB0D1 | H19 | I/O | TTL | USB data 1. |
| | USB0D2 | G18 | I/O | TTL | USB data 2. |
| | USB0D3 | J18 | I/O | TTL | USB data 3. |
| | USB0D4 | H18 | I/O | TTL | USB data 4. |
| | USB0D5 | G19 | I/O | TTL | USB data 5. |
| | USB0D6 | B12 | I/O | TTL | USB data 6. |
| | USB0D7 | D8 | I/O | TTL | USB data 7. |
| | USB0DIR | C12 | O | TTL | Indicates that the external PHY is able to accept data from the USB controller. |
| | USB0DM | B18 | I/O | Analog | Bidirectional differential data pin (D- per USB specification) for USB0. |
| | USB0DP | C18 | I/O | Analog | Bidirectional differential data pin (D+ per USB specification) for USB0. |
| | USB0EPEN | V5<br>R7<br>B3 | O | TTL | Optionally used in Host mode to control an external power source to supply power to the USB bus. |
| | USB0ID | A16 | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is the A side of the cable and pulled up is the B side). |
| | USB0NXT | B13 | O | TTL | Asserted by the external PHY to throttle all data types. |
| | USB0PFLT | R7<br>B2 | I | TTL | Optionally used in Host mode by an external power source to indicate an error state by that power source. |
| | USB0STP | A17 | O | TTL | Asserted by the USB controller to signal the end of a USB transmit packet or register write operation. |
| | USB0VBUS | B16 | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |

## 31.4 GPIO Pins and Alternate Functions

**Table 31-5. GPIO Pins and Alternate Functions**

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PA0 | V3 | - | U0Rx | I2C9SCL | T0CCP0 | - | - | - | CAN0Rx | - | - | - | - | - |
| PA1 | W3 | - | U0Tx | I2C9SDA | T0CCP1 | - | - | - | CAN0Tx | - | - | - | - | - |
| PA2 | T6 | - | U4Rx | I2C8SCL | T1CCP0 | - | - | - | - | - | - | - | - | SSI0Clk |
| PA3 | U5 | - | U4Tx | I2C8SDA | T1CCP1 | - | - | - | - | - | - | - | - | SSI0Fss |
| PA4 | V4 | - | U3Rx | I2C7SCL | T2CCP0 | - | - | - | - | - | - | - | - | SSI0XDAT0 |

**Table 31-5. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PA5 | W4 | - | U3Tx | I2C7SDA | T2CCP1 | - | - | - | - | - | - | - | - | SSI0XDAT1 |
| PA6 | V5 | - | U2Rx | I2C6SCL | T3CCP0 | - | USB0EPEN | - | - | - | - | SSI0XDAT2 | EN0RXCK | EPI0S8 |
| PA7 | R7 | - | U2Tx | I2C6SDA | T3CCP1 | - | USB0PFLT | - | - | - | USB0EPEN | SSI0XDAT3 | - | EPI0S9 |
| PB0 | A16 | USB0ID | U1Rx | I2C5SCL | T4CCP0 | - | - | - | CAN1Rx | - | - | - | - | - |
| PB1 | B16 | USB0VBUS | U1Tx | I2C5SDA | T4CCP1 | - | - | - | CAN1Tx | - | - | - | - | - |
| PB2 | A17 | - | - | I2C0SCL | T5CCP0 | - | EN0MDC | - | - | - | - | - | USB0STP | EPI0S27 |
| PB3 | B17 | - | - | I2C0SDA | T5CCP1 | - | EN0MDIO | - | - | - | - | - | USB0CLK | EPI0S28 |
| PB4 | C6 | AIN10 | U0CTS | I2C5SCL | - | - | - | - | - | - | - | - | - | SSI1Fss |
| PB5 | B6 | AIN11 | U0RTS | I2C5SDA | - | - | - | - | - | - | - | - | - | SSI1Clk |
| PB6 | F2 | - | - | I2C6SCL | T6CCP0 | - | - | - | - | - | - | - | - | - |
| PB7 | F1 | - | - | I2C6SDA | T6CCP1 | - | - | - | - | - | - | - | - | - |
| PC0 | B15 | - | TCK SWCLK | - | - | - | - | - | - | - | - | - | - | - |
| PC1 | C15 | - | TMS SWDIO | - | - | - | - | - | - | - | - | - | - | - |
| PC2 | D14 | - | TDI | - | - | - | - | - | - | - | - | - | - | - |
| PC3 | C14 | - | TDO SWO | - | - | - | - | - | - | - | - | - | - | - |
| PC4 | M2 | C1- | U7Rx | - | T7CCP0 | - | - | - | - | - | - | - | - | EPI0S7 |
| PC5 | M1 | C1+ | U7Tx | - | T7CCP1 | - | - | - | RTCCLK | - | - | - | - | EPI0S6 |
| PC6 | L2 | C0+ | U5Rx | - | - | - | - | - | - | - | - | - | - | EPI0S5 |
| PC7 | K3 | C0- | U5Tx | - | - | - | - | - | - | - | - | - | - | EPI0S4 |
| PD0 | C2 | AIN15 | - | I2C7SCL | T0CCP0 | - | C0o | - | - | - | - | - | - | SSI2XDAT1 |
| PD1 | C1 | AIN14 | - | I2C7SDA | T0CCP1 | - | C1o | - | - | - | - | - | - | SSI2XDAT0 |
| PD2 | D2 | AIN13 | - | I2C8SCL | T1CCP0 | - | C2o | - | - | - | - | - | - | SSI2Fss |
| PD3 | D1 | AIN12 | - | I2C8SDA | T1CCP1 | - | - | - | - | - | - | - | - | SSI2Clk |
| PD4 | A4 | AIN7 | U2Rx | - | T3CCP0 | - | - | - | - | - | - | - | - | SSI1XDAT2 |
| PD5 | B4 | AIN6 | U2Tx | - | T3CCP1 | - | - | - | - | - | - | - | - | SSI1XDAT3 |
| PD6 | B3 | AIN5 | U2RTS | - | T4CCP0 | - | USB0EPEN | - | - | - | - | - | - | SSI2XDAT3 |
| PD7 | B2 | AIN4 | U2CTS | - | T4CCP1 | - | USB0PFLT | - | - | NMI | - | - | - | SSI2XDAT2 |
| PE0 | H3 | AIN3 | U1RTS | - | - | - | - | - | - | - | - | - | - | - |
| PE1 | H2 | AIN2 | U1DSR | - | - | - | - | - | - | - | - | - | - | - |
| PE2 | G1 | AIN1 | U1DCD | - | - | - | - | - | - | - | - | - | - | - |
| PE3 | G2 | AIN0 | U1DTR | - | - | - | OWIRE | - | - | - | - | - | - | - |
| PE4 | A5 | AIN9 | U1RI | - | - | - | - | - | - | - | - | - | - | SSI1XDAT0 |
| PE5 | B5 | AIN8 | - | - | - | - | - | - | - | - | - | - | - | SSI1XDAT1 |
| PE6 | A7 | AIN20 | U0CTS | I2C9SCL | - | - | - | - | - | - | - | - | - | - |
| PE7 | B7 | AIN21 | U0RTS | I2C9SDA | - | - | - | - | - | NMI | - | - | - | - |
| PF0 | U6 | - | - | - | - | - | EN0LED0 | M0PWM0 | - | - | - | - | SSI3XDAT1 | TRD2 |
| PF1 | V6 | - | - | - | - | - | EN0LED2 | M0PWM1 | - | - | - | - | SSI3XDAT0 | TRD1 |

**Table 31-5. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PF2 | W6 | - | - | - | - | - | EN0MDC | M0PWM2 | - | - | - | - | SSI3Fss | TRD0 |
| PF3 | T7 | - | - | - | - | - | EN0MDIO | M0PWM3 | - | - | - | - | SSI3Clk | TRCLK |
| PF4 | V7 | - | - | - | - | - | EN0LED1 | M0FAULT0 | - | - | - | - | SSI3XDAT2 | TRD3 |
| PF5 | W7 | - | - | - | - | - | - | - | - | - | - | - | SSI3XDAT3 | - |
| PF6 | T8 | - | - | - | - | - | - | - | - | - | - | - | - | LCDMCLK |
| PF7 | U8 | - | - | - | - | - | - | - | - | - | - | - | - | LCDDATA02 |
| PG0 | N15 | - | - | I2C1SCL | - | - | EN0PPS | M0PWM4 | - | - | - | - | - | EPI0S11 |
| PG1 | T14 | - | - | I2C1SDA | - | - | - | M0PWM5 | - | - | - | - | - | EPI0S10 |
| PG2 | V11 | - | - | I2C2SCL | - | - | - | - | - | - | - | - | EN0TXCK | SSI2XDAT3 |
| PG3 | M16 | - | - | I2C2SDA | - | - | - | - | - | - | - | - | EN0TXEN | SSI2XDAT2 |
| PG4 | K17 | - | U0CTS | I2C3SCL | - | - | OWIRE | - | - | - | - | - | EN0TXD0 | SSI2XDAT1 |
| PG5 | K15 | - | U0RTS | I2C3SDA | - | - | OWALT | - | - | - | - | - | EN0TXD1 | SSI2XDAT0 |
| PG6 | V12 | - | - | I2C4SCL | - | - | OWIRE | - | - | - | - | - | EN0RXER | SSI2Fss |
| PG7 | U14 | - | - | I2C4SDA | - | - | OWIRE | - | - | - | - | - | EN0RXDV | SSI2Clk |
| PH0 | P4 | - | U0RTS | - | - | - | - | - | - | - | - | - | - | EPI0S0 |
| PH1 | R2 | - | U0CTS | - | - | - | - | - | - | - | - | - | - | EPI0S1 |
| PH2 | R1 | - | U0DCD | - | - | - | - | - | - | - | - | - | - | EPI0S2 |
| PH3 | T1 | - | U0DSR | - | - | - | - | - | - | - | - | - | - | EPI0S3 |
| PH4 | R3 | - | U0DTR | - | - | - | - | - | - | - | - | - | - | - |
| PH5 | T2 | - | U0RI | - | - | - | EN0PPS | - | - | - | - | - | - | - |
| PH6 | U2 | - | U5Rx | U7Rx | - | - | - | - | - | - | - | - | - | - |
| PH7 | V2 | - | U5Tx | U7Tx | - | - | - | - | - | - | - | - | - | - |
| PJ0 | C8 | - | U3Rx | - | - | - | EN0PPS | - | - | - | - | - | - | - |
| PJ1 | E7 | - | U3Tx | - | - | - | - | - | - | - | - | - | - | - |
| PJ2 | H17 | - | U2RTS | - | - | - | - | - | - | - | - | - | - | LCDDATA14 |
| PJ3 | F16 | - | U2CTS | - | - | - | - | - | - | - | - | - | - | LCDDATA15 |
| PJ4 | F18 | - | U3RTS | - | - | - | - | - | - | - | - | - | - | LCDDATA16 |
| PJ5 | E17 | - | U3CTS | - | - | - | - | - | - | - | - | - | - | LCDDATA17 |
| PJ6 | N1 | - | U4RTS | - | - | - | - | - | - | - | - | - | - | LCDAC |
| PJ7 | K5 | - | U4CTS | - | - | - | - | - | - | - | - | - | - | - |
| PK0 | J1 | AIN16 | U4Rx | - | - | - | - | - | - | - | - | - | - | EPI0S0 |
| PK1 | J2 | AIN17 | U4Tx | - | - | - | - | - | - | - | - | - | - | EPI0S1 |
| PK2 | K1 | AIN18 | U4RTS | - | - | - | - | - | - | - | - | - | - | EPI0S2 |
| PK3 | K2 | AIN19 | U4CTS | - | - | - | - | - | - | - | - | - | - | EPI0S3 |
| PK4 | U19 | - | - | I2C3SCL | - | - | EN0LED0 | M0PWM6 | EN0INTRN | - | - | - | EN0RXD3 | EPI0S32 |
| PK5 | V17 | - | - | I2C3SDA | - | - | EN0LED2 | M0PWM7 | - | - | - | - | EN0RXD2 | EPI0S31 |
| PK6 | V16 | - | - | I2C4SCL | - | - | EN0LED1 | M0FAULT1 | - | - | - | - | EN0TXD2 | EPI0S25 |
| PK7 | W16 | - | U0RI | I2C4SDA | - | - | RTCCLK | M0FAULT2 | - | - | - | - | EN0TXD3 | EPI0S24 |
| PL0 | G16 | - | - | I2C2SDA | - | - | - | M0FAULT3 | - | - | - | - | USB0D0 | EPI0S16 |

**Table 31-5. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PL1 | H19 | - | - | I2C2SCL | - | - | - | PhA0 | - | - | - | - | USB0D1 | EPI0S17 |
| PL2 | G18 | - | - | - | - | - | C0o | PhB0 | - | - | - | - | USB0D2 | EPI0S18 |
| PL3 | J18 | - | - | - | - | - | C1o | IDX0 | - | - | - | - | USB0D3 | EPI0S19 |
| PL4 | H18 | - | - | - | T0CCP0 | - | - | - | - | - | - | - | USB0D4 | EPI0S26 |
| PL5 | G19 | - | - | - | T0CCP1 | - | - | - | - | - | - | - | USB0D5 | EPI0S33 |
| PL6 | C18 | USB0DP | - | - | T1CCP0 | - | - | - | - | - | - | - | - | - |
| PL7 | B18 | USB0DM | - | - | T1CCP1 | - | - | - | - | - | - | - | - | - |
| PM0 | K18 | - | - | - | T2CCP0 | - | - | - | - | - | - | - | - | EPI0S15 |
| PM1 | K19 | - | - | - | T2CCP1 | - | - | - | - | - | - | - | - | EPI0S14 |
| PM2 | L18 | - | - | - | T3CCP0 | - | - | - | - | - | - | - | - | EPI0S13 |
| PM3 | L19 | - | - | - | T3CCP1 | - | - | - | - | - | - | - | - | EPI0S12 |
| PM4 | M18 | TMPR3 | U0CTS | - | T4CCP0 | - | - | - | - | - | - | - | EN0RREF_CLK | - |
| PM5 | G15 | TMPR2 | U0DCD | - | T4CCP1 | - | - | - | - | - | - | - | - | - |
| PM6 | N19 | TMPR1 | U0DSR | - | T5CCP0 | - | - | - | - | - | - | - | EN0CRS | - |
| PM7 | N18 | TMPR0 | U0RI | - | T5CCP1 | - | - | - | - | - | - | - | EN0COL | - |
| PN0 | C10 | - | U1RTS | - | - | - | - | - | - | - | - | - | - | - |
| PN1 | B11 | - | U1CTS | - | - | - | - | - | - | - | - | - | - | - |
| PN2 | A11 | - | U1DCD | U2RTS | - | - | - | - | - | - | - | - | - | EPI0S29 |
| PN3 | B10 | - | U1DSR | U2CTS | - | - | - | - | - | - | - | - | - | EPI0S30 |
| PN4 | A10 | - | U1DTR | U3RTS | I2C2SDA | - | - | - | - | - | - | - | - | EPI0S34 |
| PN5 | B9 | - | U1RI | U3CTS | I2C2SCL | - | - | - | - | - | - | - | - | EPI0S35 |
| PN6 | T12 | - | - | U4RTS | - | - | - | - | - | - | - | - | EN0TXER | LCDDATA13 |
| PN7 | U12 | - | U1RTS | U4CTS | - | - | - | - | - | - | - | - | - | LCDDATA12 |
| PP0 | D6 | C2+ | U6Rx | - | - | - | T6CCP0 | - | EN0INTRN | - | - | - | - | SSI3XDAT2 |
| PP1 | D7 | C2- | U6Tx | - | - | - | T6CCP1 | - | - | - | - | - | - | SSI3XDAT3 |
| PP2 | B13 | - | U0DTR | - | - | - | - | - | - | - | - | - | USB0NXT | EPI0S29 |
| PP3 | C12 | - | U1CTS | U0DCD | - | - | - | - | RTCCLK | - | - | - | USB0DIR | EPI0S30 |
| PP4 | D8 | - | U3RTS | U0DSR | - | OWIRE | - | - | - | - | - | - | USB0D7 | - |
| PP5 | B12 | - | U3CTS | I2C2SCL | - | OWALT | - | - | - | - | - | - | USB0D6 | - |
| PP6 | B8 | AIN23 | U1DCD | I2C2SDA | - | - | - | - | - | - | - | - | - | - |
| PP7 | A8 | AIN22 | - | - | - | - | OWIRE | - | - | - | - | - | - | - |
| PQ0 | E3 | - | - | - | T6CCP0 | - | - | - | - | - | - | - | SSI3Clk | EPI0S20 |
| PQ1 | E2 | - | - | - | T6CCP1 | - | - | - | - | - | - | - | SSI3Fss | EPI0S21 |
| PQ2 | H4 | - | - | - | T7CCP0 | - | - | - | - | - | - | - | SSI3XDAT0 | EPI0S22 |
| PQ3 | M4 | - | - | - | T7CCP1 | - | - | - | - | - | - | - | SSI3XDAT1 | EPI0S23 |
| PQ4 | A13 | - | U1Rx | - | - | - | - | - | DIVSCLK | - | - | - | - | - |
| PQ5 | W12 | - | U1Tx | - | - | - | - | - | - | - | - | - | EN0RXD0 | - |
| PQ6 | U15 | - | U1DTR | - | - | - | - | - | - | - | - | - | EN0RXD1 | - |
| PQ7 | M3 | - | U1RI | - | - | - | - | - | - | - | - | - | - | - |

**Table 31-5. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog or Special Function[a] | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b] | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 11 | 13 | 14 | 15 |
| PR0 | N5 | - | U4Tx | I2C1SCL | - | - | - | M0PWM0 | - | - | - | - | - | LCDCP |
| PR1 | N4 | - | U4Rx | I2C1SDA | - | - | - | M0PWM1 | - | - | - | - | - | LCDFP |
| PR2 | N2 | - | - | I2C2SCL | - | - | - | M0PWM2 | - | - | - | - | - | LCDLP |
| PR3 | V8 | - | - | I2C2SDA | - | - | - | M0PWM3 | - | - | - | - | - | LCDDATA03 |
| PR4 | P3 | - | - | I2C3SCL | T0CCP0 | - | - | M0PWM4 | - | - | - | - | - | LCDDATA00 |
| PR5 | P2 | - | U1Rx | I2C3SDA | T0CCP1 | - | - | M0PWM5 | - | - | - | - | - | LCDDATA01 |
| PR6 | W9 | - | U1Tx | I2C4SCL | T1CCP0 | - | - | M0PWM6 | - | - | - | - | - | LCDDATA04 |
| PR7 | R10 | - | - | I2C4SDA | T1CCP1 | - | - | M0PWM7 | - | - | - | - | EN0TXEN | LCDDATA05 |
| PS0 | D12 | - | - | - | T2CCP0 | - | - | M0FAULT0 | - | - | - | - | - | LCDDATA20 |
| PS1 | D13 | - | - | - | T2CCP1 | - | - | M0FAULT1 | - | - | - | - | - | LCDDATA21 |
| PS2 | B14 | - | U1DSR | - | T3CCP0 | - | - | M0FAULT2 | - | - | - | - | - | LCDDATA22 |
| PS3 | A14 | - | - | - | T3CCP1 | - | - | M0FAULT3 | - | - | - | - | - | LCDDATA23 |
| PS4 | V9 | - | - | - | T4CCP0 | - | - | PhA0 | - | - | - | - | EN0TXD0 | LCDDATA06 |
| PS5 | T13 | - | - | - | T4CCP1 | - | - | PhB0 | - | - | - | - | EN0TXD1 | LCDDATA07 |
| PS6 | U10 | - | - | - | T5CCP0 | - | - | IDX0 | - | - | - | - | EN0RXER | LCDDATA08 |
| PS7 | R13 | - | - | - | T5CCP1 | - | - | - | - | - | - | - | EN0RXDV | LCDDATA09 |
| PT0 | W10 | - | - | - | T6CCP0 | - | - | - | CAN0Rx | - | - | - | EN0RXD0 | LCDDATA10 |
| PT1 | V10 | - | - | - | T6CCP1 | - | - | - | CAN0Tx | - | - | - | EN0RXD1 | LCDDATA11 |
| PT2 | E18 | - | - | - | T7CCP0 | - | - | - | CAN1Rx | - | - | - | - | LCDDATA18 |
| PT3 | F17 | - | - | - | T7CCP1 | - | - | - | CAN1Tx | - | - | - | - | LCDDATA19 |

a. The TMPRn signals are digital signals enabled and configured by the Hibernation module. All other signals listed in this column are analog signals.

b. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin. Encodings 9, 10, and 12 are not used on this device.

## 31.5 Possible Pin Assignments for Alternate Functions

**Table 31-6. Possible Pin Assignments for Alternate Functions**

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| one | AIN0 | PE3 |
| | AIN1 | PE2 |
| | AIN10 | PB4 |
| | AIN11 | PB5 |
| | AIN12 | PD3 |
| | AIN13 | PD2 |
| | AIN14 | PD1 |
| | AIN15 | PD0 |
| | AIN16 | PK0 |
| | AIN17 | PK1 |
| | AIN18 | PK2 |
| | AIN19 | PK3 |
| | AIN2 | PE1 |
| | AIN20 | PE6 |
| | AIN21 | PE7 |
| | AIN22 | PP7 |
| | AIN23 | PP6 |
| | AIN3 | PE0 |
| | AIN4 | PD7 |
| | AIN5 | PD6 |
| | AIN6 | PD5 |
| | AIN7 | PD4 |
| | AIN8 | PE5 |
| | AIN9 | PE4 |
| | C0+ | PC6 |
| | C0- | PC7 |
| | C1+ | PC5 |
| | C1- | PC4 |
| | C2+ | PP0 |
| | C2- | PP1 |
| | C2o | PD2 |
| | DIVSCLK | PQ4 |
| | EN0COL | PM7 |
| | EN0CRS | PM6 |
| | EN0RREF_CLK | PM4 |
| | EN0RXCK | PA6 |
| | EN0RXD2 | PK5 |
| | EN0RXD3 | PK4 |
| | EN0TXCK | PG2 |

**Table 31-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| | EN0TXD2 | PK6 |
| | EN0TXD3 | PK7 |
| | EN0TXER | PN6 |
| | EPI0S10 | PG1 |
| | EPI0S11 | PG0 |
| | EPI0S12 | PM3 |
| | EPI0S13 | PM2 |
| | EPI0S14 | PM1 |
| | EPI0S15 | PM0 |
| | EPI0S16 | PL0 |
| | EPI0S17 | PL1 |
| | EPI0S18 | PL2 |
| | EPI0S19 | PL3 |
| | EPI0S20 | PQ0 |
| | EPI0S21 | PQ1 |
| | EPI0S22 | PQ2 |
| | EPI0S23 | PQ3 |
| | EPI0S24 | PK7 |
| | EPI0S25 | PK6 |
| | EPI0S26 | PL4 |
| | EPI0S27 | PB2 |
| | EPI0S28 | PB3 |
| | EPI0S31 | PK5 |
| | EPI0S32 | PK4 |
| | EPI0S33 | PL5 |
| | EPI0S34 | PN4 |
| | EPI0S35 | PN5 |
| | EPI0S4 | PC7 |
| | EPI0S5 | PC6 |
| | EPI0S6 | PC5 |
| | EPI0S7 | PC4 |
| | EPI0S8 | PA6 |
| | EPI0S9 | PA7 |
| | I2C0SCL | PB2 |
| | I2C0SDA | PB3 |
| | LCDAC | PJ6 |
| | LCDCP | PR0 |
| | LCDDATA00 | PR4 |
| | LCDDATA01 | PR5 |
| | LCDDATA02 | PF7 |
| | LCDDATA03 | PR3 |

**Table 31-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| | LCDDATA04 | PR6 |
| | LCDDATA05 | PR7 |
| | LCDDATA06 | PS4 |
| | LCDDATA07 | PS5 |
| | LCDDATA08 | PS6 |
| | LCDDATA09 | PS7 |
| | LCDDATA10 | PT0 |
| | LCDDATA11 | PT1 |
| | LCDDATA12 | PN7 |
| | LCDDATA13 | PN6 |
| | LCDDATA14 | PJ2 |
| | LCDDATA15 | PJ3 |
| | LCDDATA16 | PJ4 |
| | LCDDATA17 | PJ5 |
| | LCDDATA18 | PT2 |
| | LCDDATA19 | PT3 |
| | LCDDATA20 | PS0 |
| | LCDDATA21 | PS1 |
| | LCDDATA22 | PS2 |
| | LCDDATA23 | PS3 |
| | LCDFP | PR1 |
| | LCDLP | PR2 |
| | LCDMCLK | PF6 |
| | SSI0Clk | PA2 |
| | SSI0Fss | PA3 |
| | SSI0XDAT0 | PA4 |
| | SSI0XDAT1 | PA5 |
| | SSI0XDAT2 | PA6 |
| | SSI0XDAT3 | PA7 |
| | SSI1Clk | PB5 |
| | SSI1Fss | PB4 |
| | SSI1XDAT0 | PE4 |
| | SSI1XDAT1 | PE5 |
| | SSI1XDAT2 | PD4 |
| | SSI1XDAT3 | PD5 |
| | SWCLK | PC0 |
| | SWDIO | PC1 |
| | SWO | PC3 |
| | TCK | PC0 |
| | TDI | PC2 |
| | TDO | PC3 |

**Table 31-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| | TMPR0 | PM7 |
| | TMPR1 | PM6 |
| | TMPR2 | PM5 |
| | TMPR3 | PM4 |
| | TMS | PC1 |
| | TRCLK | PF3 |
| | TRD0 | PF2 |
| | TRD1 | PF1 |
| | TRD2 | PF0 |
| | TRD3 | PF4 |
| | U0Rx | PA0 |
| | U0Tx | PA1 |
| | U6Rx | PP0 |
| | U6Tx | PP1 |
| | USB0CLK | PB3 |
| | USB0D0 | PL0 |
| | USB0D1 | PL1 |
| | USB0D2 | PL2 |
| | USB0D3 | PL3 |
| | USB0D4 | PL4 |
| | USB0D5 | PL5 |
| | USB0D6 | PP5 |
| | USB0D7 | PP4 |
| | USB0DIR | PP3 |
| | USB0DM | PL7 |
| | USB0DP | PL6 |
| | USB0ID | PB0 |
| | USB0NXT | PP2 |
| | USB0STP | PB2 |
| | USB0VBUS | PB1 |

**Table 31-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| two | C0o | PD0 PL2 |
| | C1o | PD1 PL3 |
| | CAN0Rx | PA0 PT0 |
| | CAN0Tx | PA1 PT1 |
| | CAN1Rx | PB0 PT2 |
| | CAN1Tx | PB1 PT3 |
| | EN0INTRN | PK4 PP0 |
| | EN0LED0 | PF0 PK4 |
| | EN0LED1 | PF4 PK6 |
| | EN0LED2 | PF1 PK5 |
| | EN0MDC | PB2 PF2 |
| | EN0MDIO | PB3 PF3 |
| | EN0RXD0 | PQ5 PT0 |
| | EN0RXD1 | PQ6 PT1 |
| | EN0RXDV | PG7 PS7 |
| | EN0RXER | PG6 PS6 |
| | EN0TXD0 | PG4 PS4 |
| | EN0TXD1 | PG5 PS5 |
| | EN0TXEN | PG3 PR7 |
| | EPI0S0 | PH0 PK0 |
| | EPI0S1 | PH1 PK1 |
| | EPI0S2 | PH2 PK2 |
| | EPI0S29 | PN2 PP2 |
| | EPI0S3 | PH3 PK3 |
| | EPI0S30 | PN3 PP3 |
| | I2C1SCL | PG0 PR0 |
| | I2C1SDA | PG1 PR1 |
| | I2C5SCL | PB0 PB4 |
| | I2C5SDA | PB1 PB5 |
| | I2C6SCL | PA6 PB6 |
| | I2C6SDA | PA7 PB7 |
| | I2C7SCL | PA4 PD0 |
| | I2C7SDA | PA5 PD1 |
| | I2C8SCL | PA2 PD2 |
| | I2C8SDA | PA3 PD3 |
| | I2C9SCL | PA0 PE6 |
| | I2C9SDA | PA1 PE7 |
| | IDX0 | PL3 PS6 |
| | M0FAULT0 | PF4 PS0 |
| | M0FAULT1 | PK6 PS1 |
| | M0FAULT2 | PK7 PS2 |

**Table 31-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| | M0FAULT3 | PL0 PS3 |
| | M0PWM0 | PF0 PR0 |
| | M0PWM1 | PF1 PR1 |
| | M0PWM2 | PF2 PR2 |
| | M0PWM3 | PF3 PR3 |
| | M0PWM4 | PG0 PR4 |
| | M0PWM5 | PG1 PR5 |
| | M0PWM6 | PK4 PR6 |
| | M0PWM7 | PK5 PR7 |
| | NMI | PD7 PE7 |
| | OWALT | PG5 PP5 |
| | PhA0 | PL1 PS4 |
| | PhB0 | PL2 PS5 |
| | SSI2Clk | PD3 PG7 |
| | SSI2Fss | PD2 PG6 |
| | SSI2XDAT0 | PD1 PG5 |
| | SSI2XDAT1 | PD0 PG4 |
| | SSI2XDAT2 | PD7 PG3 |
| | SSI2XDAT3 | PD6 PG2 |
| | SSI3Clk | PF3 PQ0 |
| | SSI3Fss | PF2 PQ1 |
| | SSI3XDAT0 | PF1 PQ2 |
| | SSI3XDAT1 | PF0 PQ3 |
| | SSI3XDAT2 | PF4 PP0 |
| | SSI3XDAT3 | PF5 PP1 |
| | U0DTR | PH4 PP2 |
| | U1CTS | PN1 PP3 |
| | U2Rx | PA6 PD4 |
| | U2Tx | PA7 PD5 |
| | U3Rx | PA4 PJ0 |
| | U3Tx | PA5 PJ1 |
| | U5Rx | PC6 PH6 |
| | U5Tx | PC7 PH7 |
| | U7Rx | PC4 PH6 |
| | U7Tx | PC5 PH7 |
| | USB0PFLT | PA7 PD7 |

**Table 31-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| three | EN0PPS | PG0 PH5 PJ0 |
| | I2C3SCL | PG4 PK4 PR4 |
| | I2C3SDA | PG5 PK5 PR5 |
| | I2C4SCL | PG6 PK6 PR6 |
| | I2C4SDA | PG7 PK7 PR7 |
| | RTCCLK | PC5 PK7 PP3 |
| | T2CCP0 | PA4 PM0 PS0 |
| | T2CCP1 | PA5 PM1 PS1 |
| | T5CCP0 | PB2 PM6 PS6 |
| | T5CCP1 | PB3 PM7 PS7 |
| | T7CCP0 | PC4 PQ2 PT2 |
| | T7CCP1 | PC5 PQ3 PT3 |
| | U0DCD | PH2 PM5 PP3 |
| | U0DSR | PH3 PM6 PP4 |
| | U0RI | PH5 PK7 PM7 |
| | U1DCD | PE2 PN2 PP6 |
| | U1DSR | PE1 PN3 PS2 |
| | U1DTR | PE3 PN4 PQ6 |
| | U1RI | PE4 PN5 PQ7 |
| | U1RTS | PE0 PN0 PN7 |
| | U1Rx | PB0 PQ4 PR5 |
| | U1Tx | PB1 PQ5 PR6 |
| | U2CTS | PD7 PJ3 PN3 |
| | U2RTS | PD6 PJ2 PN2 |
| | U3CTS | PJ5 PN5 PP5 |
| | U3RTS | PJ4 PN4 PP4 |
| | U4CTS | PJ7 PK3 PN7 |
| | U4RTS | PJ6 PK2 PN6 |
| | U4Rx | PA2 PK0 PR1 |
| | U4Tx | PA3 PK1 PR0 |
| | USB0EPEN | PA6 PA7 PD6 |

**Table 31-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| four | T0CCP0 | PA0 PD0 PL4 PR4 |
| | T0CCP1 | PA1 PD1 PL5 PR5 |
| | T1CCP0 | PA2 PD2 PL6 PR6 |
| | T1CCP1 | PA3 PD3 PL7 PR7 |
| | T3CCP0 | PA6 PD4 PM2 PS2 |
| | T3CCP1 | PA7 PD5 PM3 PS3 |
| | T4CCP0 | PB0 PD6 PM4 PS4 |
| | T4CCP1 | PB1 PD7 PM5 PS5 |
| | T6CCP0 | PB6 PP0 PQ0 PT0 |
| | T6CCP1 | PB7 PP1 PQ1 PT1 |
| | U0RTS | PB5 PE7 PG5 PH0 |
| five | I2C2SCL | PG2 PL1 PN5 PP5 PR2 |
| | I2C2SDA | PG3 PL0 PN4 PP6 PR3 |
| | U0CTS | PB4 PE6 PG4 PH1 PM4 |
| six | OWIRE | PE3 PG4 PG6 PG7 PP4 PP7 |

## 31.6 Connections for Unused Signals

Table 31-7 on page 2093 show how to handle signals for functions that are not used in a particular system implementation for devices that are in a 212-ball BGA package. Two options are shown in the table: an acceptable practice and a preferred practice for reduced power consumption and improved EMC characteristics. If a module is not used in a system, and its inputs are grounded, it is important that the clock to the module is never enabled by setting the corresponding bit in the **RCGCx** register.

**Table 31-7. Connections for Unused Signals (212-Ball BGA)**

| Function | Signal Name | Pin Number | Acceptable Practice | Preferred Practice |
|---|---|---|---|---|
| ADC | VREFA+ | F4 | VDDA | VDDA |
| | VREFA- | G5 | GND | GND |
| Ethernet | EN0RXIN | V13 | NC | NC |
| | EN0RXIP | W13 | NC | NC |
| | EN0TXON | V14 | NC | NC |
| | EN0TXOP | V15 | NC | NC |
| | RBIAS | W15 | Connect to ground through 4.87K resistor | NC[a] |
| GPIO | PA1(UART0TX) | W3 | NC | GND[b] |
| | PA4 (SSI0XDAT0) | V4 | NC | GND[c] |
| | All unused GPIOs | - | NC | GND |

**Table 31-7. Connections for Unused Signals (212-Ball BGA) *(continued)***

| Function | Signal Name | Pin Number | Acceptable Practice | Preferred Practice |
|---|---|---|---|---|
| Hibernate | GNDX | R18 | GND | GND |
| | $\overline{\text{HIB}}$ | M17 | NC | NC |
| | VBAT | P19 | NC | VDD |
| | $\overline{\text{WAKE}}$ | U18 | NC | GND |
| | XOSC0 | T18 | NC | GND |
| | XOSC1 | T19 | NC | NC |
| No Connects | NC | See NC pin numbers in Table 31-3 on page 2048 | NC | NC |
| System Control | GNDX2 | D18 | GND | GND |
| | OSC0 | E19 | NC | GND |
| | OSC1 | D19 | NC | NC |
| | $\overline{\text{RST}}$ | P18 | VDD | Pull up as shown in Figure 5-1 on page 235 |
| USB | USB0DM / PL7 | B18 | NC | Pull-down to GND with 1K resistor[d] |
| | USB0DP /PL6 | C18 | NC | Pull-down to GND with 1K resistor[d] |

a. The internal Ethernet PHY should not be enabled when RBIAS is not connected. The ROM boot loader will enable the internal Ethernet PHY if no code is present in the flash and a 24-25 Mhz crystal is attached to the OSC0/OSC1 pins.

b. PA1 (UART0TX) may be enabled as an output by the ROM boot loader if no code is present in the flash and PA0 (UART0RX) receives a valid boot signature. Ensure that this condition will not occur if PA1 is to be connected directly to GND.

c. PA4 (SSI0XDAT0) may be enabled as an output by the ROM boot loader if no code is present in the flash and the SSI0X (PA2, PA3, PA5) receives a valid boot signature. Ensure that this condition will not occur if PA4 is to be connected directly to GND.

d. The ROM boot loader may configure these pins as USB pins if no code is present in the flash therefore they should not be directly connected to ground.

# 32 Electrical Characteristics

## 32.1 Maximum Ratings

The maximum ratings are the limits to which the device can be subjected without permanently damaging the device. Device reliability may be adversely affected by exposure to absolute-maximum ratings for extended periods.

**Note:** The device is not guaranteed to operate properly at the maximum ratings.

**Table 32-1. Absolute Maximum Ratings**

| Parameter | Parameter Name[a] | Value | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| $V_{DD}$ | $V_{DD}$ supply voltage | 0 | 4 | V |
| $V_{DDA}$ | $V_{DDA}$ supply voltage | 0 | 4 | V |
| $V_{BAT}$ | $V_{BAT}$ battery supply voltage | 0 | 4 | V |
| $V_{BATRMP}$ | $V_{BAT}$ battery supply voltage ramp time | 0 | 0.7 | V/µs |
| $V_{IN_GPIO}$ | Input voltage[b] | -0.3 | 4 | V |
| $I_{GPIOMAX}$ | Maximum current per output pin | - | 64 | mA |
| $T_S$ | Unpowered storage temperature range | -65 | 150 | °C |
| $T_{JMAX}$ | Maximum junction temperature | - | 125 | °C |

a. Voltages are measured with respect to GND.

b. Applies to static and dynamic signals including overshoot.

**Important:** This device contains circuitry to protect the I/Os against damage due to high-static voltages; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (see “Connections for Unused Signals” on page 2093).

**Table 32-2. ESD Absolute Maximum Ratings**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| Component-Level ESD Stress Voltage[a] | $V_{ESDHBM}$[b] | - | - | 2.0 | kV |
| | $V_{ESDCDM}$[c] | - | - | 500 | V |

a. Electrostatic discharge (ESD) is used to measure device sensitivity/immunity to damage caused by electrostatic discharges in device.

b. All pins are HBM compliant to 2 kV for all combinations as per JESD22-A114F, except for the following stress combinations:

- The Ethernet EN0RXIN, EN0TX0N, EN0RXIP, and EN0TXOP pins to each other.
- The GPIO pins PM4, PM5, PM6, and PM7 to other pins.

These exceptions are compliant to 500 V and do not require any special handling beyond typical ESD control procedures during assembly operations per JEDEC publication JEP155. Note that these pins do meet the 500 V CDM specification.

c. Level listed is the passing level per EIA-JEDEC JESD22-C101E. JEDEC document JEP157 states that 250 V CDM allows safe manufacturing with a standard ESD control process.

## 32.2 Operating Characteristics

### Table 32-3. Temperature Characteristics

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Ambient operating temperature range | $T_A$ | -40 to 85 (industrial temperature part)<br>-40 to 105 (extended temperature part) | °C |
| Junction operating temperature range | $T_J$ | -40 to 105 (industrial temperature part)<br>-40 to 125 (extended temperature part) | °C |

### Table 32-4. 212 BGA Power Dissipation[ab]

| Parameter | Parameter Name | $T_A$ | $T_J$ | Min | Max | Unit |
|---|---|---|---|---|---|---|
| $P_{DI}$ | Industrial temperature device power dissipation | 85 °C (industrial temperature part) | 125 °C (industrial temperature part) | - | 1018 | mW |
| $P_{DE}$ | Extended temperature device power dissipation | 105 °C (extended temperature part) | 125 °C (extended temperature part) | - | 509 | mW |

a. If the device exceeds the power dissipation value shown, then modifications such as heat sinks or fans must be used to conform to the limits shown.

b. A larger power dissipation allowance can be achieved by lowering $T_A$ as long as $T_{JMAX}$ shown in Table 32-1 on page 2095 is not exceeded.

### Table 32-5. Thermal Characteristics[a]

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Thermal resistance (junction to ambient)[b] | $\Theta_{JA}$ | 39.3 | °C/W |
| Thermal resistance (junction to board)[b] | $\Theta_{JB}$ | 18.6 | °C/W |
| Thermal resistance (junction to case)[b] | $\Theta_{JC}$ | 12.9 | °C/W |
| Thermal metric (junction to top of package) | $\Psi_{JT}$ | 0.2 | °C/W |
| Thermal metric (junction to board) | $\Psi_{JB}$ | 18.4 | °C/W |
| Junction temperature formula | $T_J$ | $T_C + (P \bullet \Psi_{JT})$[c]<br>$T_{PCB} + (P \bullet \Psi_{JB})$[d]<br>$T_A + (P \bullet \Theta_{JA})$[e]<br>$T_B + (P \bullet \Theta_{JB})$[fg] | °C |

a. For more details about thermal metrics and definitions, see the *Semiconductor and IC Package Thermal Metrics Application Report* (literature number SPRA953).

b. Junction to ambient thermal resistance ($\Theta_{JA}$), junction to board thermal resistance ($\Theta_{JB}$), and junction to case thermal resistance ($\Theta_{JC}$) numbers are determined by a package simulator.

c. $T_C$ is the case temperature and P is the device power consumption.

d. $T_{PCB}$ is the temperature of the board acquired by following the steps listed in the EAI/JESD 51-8 standard summarized in the *Semiconductor and IC Package Thermal Metrics Application Report* (literature number SPRA953). P is the device power consumption.

e. Because $\Theta_{JA}$ is highly variable and based on factors such as board design, chip/pad size, altitude, and external ambient temperature, it is recommended that equations containing $\Psi_{JT}$ and $\Psi_{JB}$ be used for best results.

f. $T_B$ is temperature of the board.

g. $\Theta_{JB}$ is not a pure reflection of the internal resistance of the package because it includes the resistance of the testing board and environment. It is recommended that equations containing $\Psi_{JT}$ and $\Psi_{JB}$ be used for best results.

## 32.3 Recommended Operating Conditions

The following sections describe the recommended DC operating conditions and GPIO operating characteristics for the device.

### 32.3.1 DC Operating Conditions

**Table 32-6. Recommended DC Operating Conditions**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ | $V_{DD}$ supply voltage | 2.97 | 3.3 | 3.63 | V |
| $V_{DDA}$[a] | $V_{DDA}$ supply voltage | 2.97 | 3.3 | 3.63 | V |
| $V_{DDC}$ | $V_{DDC}$ supply voltage, Run mode | 1.14 | 1.2 | 1.32 | V |
| $V_{DDCDS}$ | $V_{DDC}$ supply voltage, Deep-Sleep mode | 0.85 | - | 0.95 | V |

a. To ensure proper operation, $V_{DDA}$ must be powered up before $V_{DD}$ if sourced from different supplies, or connected to the same supply as $V_{DD}$. There is not a restriction on order for powering off.

### 32.3.2 Recommended GPIO Operating Characteristics

Two types of pads are provided on the device:

- Fast GPIO pads: These pads provide variable, programmable drive strength and optimized voltage output levels.
- Slow GPIO pads: These pads provide 2-mA drive strength and are designed to be sensitive to voltage inputs. The following GPIOs port pins are designed with Slow GPIO Pads:
  – PJ1

All other GPIOs have a fast GPIO pad-type.

**Note:** Port pins PL6 and PL7 operate as Fast GPIO pads, but have 4-mA drive capability only. GPIO register controls for drive strength, slew rate and open drain have no effect on these pins. The registers which have no effect are as follows: **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR12R**, **GPIOSLR**, and **GPIOODR**.

**Note:** Port pins PM[7:4] operate as Fast GPIO pads but support only 2-, 4-, 6-, and 8-mA drive capability. 10- and 12-mA drive are not supported. All standard GPIO register controls, except for the **GPIODR12R** register, apply to these port pins.

Table 32-7 on page 2097 detail the GPIO operating conditions for these two different pad types.

**Table 32-7. Recommended FAST GPIO Pad Operating Conditions**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{IH}$ | Fast GPIO high-level input voltage | 0.65 * $V_{DD}$ | - | 4 | V |
| $I_{IH}$ | Fast GPIO high-level input current [a] | - | - | 300 | nA |
| $V_{IL}$ | Fast GPIO low-level input voltage | 0 | - | 0.35 * $V_{DD}$ | V |
| $I_{IL}$ | Fast GPIO low-level input current[a] | - | - | -200 | nA |
| $V_{HYS}$ | Fast GPIO Input Hysteresis | 0.49 | - | - | V |
| $V_{OH}$ | Fast GPIO High-level output voltage | 2.4 | - | - | V |
| $V_{OL}$ | Fast GPIO Low-level output voltage | - | - | 0.40 | V |

**Table 32-7. Recommended FAST GPIO Pad Operating Conditions *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $I_{OH}$ | Fast GPIO High-level source current, $V_{OH}$=2.4 V[b] | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |
| | 4-mA Drive | 4.0 | - | - | mA |
| | 8-mA Drive | 8.0 | - | - | mA |
| | 10-mA Drive | 10.0 | - | - | mA |
| | 12-mA Drive | 12.0 | - | - | mA |
| $I_{OL}$ | Fast GPIO Low-level sink current, $V_{OL}$=0.4 V[b] | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |
| | 4-mA Drive | 4.0 | - | - | mA |
| | 8-mA Drive | 8.0 | - | - | mA |
| | 10-mA Drive | 10.0 | - | - | mA |
| | 12-mA Drive | 12.0 | - | - | mA |
| | 12-mA Drive overdriven to 18-mA | 18.0 | - | - | mA |

a. Output/pull-up/pull-down disabled; only input enabled

b. $I_O$ specifications reflect the maximum current where the corresponding output voltage meets the $V_{OH}/V_{OL}$ thresholds. $I_O$ current can exceed these limits (subject to absolute maximum ratings).

**Table 32-8. Recommended Slow GPIO Pad Operating Conditions**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{IH}$ | Slow GPIO high-level input voltage | 0.65 * $V_{DD}$ | - | 4 | V |
| $I_{IH}$ | Slow GPIO high-level input current [a] | - | - | 4.1 | nA |
| $V_{IL}$ | Slow GPIO low-level input voltage | 0 | - | 0.35 * $V_{DD}$ | V |
| $I_{IL}$ | Slow GPIO low-level input current[a] | - | - | -1 | nA |
| $V_{HYS}$ | Slow GPIO Input Hysteresis | 0.49 | - | - | V |
| $V_{OH}$ | Slow GPIO High-level output voltage | 2.4 | - | - | V |
| $V_{OL}$ | Slow GPIO Low-level output voltage | - | - | 0.4 | V |
| $I_{OH}$ | High-level source current, $V_{OH}$=2.4 V[b] | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |
| $I_{OL}$ | Low-level sink current, $V_{OL}$=0.4 V[b] | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |

a. Output/pull-up/pull-down disabled; only input enabled.

b. $I_O$ specifications reflect the maximum current where the corresponding output voltage meets the $V_{OH}/V_{OL}$ thresholds. $I_O$ current can exceed these limits (subject to absolute maximum ratings).

### 32.3.2.1 GPIO Current Restrictions

**Table 32-9. GPIO Current Restrictions[a]**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $I_{MAXL}$ | Cumulative maximum GPIO current per side, left[b] | - | - | 110 | mA |
| $I_{MAXB}$ | Cumulative maximum GPIO current per side, bottom[b] | - | - | 116 | mA |
| $I_{MAXR}$ | Cumulative maximum GPIO current per side, right[b] | - | - | 110 | mA |

**Table 32-9. GPIO Current Restrictions *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $I_{MAXT}$ | Cumulative maximum GPIO current per side, top[b] | - | - | 88 | mA |

a. Based on design simulations, not tested in production.

b. Sum of sink and source current for GPIOs as shown in Table 32-10 on page 2099.

**Table 32-10. Maximum GPIO Package Side Assignments**

| Side | GPIOs |
|---|---|
| Left | PB[6-7], PC[4-7], PD[0-3], PE[0-3], PH[0-7], PJ[6-7], PK[0-3], PQ[0-3,7], PR[0-5] |
| Bottom | PA[0-7], PF[0-7], PG[0-7], PK[4-7], PN[6-7], PR[3,6-7], PQ[5-6], PS[4-7], PT[0-1] |
| Right | PB[0-3], PJ[2-5], PL[0-5,6-7], PM[0-7], PT[2-3] |
| Top | PB[4-5], PC[0-3], PD[4-7], PE[4-7], PJ[0-1], PN[0-5],PP[0-7], PQ[4], PS[0-3], |

### *I/O Reliability*

For typical continuous drive applications, I/O pins configured between 2 mA and 12 mA and operating at -40 to 85°C, meet the standard 10-year lifetime reliability. If a continuous current sink of 18 mA is required, then operation is limited to 0 to 75°C in order to meet the standard 10-year reliability.

At 105°C, I/O configured for continuous drive meet the standard 2.5 year lifetime reliability.

In typical switching applications (40% switch rate) operating at -40 to 85°C, all I/O configurations except 2 mA meet the standard 10-year lifetime reliability with 50-pF loading. By limiting the capacitive loading to 20 pF for an I/O configured to 2 mA, the 10-year lifetime reliability can be met at -40 to 85°C.

In typical switching applications (40% switch rate) operating at 105°C, all I/O configurations except 2 mA meet the standard 2.5-year lifetime reliability. By reducing the capacitive loading to 20 pF with a typical switching rate at 105°C, a 2-mA I/O configuration meets a 2.5 year lifetime reliability.

## 32.4 Load Conditions

Table 32-11 on page 2100 contains the load conditions used for timing measurements.

**Figure 32-1. Load Conditions**

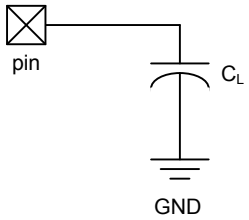



**Table 32-11. Load Conditions**

| Signals | Load Value ($C_L$) |
|---|---|
| LCDDATA[23:0] | 25 pF |
| EPI0S[35:0] SDRAM interface | 30 pF |
| EPI0S[35:0] General-Purpose interface | |
| EPI0S[35:0] Host-Bus interface | |
| EPI0S[35:0] PSRAM interface | 40 pF |
| All other digital I/O signals | 50 pF |

## 32.5 JTAG and Boundary Scan

**Table 32-12. JTAG Characteristics**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| J1 | $F_{TCK}$ | TCK operational clock frequency[a] | 0 | - | 10 | MHz |
| J2 | $T_{TCK}$ | TCK operational clock period | 100 | - | - | ns |
| J3 | $T_{TCK_LOW}$ | TCK clock Low time | - | $t_{TCK}/2$ | - | ns |
| J4 | $T_{TCK_HIGH}$ | TCK clock High time | - | $t_{TCK}/2$ | - | ns |
| J5 | $T_{TCK_R}$ | TCK rise time | 0 | - | 10 | ns |
| J6 | $T_{TCK_F}$ | TCK fall time | 0 | - | 10 | ns |
| J7 | $T_{TMS_SU}$ | TMS setup time to TCK rise | 8 | - | - | ns |
| J8 | $T_{TMS_HLD}$ | TMS hold time from TCK rise | 4 | - | - | ns |
| J9 | $T_{TDI_SU}$ | TDI setup time to TCK rise | 18 | - | - | ns |
| J10 | $T_{TDI_HLD}$ | TDI hold time from TCK rise | 4 | - | - | ns |
| J11 | $T_{TDO_ZDV}$ | TCK fall to Data Valid from High-Z, 2-mA drive | - | 13 | 35 | ns |
| | | TCK fall to Data Valid from High-Z, 4-mA drive | | 9 | 26 | ns |
| | | TCK fall to Data Valid from High-Z, 8-mA drive | | 8 | 26 | ns |
| | | TCK fall to Data Valid from High-Z, 8-mA drive with slew rate control | | 10 | 29 | ns |
| | | TCK fall to Data Valid from High-Z, 10-mA drive | | 11 | 13 | ns |
| | | TCK fall to Data Valid from High-Z, 12-mA drive | | 11 | 14 | ns |
| J12 | $T_{TDO_DV}$ | TCK fall to Data Valid from Data Valid, 2-mA drive | - | 14 | 20 | ns |
| | | TCK fall to Data Valid from Data Valid, 4-mA drive | | 10 | 26 | ns |
| | | TCK fall to Data Valid from Data Valid, 8-mA drive | | 8 | 21 | ns |
| | | TCK fall to Data Valid from Data Valid, 8-mA drive with slew rate control | | 10 | 26 | ns |
| | | TCK fall to Data Valid from Data Valid, 10-mA drive | | 12 | 14 | ns |
| | | TCK fall to Data Valid from Data Valid, 12-mA drive | | 12 | 15 | ns |
| J13 | $T_{TDO_DVZ}$ | TCK fall to High-Z from Data Valid, 2-mA drive | - | 7 | 16 | ns |
| | | TCK fall to High-Z from Data Valid, 4-mA drive | | 7 | 16 | ns |
| | | TCK fall to High-Z from Data Valid, 8-mA drive | | 7 | 16 | ns |
| | | TCK fall to High-Z from Data Valid, 8-mA drive with slew rate control | | 8 | 19 | ns |
| | | TCK fall to High-Z from Data Valid, 10-mA drive | | 20 | 22 | ns |
| | | TCK fall to High-Z from Data Valid, 12-mA drive | | 20 | 25 | ns |

a. A ratio of at least 8:1 must be kept between the system clock and TCK.

**Figure 32-2. JTAG Test Clock Input Timing**

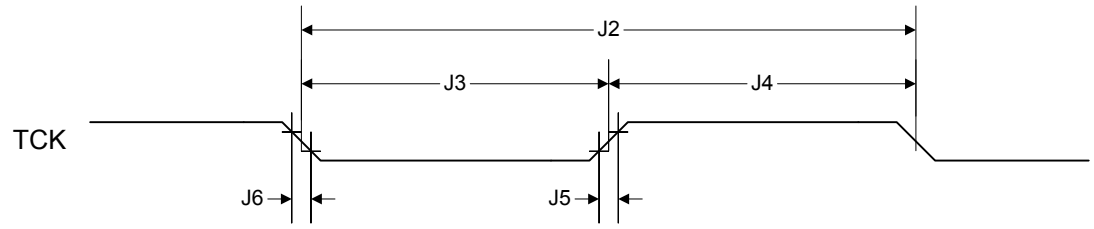
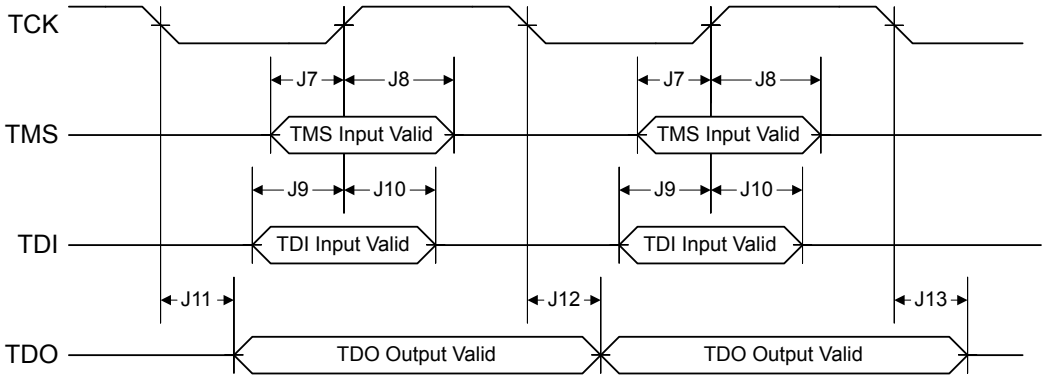



**Figure 32-3. JTAG Test Access Port (TAP) Timing**



## 32.6 Power and Brown-Out

**Table 32-13. Power and Brown-Out Levels**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| P1 | $T_{VDDA_RISE}$ | Analog Supply voltage ($V_{DDA}$) rise time | - | - | ∞ | µs |
| P2 | $T_{VDD_RISE}$ | I/O Supply voltage ($V_{DD}$) rise time | - | - | ∞ | µs |
| P3 | $T_{VDDC_RISE}$[a] | Core Supply Voltage ($V_{DDC}$) rise time | 10 | - | 150 | µs |
| P4 | $V_{POR}$ | Power-On Reset Threshold (Rising Edge) | 1.98 | 2.35 | 2.72 | V |
| | | Power-On Reset Threshold (Falling Edge) | 1.84 | 2.20 | 2.56 | V |
| | | Power-On Reset Hysteresis | 0.06 | 0.15 | 0.24 | V |
| P5 | $V_{DDA_POK}$ | $V_{DDA}$ Power-OK Threshold (Rising Edge) | 2.67 | 2.82 | 2.97 | V |
| P6 | $V_{DDA_BOR0}$ | $V_{DDA}$ Brown-Out Reset Threshold | 2.71 | 2.80 | 2.89 | V |
| P7 | $V_{DD_POK}$ | $V_{DD}$ Power-OK Threshold (Rising Edge) | 2.65 | 2.80 | 2.90 | V |
| | | $V_{DD}$ Power-OK Threshold (Falling Edge) | 2.67 | 2.76 | 2.85 | V |
| P8 | $V_{DD_BOR0}$ | $V_{DD}$ Brown-Out Reset Threshold | 2.77 | 2.86 | 2.95 | V |
| P9 | $V_{DDC_POK}$ | $V_{DDC}$ Power-OK Threshold (Rising Edge) | 0.85 | 0.95 | 1.10 | V |
| | | $V_{DDC}$ Power-OK Threshold (Falling Edge) | 0.71 | 0.80 | 0.85 | V |

a. The MIN and MAX values are based on an external filter capacitor load within the range of $C_{LDO}$. Please refer to "On-Chip Low Drop-Out (LDO) Regulator" on page 2111 for the $C_{LDO}$ value.

### 32.6.1 $V_{DDA}$ Levels

The $V_{DDA}$ supply has three monitors:

- Power-On Reset (POR)
- Power-OK (POK)
- Brown Out Reset (BOR)

The POR monitor is used to keep the analog circuitry in reset until the $V_{DDA}$ supply has reached the correct range for the analog circuitry to begin operating. The POK monitor is used to keep the digital circuitry in reset until the $V_{DDA}$ power supply is at an acceptable operational level. The digital reset is only released when the Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are in operational ranges. The BOR monitor is used to generate a reset to the device or assert an interrupt if the $V_{DDA}$ supply drops below its operational range.

**Note:** $V_{DDA}$ BOR and $V_{DD}$ BOR events are a combined BOR to the system logic, such that if either BOR event occurs, the following bits are affected:

- BORRIS bit in the **Raw Interrupt Status (RIS**) register, System Control offset 0x050. See page 273.
- BORMIS bit in the **Masked Interrupt Status and Clear (MISC)** register, System Control offset 0x058. This bit is set only if the BORIM bit in the **Interrupt Mask Control (IMC)** register has been set. See page 275 and page 277.

- BOR bit in the **Reset Cause (RESC)** register, System Control offset 0x05C. This bit is set only if either of the BOR events have been configured to initiate a reset. See page 279.

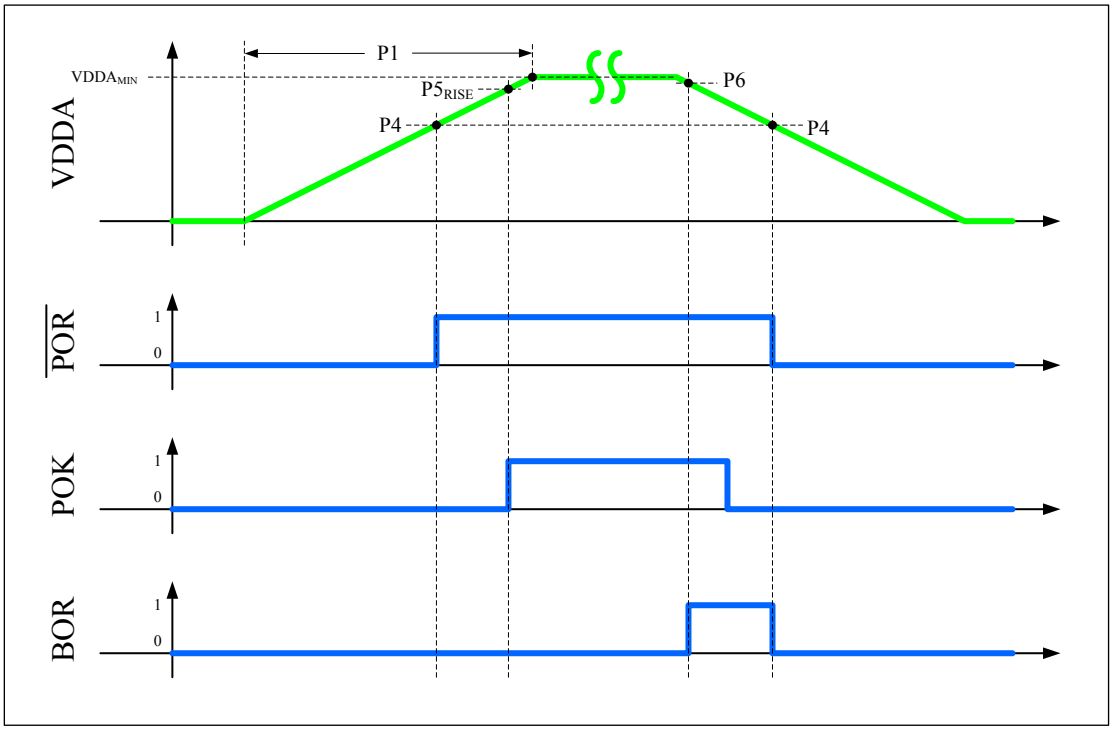
In addition, the following bits control both BOR events:

- BORIM bit in the **Interrupt Mask Control (IMC)** register, System Control offset 0x054.
- VDDA_UBOR0 and VDD_UBOR0 bits in the **Power-Temperature Cause (PWRTC)** register.

Please refer to “System Control” on page 231 for more information on how to configure these registers.

Figure 32-4 on page 2104 shows the relationship between $V_{DDA}$, POK, POR and a BOR event.

**Figure 32-4. Power and Brown-Out Assertions vs $V_{DDA}$ Levels**




## 32.6.2 $V_{DD}$ Levels

The $V_{DD}$ supply has two monitors:

- Power-OK (POK)
- Brown Out Reset (BOR)

The POK monitor is used to keep the digital circuitry in reset until the $V_{DD}$ power supply is at an acceptable operational level. The digital reset is only released when the Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are

in operational ranges. The BOR monitor is used to generate a reset to the device or assert an interrupt if the $V_{DD}$ supply drops below its operational range.

**Note:** $V_{DDA}$ BOR and $V_{DD}$ BOR events are a combined BOR to the system logic, such that if either BOR event occurs, the following bits are affected:

- BORRIS bit in the **Raw Interrupt Status (RIS)** register, System Control offset 0x050. See page 273.
- BORMIS bit in the **Masked Interrupt Status and Clear (MISC)** register, System Control offset 0x058. This bit is set only if the BORIM bit in the **Interrupt Mask Control (IMC)** register has been set. See page 275 and page 277.
- BOR bit in the **Reset Cause (RESC)** register, System Control offset 0x05C. This bit is set only if either of the BOR events have been configured to initiate a reset. See page 279.

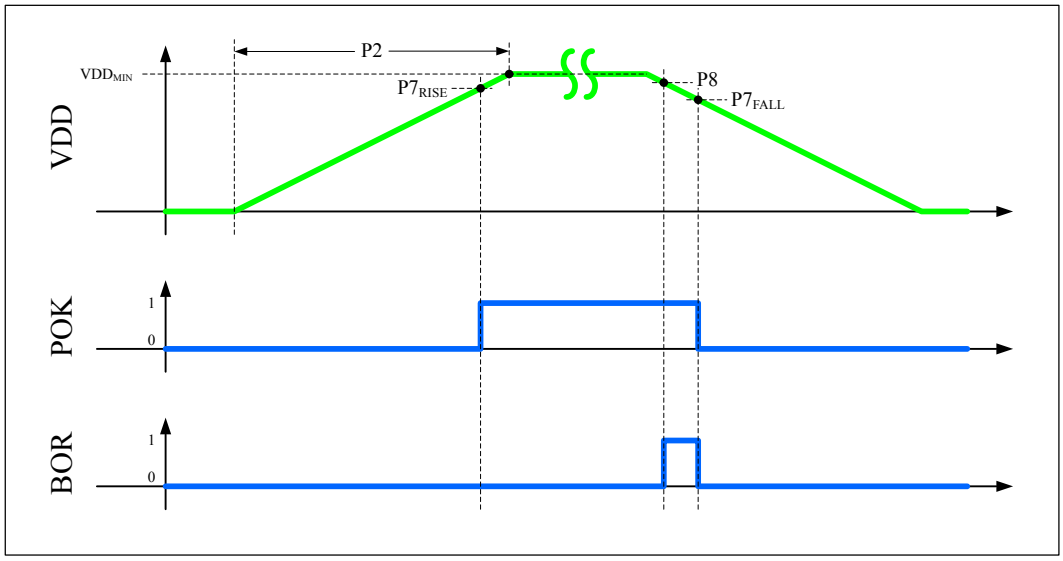
In addition, the following bits control both BOR events:

- BORIM bit in the **Interrupt Mask Control (IMC)** register, System Control offset 0x054.
- VDDA_UBOR0 and VDD_UBOR0 bits in the **Power-Temperature Cause (PWRTC)** register.

Please refer to “System Control” on page 231 for more information on how to configure these registers.

Figure 32-5 on page 2105 shows the relationship between $V_{DD}$, POK, POR and a BOR event.

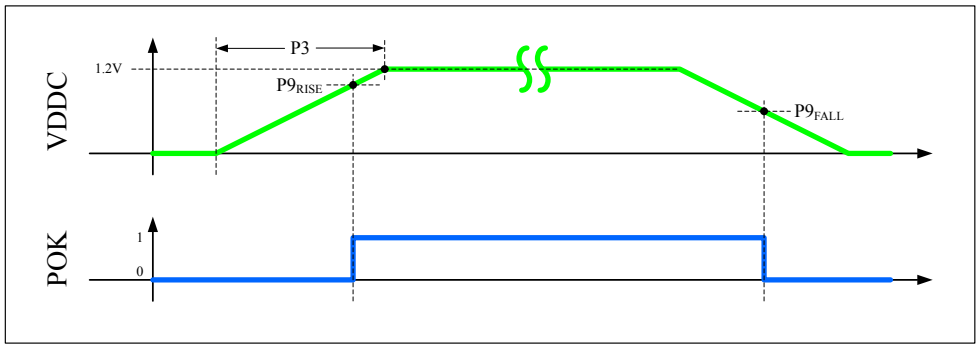
**Figure 32-5. Power and Brown-Out Assertions vs $V_{DD}$ Levels**




## 32.6.3 $V_{DDC}$ Levels

The $V_{DDC}$ supply has one monitor, the Power-OK (POK). The POK monitor is used to keep the digital circuitry in reset until the $V_{DDC}$ power supply is at an acceptable operational level. The digital reset is only released when the Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are in operational ranges. Figure 32-6 on page 2106 shows the relationship between POK and $V_{DDC}$.

**Figure 32-6. POK Assertion vs $V_{DDC}$**

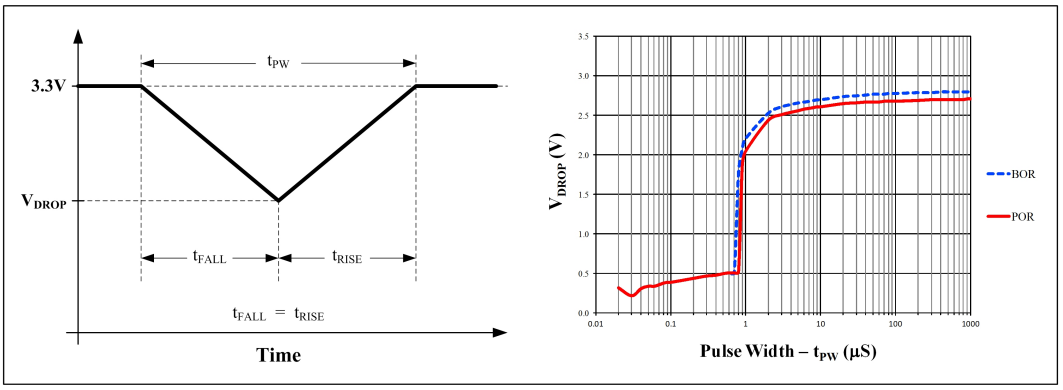



## 32.6.4 Response

### 32.6.4.1 $V_{DD}$ Glitch Response

Figure 32-7 on page 2106 shows the response of the BOR and the POR circuit to glitches on the VDD supply.

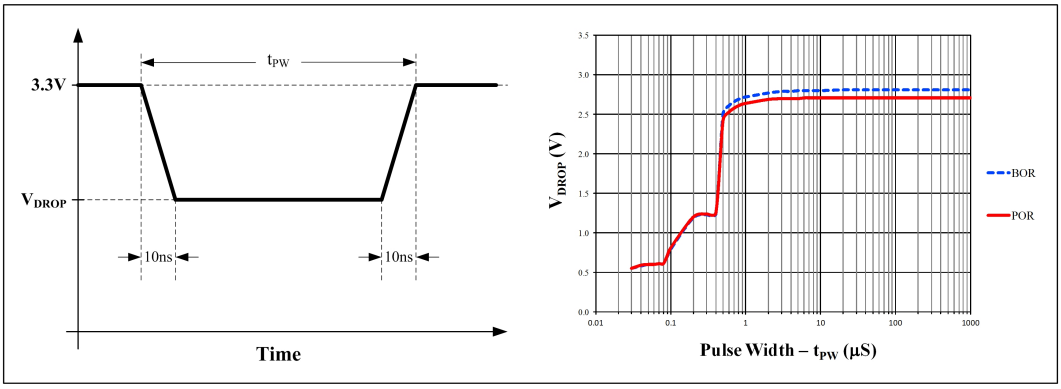
**Figure 32-7. POR-BOR $V_{DD}$ Glitch Response**




### 32.6.4.2 $V_{DD}$ Droop Response

Figure 32-8 on page 2107 shows the response of the BOR and the POR monitors to a drop on the VDD supply.

**Figure 32-8. POR-BOR $V_{DD}$ Droop Response**



## 32.7 Reset

**Table 32-14. Reset Characteristics**[a]

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| R1 | $T_{DPORDLY}$[b] | Digital POR to Internal Reset assertion delay[c] | 0.44 | - | 126 | µs |
| R2 | $T_{IRTOUT}$[bd] | Standard Internal Reset time | - | 14 | 16 | ms |
| | | Internal Reset time with recovery code repair (program or erase)[e] | 24.4 | - | 6400[f] | ms |
| R3 | $T_{BOR0DLY}$[b] | BOR0 to Internal Reset assertion delay[c] | 0.44 | - | 125 | µs |
| R4 | $T_{RSTMIN}$ | Minimum $\overline{RST}$ pulse width | - | 0.25[g]/100[h] | - | µs |
| R5 | $T_{IRHWDLY}$ | $\overline{RST}$ to Internal Reset assertion delay | - | 0.85 | - | µs |
| R6 | $T_{IRSWR}$[b] | Internal reset timeout after software-initiated system reset | - | 2.44 | - | µs |
| R7 | $T_{IRWDR}$[b] | Internal reset timeout after Watchdog reset | - | 2.44 | - | µs |
| R8 | $T_{IRMFR}$[b] | Internal reset timeout after MOSC failure reset | - | 2.44 | - | µs |

a. Minimum timings are for reset assertion using PIOSC as a clock source. Maximum timings are for reset assertion using LFIOSC in Deep-Sleep Operation.

b. These values are based on simulation.

c. Timing values are dependent on the $V_{DD}$ power-down ramp rate.

d. This is the delay from the time POR is released until the reset vector is fetched.

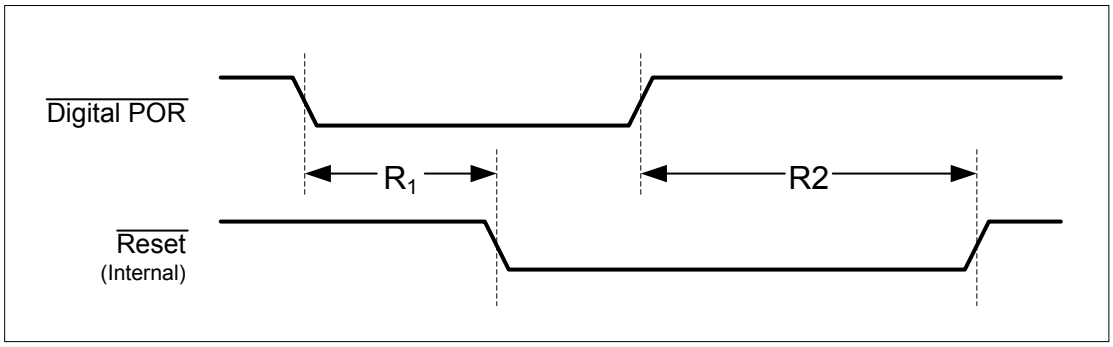
e. This parameter applies only in situations where a power-loss or brown-out event occurs during an EEPROM program or erase operation, and EEPROM needs to be repaired (which is a rare case). For all other sequences, there is no impact to normal Power-On Reset (POR) timing. This delay is in addition to other POR delays.

f. This value represents the maximum internal reset time when the EEPROM reaches its endurance limit.

g. Standard operation.

h. Deep-Sleep operation with PIOSC powered down.

**Figure 32-9. Digital Power-On Reset Timing**




The digital Power-On Reset is only released when the analog Power-On Reset has deasserted and all of the Power-OK monitors for each of the supplies indicate that power levels are in operational ranges.

**Figure 32-10. Brown-Out Reset Timing**

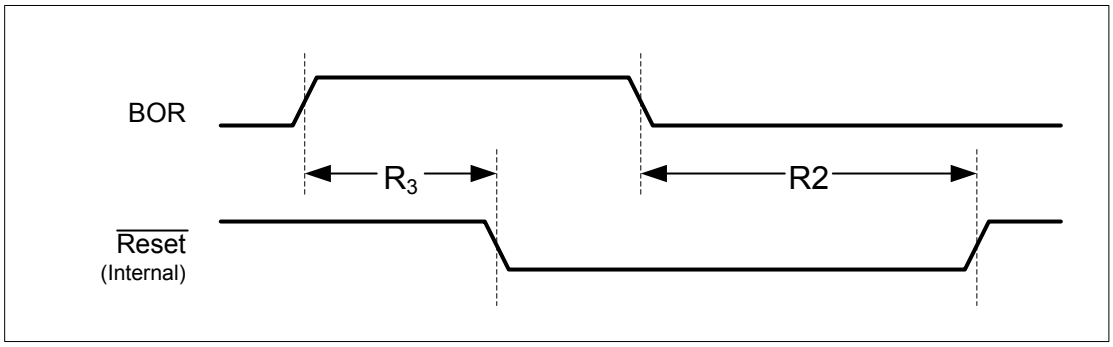



**Figure 32-11. External Reset Timing ($\overline{RST}$)**

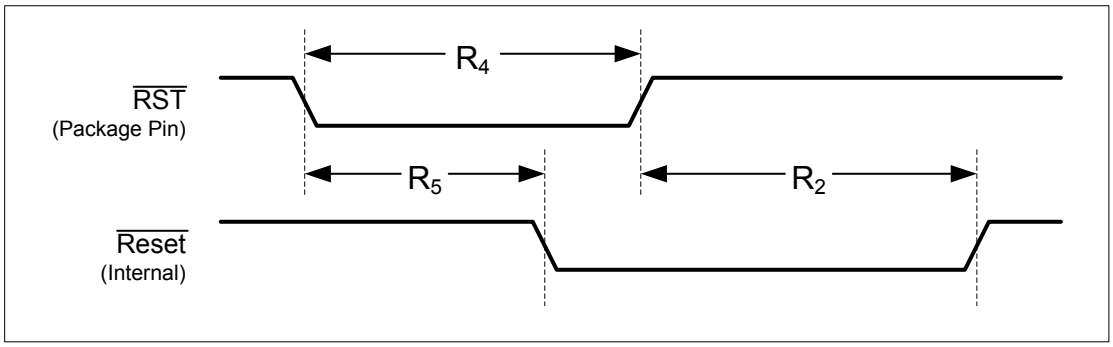



**Figure 32-12. Software Reset Timing**

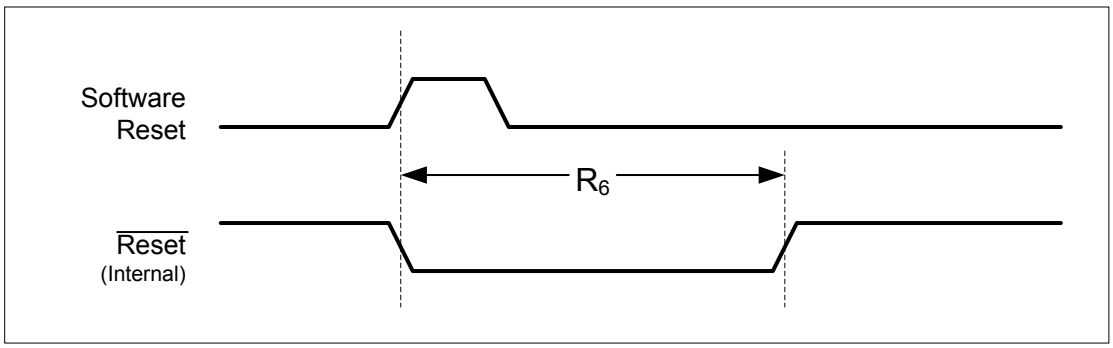



**Figure 32-13. Watchdog Reset Timing**

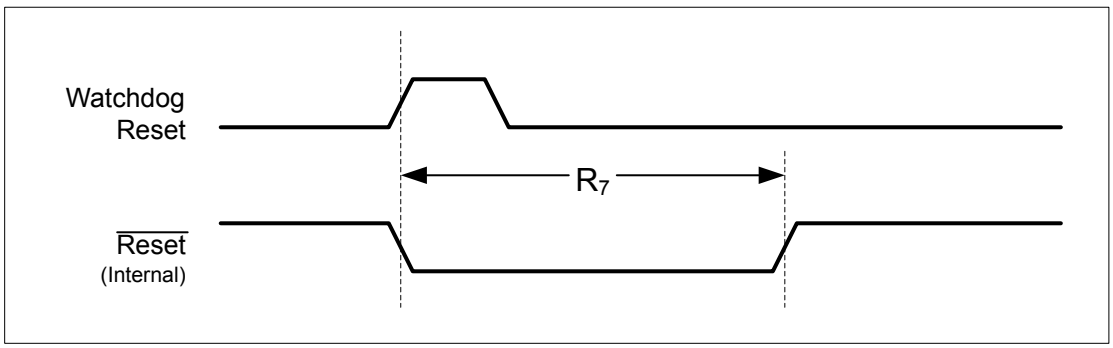
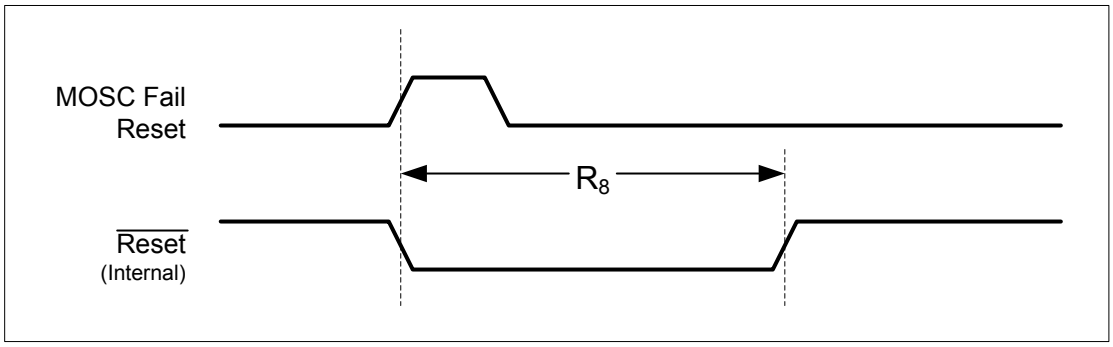


**Figure 32-14. MOSC Failure Reset Timing**



## 32.8 On-Chip Low Drop-Out (LDO) Regulator

**Table 32-15. LDO Regulator Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $C_{LDO}$ | External filter capacitor size for internal power supply[a] | 2.5 | - | 4.0 | µF |
| ESR | Filter capacitor equivalent series resistance | 0 | - | 100 | mΩ |
| ESL | Filter capacitor equivalent series inductance | - | - | 0.5 | nH |
| $V_{LDO}$ | LDO output voltage, Run mode | 1.13 | 1.2 | 1.27 | V |
| $I_{INRUSH}$ | Inrush current | 50 | - | 250 | mA |

a. The capacitor should be connected as close as possible to pin E10.

# 32.9 Clocks

The following sections provide specifications on the various clock sources and mode.

## 32.9.1 PLL Specifications

The following tables provide specifications for using the PLL.

**Note:** If the integrated Ethernet PHY is used in this device, then $F_{REF_XTAL}$ and $F_{REF_EXT}$ are required to be 25 MHz.

**Table 32-16. Phase Locked Loop (PLL) Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{REF_XTAL}$ | Crystal reference | 5 | - | 25 | MHz |
| $F_{REF_EXT}$ | External clock reference | 5 | - | 25 | MHz |
| $F_{PLLR}$ | PLL VCO frequency at 1.2V[a] | - | - | 480 | MHz |
| $F_{PLLS}$ | PLL VCO frequency at 0.9V[b] | - | - | 480 | MHz |
| $T_{READY}$ | PLL lock time (enabling the PLL) when PLL is transitioning from power down to power up | - | - | 512 * (reference clock period) | µs |
| | PLL lock time when the PLL VCO frequency is changed (PLL is already enabled) | - | - | 128 * (reference clock period) | µs |
| | PLL lock time, changing the OSCSRC between MOSC and PIOSC | - | - | 128 * (reference clock period) | µs |

a. PLL frequency is manually calculated using the values in the PLLFREQ0 and PLLFREQ1 registers.

b. If the LDO is dropped to 0.9V, the system must be run 1/4 of the maximum frequency at most. The Q value in the **PLLFREQ1** register must be set to 0x3 rather than using the PSYSDIV field in the **RSCLKCFG** register for the divisor.

### 32.9.1.1 PLL Configuration

The PLL is disabled by default during power-on reset and is enabled later by software if required. Software specifies the output divisor to set the system clock frequency and enables the PLL to drive the output. The PLL is controlled using the **PLLFREQ0**, **PLLFREQ1** and **PLLSTAT** registers. Changes made to these registers do not become active until after the NEWFREQ bit in the **RSCLKCFG** register is enabled.

The clock source for the main PLL is selected by configuring the PLLSRC field in the **Run and Sleep Clock Configuration (RSCLKCFG)** register.

The PLL allows for the generation of system clock frequencies in excess of the reference clock provided. The reference clocks for the PLL are the PIOSC and the MOSC. The PLL is controlled by two registers, **PLLFREQ0** and **PLLFREQ1**. The PLL VCO frequency ($f_{VCO}$) is determined through the following calculation:

$f_{VCO} = f_{IN} * MDIV$

where

$f_{IN} = f_{XTAL}/(Q+1)(N+1)$ or $f_{PIOSC}/(Q+1)(N+1)$

$MDIV = MINT + (MFRAC / 1024)$

The Q and N values are programmed in the **PLLFREQ1** register. Note that to reduce jitter, MFRAC should be programmed to 0x0.

When the PLL is active, the system clock frequency (SysClk) is calculated using the following equation:

SysClk = $f_{VCO}$/ (PSYSDIV + 1)

The PLL system divisor factor (PSYSDIV) determines the value of the system clock. Table 5-6 on page 249 shows how the system divisor encodings affect the system clock frequency when the $f_{VCO}$ = 480 MHz.

**Table 32-17. System Divisor Factors for $f_{vco}$=480 MHz**

| System Clock (SYSCLK) (MHz) | $f_{VCO}$ (MHz)= 480 MHz |
|---|---|
| | System Divisors (PSYSDIV +1)[a] |
| 120 | 4 |
| 60 | 8 |
| 48 | 10 |
| 30 | 16 |
| 24 | 20 |
| 12 | 40 |
| 6 | 80 |

a. The use of non-integer divisors introduce additional jitter which may affect interface performance.

If the main oscillator provides the clock reference to the PLL, the translation provided by hardware and used to program the PLL is available for software in the **PLL Frequency n (PLLFREQn)** registers (see page 304). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency. Table 5-7 on page 250 shows the actual PLL frequency and error for a given crystal choice.

Table 5-7 on page 250 provides examples of the programming expected for the **PLLFREQ0** and **PLLFREQ1** registers. The first column specifies the input crystal frequency and the last column displays the PLL frequency given the values of MINT and N, when Q=0.

**Table 32-18. Actual PLL Frequency**[a]

| Crystal Frequency (MHz) | MINT (Decimal Value) | MINT (Hexadecimal Value) | N | Reference Frequency (MHz)[b] | PLL Frequency (MHz) |
|---|---|---|---|---|---|
| 5 | 64 | 0x40 | 0x0 | 5 | 320 |
| 6 | 160 | 0x35 | 0x2 | 2 | 320 |
| 8 | 40 | 0x28 | 0x0 | 8 | 320 |
| 10 | 32 | 0x20 | 0x0 | 10 | 320 |
| 12 | 80 | 0x50 | 0x2 | 4 | 320 |
| 16 | 20 | 0x14 | 0x0 | 16 | 320 |
| 18 | 160 | 0xA0 | 0x8 | 2 | 320 |
| 20 | 16 | 0x10 | 0x0 | 20 | 320 |
| 24 | 40 | 0x28 | 0x2 | 8 | 320 |
| 25 | 64 | 0x40 | 0x4 | 5 | 320 |
| 5 | 96 | 0x60 | 0x0 | 5 | 480 |
| 6 | 80 | 0x50 | 0x0 | 6 | 480 |
| 8 | 60 | 0x3C | 0x0 | 8 | 480 |

**Table 32-18. Actual PLL Frequency *(continued)***

| Crystal Frequency (MHz) | MINT (Decimal Value) | MINT (Hexadecimal Value) | N | Reference Frequency (MHz)[b] | PLL Frequency (MHz) |
|---|---|---|---|---|---|
| 10 | 48 | 0x30 | 0x0 | 10 | 480 |
| 12 | 40 | 0x28 | 0x0 | 12 | 480 |
| 16 | 30 | 0x1E | 0x0 | 16 | 480 |
| 18 | 80 | 0x50 | 0x2 | 6 | 480 |
| 20 | 24 | 0x18 | 0x0 | 20 | 480 |
| 24 | 20 | 0x14 | 0x0 | 24 | 480 |
| 25 | 96 | 0x60 | 0x4 | 5 | 480 |

a. For all examples listed, Q=0

b. For a given crystal frequency, N should be chosen such that the reference frequency is within 4 to 30 MHz.

## 32.9.2 PIOSC Specifications

**Table 32-19. PIOSC Clock Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{PIOSC}$ | Factory calibration, 0 to +105°C:<br>Internal 16-MHz precision oscillator frequency variance across voltage and temperature range when factory calibration is used | - | - | ±4.5% | - |
| | Factory calibration, -40°C to <0°C | - | - | ±10% | - |
| | Recalibration:<br>Internal 16-MHz precision oscillator frequency variance when recalibration is used at a specific temperature | - | - | ±1% | - |
| $T_{START}$ | PIOSC startup time[a] | - | - | 1 | µs |

a. PIOSC startup time is part of reset and is included in the internal reset timeout value ($T_{IRTOUT}$) given in Table 32-14 on page 2108. Note that the $T_{START}$ value is based on simulation.

## 32.9.3 Low-Frequency Internal Oscillator Specifications

**Table 32-20. Low-Frequency Oscillator Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{LFIOSC}$ | Internal low-frequency oscillator frequency | 10 | 33 | 75 | KHz |

## 32.9.4 Hibernation Clock Source Specifications

**Table 32-21. Hibernation Internal Low Frequency Oscillator Clock Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{HIBLFIOSC}$ | Internal Low Frequency Hibernation oscillator frequency | 10 | 33 | 90 | KHz |

**Table 32-22. Hibernation External Oscillator (XOSC) Input Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{HIBXOSC}$[a] | Parallel resonance frequency | - | 32.768 | - | KHz |
| $C_1$, $C_2$ | External load capacitance on `XOSC0`, `XOSC1` pins[b] | 12 | - | 24 | pF |
| $C_{PKG}$ | Device package stray shunt capacitance[b] | - | 0.5 | - | pF |

**Table 32-22. Hibernation External Oscillator (XOSC) Input Characteristics *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $C_{PCB}$ | PCB stray shunt capacitance[b] | - | 0.5 | - | pF |
| $C_{SHUNT}$ | Total shunt capacitance[b] | - | - | 4 | pF |
| ESR | Crystal effective series resistance, OSCDRV = 0[c] | - | - | 50 | kΩ |
| | Crystal effective series resistance, OSCDRV = 1[c] | - | - | 75 | kΩ |
| DL | Oscillator output drive level | - | - | 0.25 | µW |
| $T_{START}$ | Oscillator startup time, when using a crystal[d] | - | 600 | 1500[e] | ms |
| $V_{IH}$[f] | CMOS input high level, when using an external oscillator with Supply > 3.3 V | 2.64 | - | - | V |
| | CMOS input high level, when using an external oscillator with 1.8 V ≤ Supply ≤ 3.3 V | 0.8 * Supply | - | - | V |
| $V_{IL}$[f] | CMOS input low level, when using an external oscillator with 1.8 V ≤ Supply ≤ 3.63 V | - | - | 0.2 * Supply | V |
| $V_{HYS}$[f] | CMOS input buffer hysteresis, when using an external oscillator with 1.8 V ≤ Supply ≤ 3.63 V | 360 | 960 | 1390 | mV |
| $DC_{HIBOSC_EXT}$ | External single-Ended (Bypass) reference duty cycle | 30 | - | 70 | % |

a. The HIB XOSC pins are non-failsafe and must follow the limits detailed in “Non-Power I/O Pins” on page 2128.

b. See information below table.

c. Crystal ESR specified by crystal manufacturer.

d. Oscillator startup time is specified from the time the oscillator is enabled to when it reaches a stable point of oscillation such that the internal clock is valid.

e. Only valid for recommended supply conditions. Measured with OSCDRV bit set (high drive strength enabled, 24 pF).

f. Specification is relative to the larger of $V_{DD}$ or $V_{BAT}$.

The load capacitors added on the board, $C_1$ and $C_2$, should be chosen such that the following equation is satisfied (see Table 32-22 on page 2114 for typical values).

- $C_L$ = load capacitance specified by crystal manufacturer
- $C_L = (C_1 * C_2)/(C_1+C_2) + C_{PKG} + C_{PCB}$
- $C_{SHUNT} = C_{PKG} + C_{PCB} + C_0$ (total shunt capacitance seen across XOSC0, XOSC1)
- $C_{PKG}$, $C_{PCB}$ as measured across the XOSC0, XOSC1 pins excluding the crystal
- Clear the OSCDRV bit in the **Hibernation Control (HIBCTL)** register for $C_{1,2} \leq 18$ pF; set the OSCDRV bit for $C_{1,2} > 18$ pF.
- $C_0$ = Shunt capacitance of crystal specified by the crystal manufacturer

## 32.9.5 Main Oscillator Specifications

**Table 32-23. Main Oscillator Input Characteristics[a]**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{MOSC}$ | Parallel resonance frequency | 4[b] | - | 25 | MHz |
| $F_{REF_XTAL_BYPASS}$ | External clock reference (PLL in BYPASS mode) | 0 | - | 120 | MHz |
| $C_1$, $C_2$ | External load capacitance on OSC0, OSC1 pins[c] | 12 | - | 24 | pF |
| $C_{PKG}$ | Device package stray shunt capacitance[c] | - | 0.5 | - | pF |

**Table 32-23. Main Oscillator Input Characteristics *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $C_{PCB}$ | PCB stray shunt capacitance[c] | - | 0.5 | - | pF |
| $C_{SHUNT}$ | Total shunt capacitance[c] | - | - | 4 | pF |
| ESR | Crystal effective series resistance, 4 MHz[de] | - | - | 300 | Ω |
| | Crystal effective series resistance, 6 MHz[de] | - | - | 200 | Ω |
| | Crystal effective series resistance, 8 MHz[de] | - | - | 130 | Ω |
| | Crystal effective series resistance, 12 MHz[de] | - | - | 120 | Ω |
| | Crystal effective series resistance, 16 MHz[de] | - | - | 100 | Ω |
| | Crystal effective series resistance, 25 MHz[de] | - | - | 50 | Ω |
| DL | Oscillator output drive level[f] | - | $OSC_{PWR}$ | - | mW |
| $T_{START}$ | Oscillator startup time, when using a crystal[g] | - | - | 18 | ms |
| $V_{IH}$ | CMOS input high level, when using an external oscillator | 0.65 * $V_{DD}$ | - | $V_{DD}$ | V |
| $V_{IL}$ | CMOS input low level, when using an external oscillator | GND | - | 0.35 * $V_{DD}$ | V |
| $V_{HYS}$ | CMOS input buffer hysteresis, when using an external oscillator | 150 | - | - | mV |
| $DC_{OSC_EXT}$ | External clock reference duty cycle | 45 | - | 55 | % |

a. Refer to Table 32-50 on page 2148 and Table 32-51 on page 2148 for additional Ethernet crystal requirements.

b. 5 MHz is the minimum when using the PLL.

c. See information below table.

d. Crystal ESR specified by crystal manufacturer.

e. Crystal vendors can be contacted to confirm these specifications are met for a specific crystal part number if the vendors generic crystal datasheet show limits outside of these specifications.

f. $OSC_{PWR} = (2 * pi * F_P * C_L * 2.5)^2 * ESR / 2$. An estimation of the typical power delivered to the crystal is based on the $C_L$, $F_P$ and ESR parameters of the crystal in the circuit as calculated by the $OSC_{PWR}$ equation. Ensure that the value calculated for $OSC_{PWR}$ does not exceed the crystal's drive-level maximum.

g. Oscillator startup time is specified from the time the oscillator is enabled to when it reaches a stable point of oscillation such that the internal clock is valid.

The load capacitors added on the board, $C_1$ and $C_2$, should be chosen such that the following equation is satisfied (see Table 32-23 on page 2115 for typical values and Table 32-24 on page 2117 for detailed crystal parameter information).

- $C_L$ = load capacitance specified by crystal manufacturer
- $C_L = (C_1*C_2)/(C_1+C_2) + C_{SHUNT}$
- $C_{SHUNT} = C_0 + C_{PKG} + C_{PCB}$ (total shunt capacitance seen across OSC0, OSC1 crystal inputs)
- $C_{PKG}$, $C_{PCB}$ = the mutual caps as measured across the OSC0,OSC1 pins excluding the crystal.
- $C_0$ = Shunt capacitance of crystal specified by the crystal manufacturer

Table 32-24 on page 2117 lists part numbers of crystals that have been simulated and confirmed to operate within the specifications in Table 32-23 on page 2115. Other crystals that have nearly identical crystal parameters can be expected to work as well.

In the table below, the crystal parameters labeled C0, C1 and L1 are values that are obtained from the crystal manufacturer. These numbers are usually a result of testing a relevant batch of crystals

on a network analyzer. The parameters labeled ESR, DL and $C_L$ are maximum numbers usually available in the data sheet for a crystal.

The table also includes three columns of Recommended Component Values. These values apply to system board components. $C_1$ and $C_2$ are the values in pico Farads of the load capacitors that should be put on each leg of the crystal pins to ensure oscillation at the correct frequency. Rs is the value in kΩ of a resistor that is placed in series with the crystal between the OSC1 pin and the crystal pin. Rs dissipates some of the power so the Max DI crystal parameter is not exceeded. Only use the recommended $C_1$, $C_2$, and Rs values with the associated crystal part. The values in the table were used in the simulation to ensure crystal startup and to determine the worst case drive level (WC DI). The value in the WC DI column should not be greater than the Max DI Crystal parameter. The WC DI value can be used to determine if a crystal with similar parameter values but a lower Max DI value is acceptable.

**Table 32-24. Crystal Parameters**

| MFG | MFG Part# | Holder | PKG Size (mm x mm) | Freq (MHz) | Crystal Spec (Tolerance / Stability) | Crystal Parameters | | | | | | Recommended Component Values | | | WC DI (µW) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Typical Values | | | Max Values | | | | | | |
| | | | | | | C0 (pF) | C1 (fF) | L1 (mH) | ESR (Ω) | Max DI (µW) | $C_L$ (pf) | $C_1$ (pF) | $C_2$ (pF) | Rs (kΩ) | |
| NDK | NX8045GB-4.000M-STD-CJL-5 | NX8045GB | 8 x 4.5 | 4 | 30/50 ppm | 1.00 | 2.70 | 598.10 | 300 | 500 | 8 | 12 | 12 | 0 | 132 |
| FOX | FQ1045A-4 | 2-SMD | 10 x 4.5 | 4 | 30/30 ppm | 1.18 | 4.05 | 396.00 | 150 | 500 | 10 | 14 | 14 | 0 | 103 |
| NDK | NX8045GB-5.000M-STD-CSF-4 | NX8045GB | 8 x 4.5 | 5 | 30/50 ppm | 1.00 | 2.80 | 356.50 | 250 | 500 | 8 | 12 | 12 | 0 | 164 |
| NDK | NX8045GB-6.000M-STD-CSF-4 | NX8045GB | 8 x 4.5 | 6 | 30/50 ppm | 1.30 | 4.10 | 173.20 | 250 | 500 | 8 | 12 | 12 | 0 | 214 |
| FOX | FQ1045A-6 | 2-SMD | 10 x 4.5 | 6 | 30/30 ppm | 1.37 | 6.26 | 112.30 | 150 | 500 | 10 | 14 | 14 | 0 | 209 |
| NDK | NX8045GB-8.000M-STD-CSF-6 | NX8045GB | 8 x 4.5 | 8 | 30/50 ppm | 1.00 | 2.80 | 139.30 | 200 | 500 | 8 | 12 | 12 | 0 | 277 |
| FOX | FQ7050B-8 | 4-SMD | 7 x 5 | 8 | 30/30 ppm | 1.95 | 6.69 | 59.10 | 80 | 500 | 10 | 14 | 14 | 0 | 217 |
| ECS | ECS-80-16-28A-TR | HC49/US | 12.5 x 4.85 | 8 | 50/30 ppm | 1.82 | 4.90 | 85.70 | 80 | 500 | 16 | 24 | 24 | 0 | 298 |
| Abracon | AABMM-12.0000MHz-10-D-1-X-T | ABMM | 7.2 x 5.2 | 12 | 10/20 ppm | 2.37 | 8.85 | 20.5 | 50 | 500 | 10 | 12 | 12 | 2.0[a] | 124 |
| NDK | NX3225GA-12.000MHZ-STD-CRG-2 | NX3225GA | 3.2 x 2.5 | 12 | 20/30 ppm | 0.70 | 2.20 | 81.00 | 100 | 200 | 8 | 12 | 12 | 2.5 | 147 |
| NDK | NX5032GA-12.000MHZ-LN-CD-1 | NX5032GA | 5 x 3.2 | 12 | 30/50 ppm | 0.93 | 3.12 | 56.40 | 120 | 500 | 8 | 12 | 12 | 0 | 362 |

**Table 32-24. Crystal Parameters *(continued)***

| MFG | MFG Part# | Holder | PKG Size (mm x mm) | Freq (MHz) | Crystal Spec (Tolerance / Stability) | Crystal Parameters | | | | | | Recommended Component Values | | | WC DI (µW) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | Typical Values | | | Max Values | | | | | | |
| | | | | | | C0 (pF) | C1 (fF) | L1 (mH) | ESR (Ω) | Max DI (µW) | $C_L$ (pf) | $C_1$ (pF) | $C_2$ (pF) | Rs (kΩ) | |
| FOX | FQ5032B-12 | 4-SMD | 5 x 3.2 | 12 | 30/30 ppm | 1.16 | 4.16 | 42.30 | 80 | 500 | 10 | 14 | 14 | 0 | 370 |
| Abracon | AABMM-16.0000MHz-10-D-1-X-T | ABMM | 7.2 x 5.2 | 16 | 10/20 ppm | 3.00 | 11.00 | 9.30 | 50 | 500 | 10 | 12 | 12 | 2.0[a] | 143 |
| Ecliptek | ECX-6595-16.000M | HC-49/UP | 13.3 x 4.85 | 16 | 15/30 ppm | 3.00 | 12.7 | 8.1 | 50 | 1000 | 10 | 12 | 12 | 2.0[a] | 139 |
| NDK | NX3225GA-16.000MHZ-STD-CRG-2 | NX3225GA | 3.2 x 2.5 | 16 | 20/30 ppm | 1.00 | 2.90 | 33.90 | 80 | 200 | 8 | 12 | 12 | 2 | 188 |
| NDK | NX5032GA-16.000MHZ-LN-CD-1 | NX5032GA | 5 x 3.2 | 16 | 30/50ppm | 1.02 | 3.82 | 25.90 | 120[b] | 500 | 8 | 10 | 10 | 0 | 437 |
| ECS | ECS-160-9-42-CKM-TR | ECX-42 | 4 x 2.5 | 16 | 10/10 ppm | 1.47 | 3.90 | 25.84 | 60 | 300 | 9 | 12 | 12 | 0.5 | 289 |
| Abracon | AABMM-25.0000MHz-10-D-1-X-T | ABMM | 7.2 x 5.2 | 25 | 10/20 ppm | 3.00 | 11.00 | 3.70 | 50 | 500 | 10 | 12 | 12 | 2.0[a] | 158 |
| Ecliptek | ECX-6593-25.000M | HC-49/UP | 13.3 x 4.85 | 25 | 15/30 ppm | 3.00 | 12.8 | 3.2 | 40 | 1000 | 10 | 12 | 12 | 1.5[a] | 159 |
| NDK | NX3225GA-25.000MHZ-STD-CRG-2 | NX3225GA | 3.2 x 2.5 | 25 | 20/30 ppm | 1.10 | 4.70 | 8.70 | 50 | 200 | 8 | 12 | 12 | 2 | 181 |
| NDK | NX5032GA-25.000MHZ-LD-CD-1 | NX5032GA | 5 x 3.2 | 25 | 30/50 ppm | 1.3 | 5.1 | 7.1 | 70 | 500 | 8 | 10 | 10 | 1.0[a] | 216 |
| | | | | | | | | | | | | 12 | 12 | 0.75[c] | 269 |
| AURIS | Q-25.000M-HC3225/4-F-30-30-E-12-TR | HC3225/4 | 3.2 x 2.5 | 25 | 30/30 ppm | 1.58 | 5.01 | 8.34 | 50 | 500 | 12 | 16 | 16 | 1 | 331 |
| FOX | FQ5032B-25 | 4-SMD | 5 x 3.2 | 25 | 30/30 ppm | 1.69 | 7.92 | 5.13 | 50 | 500 | 10 | 14 | 14 | 0.5 | 433 |
| TXC | 7A2570018 | NX5032GA | 5 x 3.2 | 25 | 20/25 ppm | 2.0 | 6.7 | 6.1 | 30 | 350 | 10 | 12 | 12 | 2.0[c] | 124 |

a. $R_S$ values as low as 0 Ohms can be used. Using a lower $R_S$ value will result in the WC DL to increase towards the Max DL of the crystal.

b. Although this ESR value is outside of the recommended crystal ESR maximum for this frequency, this crystal has been simulated to confirm proper operation and is valid for use with this device.

c. $R_S$ values as low as 500 Ohms can be used. Using a lower $R_S$ value will result in the WC DL to increase towards the Max DL of the crystal.

### 32.9.6 System Clock Specification with ADC Operation

**Table 32-25. System Clock Characteristics with ADC Operation**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{sysadc}$ | System clock frequency when the ADC module is operating (when PLL is bypassed). | - | 16 | - | MHz |

### 32.9.7 System Clock Specification with USB Operation

**Table 32-26. System Clock Characteristics with USB Operation**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{sysusb}$ | System clock frequency when the USB module is operating (note that MOSC must be the clock source, either with or without using the PLL) | 30 | - | - | MHz |

## 32.10 Sleep Modes

The following tables can be used to calculate the maximum wake time from Sleep or Deep Sleep depending on the specific application. Depending on the application configuration, each of the parameters, except for $T_{FLASH}$, add sequential latency to the wake time. Flash restoration happens in parallel to the other wake processes and its wake time is normally absorbed by the other latencies. As an example, the wake time for a device in Deep Sleep, with the PIOSC and PLL turned off and the Flash and SRAM in low power mode is calculated as follows:

Wake Time = $T_{PIOSCDS} + T_{PLLDS} + T_{SRAMLPDS}$

$T_{FLASH}$ does not contribute to this equation since all other parameters are greater in value.

Note that in Sleep mode the wake time due to a clock source is zero because the device uses the same clock configuration in Run mode; thus, there is no latency involved with respect to the clocks.

**Table 32-27. Wake from Sleep Characteristics**

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| D1 | $T_{PIOSC}$ | Time to restore PIOSC as System Clock in Sleep mode | - | - | N/A[a] | µs |
| D2 | $T_{MOSC}$ | Time to restore MOSC as System Clock in Sleep mode | - | - | N/A[b] | µs |
| D3 | $T_{PLL}$ | Time to restore PLL as System Clock in Sleep mode | - | - | N/A[c] | µs |
| D4 | $T_{LDO}$ | Time to restore LDO to 1.2 V in Sleep mode | - | - | 39 | µs |
| D5 | $T_{FLASH}$ | Time to restore Flash to active state from low power state in Sleep mode | - | - | 5 | µs |
| D6 | $T_{SRAMLP}$ | Time to restore SRAM to active state from low power state in Sleep mode | - | - | 15 | µs |
| D7 | $T_{SRAMSTBY}$ | Time to restore SRAM to active state from standby state in Sleep mode | - | - | 15 | µs |

a. Because the PIOSC is enabled in both Run and Sleep Mode for this configuration, no restoration time is required.

b. Because the MOSC is enabled in both Run and Sleep Mode for this configuration, no restoration time is required.

c. Because the PLL is enabled in both Run and Sleep Mode for this configuration, no restoration time is required.

**Table 32-28. Wake from Deep Sleep Characteristics**

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| D8 | $T_{PIOSCDS}$ | Time to restore PIOSC as System Clock in Deep Sleep Mode | - | - | 14 | Deep Sleep Clock Cycles[a] |
| D9 | $T_{MOSCDS}$ | Time to restore MOSC as System Clock in Deep Sleep Mode | - | - | 18 | ms |
| D10 | $T_{PLLDS}$ | Time to restore PLL as System Clock in Deep Sleep Mode | - | - | 1 cycle of Deep Sleep Clock + 512 cycles of PLL reference Clock[a] | clocks |
| D11 | $T_{LDODS}$ | Time to restore LDO to 1.2 V in Deep Sleep Mode | - | - | 39 | µs |
| D12 | $T_{FLASHLPDS}$ | Time to restore Flash to active state from low power state | - | - | 5 | µs |
| D13 | $T_{SRAMLPDS}$ | Time to restore SRAM to active state from low power state | - | - | 15 | µs |

**Table 32-28. Wake from Deep Sleep Characteristics *(continued)***

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| D14 | $T_{SRAMSTBYDS}$ | Time to restore SRAM to active state from standby state | - | - | 15 | µs |

a. Deep Sleep Clock can vary. See page 293 for the Deep Sleep Clock options.

## 32.11 Hibernation Module

The Hibernation module requires special system implementation considerations because it is intended to power down all other sections of its host device, refer to “Hibernation Module” on page 563.

### Table 32-29. Hibernation Module Battery Characteristics

| Parameter | Parameter Name | Min | Nominal | Max | Unit |
|---|---|---|---|---|---|
| $V_{BAT}$ | Battery supply voltage | 1.8 | 3.0 | 3.6[a] | V |
| $V_{BATRMP}$[b] | $V_{BAT}$ battery supply voltage ramp time | 0 | - | 0.7 | V/µs |
| $V_{LOWBAT}$ | Low battery detect voltage, VBATSEL=0x0 | 1.8 | 1.9 | 2.0 | V |
| | Low battery detect voltage, VBATSEL=0x1 | 2.0 | 2.1 | 2.2 | V |
| | Low battery detect voltage, VBATSEL=0x2 | 2.2 | 2.3 | 2.4 | V |
| | Low battery detect voltage, VBATSEL=0x3 | 2.4 | 2.5 | 2.6 | V |

a. To ensure proper functionality, any voltage input higher than $V_{BAT}$ Max, must be connected through a diode.

b. For recommended $V_{BAT}$ RC circuit values, refer to the diagrams located in“Hibernation Clock Source” on page 567.

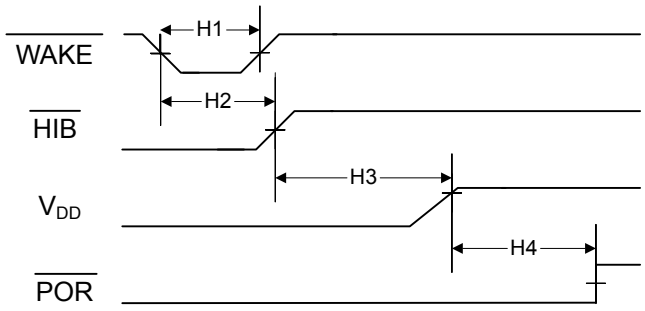
### Table 32-30. Hibernation Module Characteristics

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| H1 | $T_{WAKE}$ | WAKE assertion time | 100 | - | - | ns |
| H2 | $T_{WAKE_TO_HIB}$ | WAKE assert to HIB desassert (wake up time) | - | - | 1 | Hibernation clock period |
| H3 | $T_{VDD_RAMP}$ | $V_{DD}$ ramp to 3.0 V | - | Depends on characteristics of power supply | - | µs |
| H4 | $T_{VDD_CODE}$ | $V_{DD}$ at 3.0 V to internal POR deassert; first instruction executes | - | - | 500 | µs |
| H5 | $DC_{RTCCLK}$ | Duty cycle for RTCCLK output signal, when using a 32.768-kHz crystal | 40 | - | 60 | % |
| | | Duty cycle for RTCCLK output signal, when using a 32.768-kHz external single-ended (bypass) clock source | 30 | - | 70 | % |

### Table 32-31. Hibernation Module Tamper I/O Characteristics

| Parameter | Parameter Name | Min | Nominal | Max | Unit |
|---|---|---|---|---|---|
| $R_{TPU}$ | TMPRn pull-up resistor | 3.5<br>- | 4.4 | 5.2<br>- | MΩ |
| $T_{SP}$ | TMPRn pulse width with short glitch filter | 62 | - | - | µs |
| $T_{LP}$ | TMPRn pulse width with long glitch filter | 94 | - | - | ms |
| $T_{NMIS}$ | TMPRn assertion to NMI (short glitch filter) | - | - | 95 | µs |
| $T_{NMIL}$ | TMPRn assertion to NMI (long glitch filter) | - | - | 94 | ms |
| $V_{IH}$ | TMPRn high-level input voltage when operating from VBAT | $V_{BAT}$*0.8 | - | - | V |

**Figure 32-15. Hibernation Module Timing**



## 32.12 Flash Memory

**Table 32-32. Flash Memory Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $PE_{CYC}$ | Number of program/erase cycles[a] | 100,000 | - | - | cycles |
| $T_{RET}$ | Data retention with 100% power-on hours at $T_J$=85˚C | 20 | - | - | years |
| $T_{RET_EXTEMP}$ | Data retention with 10% power-on hours at $T_J$=125˚C and 90% power-on hours at $T_J$=100˚C | 11 | - | - | years |
| $T_{PROG64}$ | Program time for double-word-aligned (64 bits) data[b] | 30 | 100 | 300 | µs |
| $T_{ERASE}$ | Page erase time, <1k cycles | - | 8 | 15 | ms |
| | Page erase time, 10k cycles | - | 15 | 40 | ms |
| | Page erase time, 100k cycles | - | 75 | 500 | ms |
| $T_{ME}$ | Mass erase time, <1k cycles | - | 10 | 25 | ms |
| | Mass erase time, 10k cycles | - | 20 | 70 | ms |
| | Mass erase time, 100k cycles | - | 300 | 2500 | ms |

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

b. If programming fewer than 64 bits of data, the programming time is the same. For example, if only 32 bits of data need to be programmed, the other 32 bits are masked off.

## 32.13 EEPROM

**Table 32-33. EEPROM Characteristics**[a]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $EPE_{CYC}$[b] | Number of mass program/erase cycles of a single word[c] | 500,000 | - | - | cycles |
| $ET_{RET}$ | Data retention with 100% power-on hours at $T_J$=85˚C | 20 | - | - | years |
| $ET_{RET_EXTEMP}$ | Data retention with 10% power-on hours at $T_J$=125˚C and 90% power-on hours at $T_J$=100˚C | 11 | - | - | years |
| $ET_{PROG}$ | Program time for 32 bits of data with memory space available | - | 110 | 600 | µs |
| | Program time for 32 bits of data in which a copy to the copy buffer is required, the copy buffer has space and less than 10% of EEPROM endurance used | - | 30 | - | ms |
| | Program time for 32 bits of data in which a copy to the copy buffer is required, the copy buffer has space and greater than 90% of EEPROM endurance used | - | - | 900 | ms |
| | Program time for 32 bits of data in which a copy of the copy buffer is required, the copy buffer requires an erase and less than 10% of EEPROM endurance used | - | 60 | - | ms |
| | Program time for 32 bits of data a copy to the copy buffer is required, the copy buffer requires an erase and greater than 90% of EEPROM endurance used | - | - | 1800 | ms |
| $ET_{READ}$ | Read access time[d] | - | 7+ 2*(EWS) | 9+4*(EWS) | system clocks |
| $ET_{ME}$ | Mass erase time, <1k cycles | - | 8 | 15 | ms |
| | Mass erase time, 10k cycles | - | 15 | 40 | ms |
| | Mass erase time, 100k cycles | - | 75 | 500 | ms |

a. Because the EEPROM operates as a background task and does not prevent the CPU from executing from Flash memory, the operation will complete within the maximum time specified provided the EEPROM operation is not stalled by a Flash memory program or erase operation.

b. One word can be written more than 500K times, but these writes impact the endurance of the words in the meta-block that the word is within. Different words can be written such that any or all words can be written more than 500K times when write counts per word stay about the same. See the section called “Endurance” on page 651 for more information.

c. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

d. The EWS field is programmed in the **MEMTIM0** register at Sysctl Offset 0x0C0.

## 32.14 Input/Output Pin Characteristics

**Note:** All GPIO signals are 3.3-V tolerant, except for PB1 (USB0VBUS) which is 5-V tolerant. See "Signal Description" on page 773 for more information on GPIO configuration.

Two types of pads are provided on the device:

- Fast GPIO pads: These pads provide variable, programmable drive strength and optimized voltage output levels.
- Slow GPIO pads: These pads provide 2-mA drive strength and are designed to be sensitive to voltage inputs. The following GPIOs port pins are designed with Slow GPIO Pads:
  – PJ1

**Note:** Port pins PL6 and PL7 operate as Fast GPIO pads, but have 4-mA drive capability only. GPIO register controls for drive strength, slew rate and open drain have no effect on these pins. The registers which have no effect are as follows: **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR12R**, **GPIOSLR**, and **GPIOODR**.

**Note:** Port pins PM[7:4] operate as Fast GPIO pads but support only 2-, 4-, 6-, and 8-mA drive capability. 10- and 12-mA drive are not supported. All standard GPIO register controls, except for the **GPIODR12R** register, apply to these port pins.

**Table 32-34. Fast GPIO Module Characteristics**[abcd]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $C_{LGPIO}$ | Capacitive loading for measurements given in this table[e] | - | - | 50 | pF |
| $R_{GPIOPU}$ | Fast GPIO internal pull-up resistor[f] | 12.1 | 16.0 | 20.2 | kΩ |
| $R_{GPIOPU4MA}$ | Fast GPIO PL6 and PL7 (4-mA only) pull-up resistor | 25 | - | 40 | kΩ |
| $R_{GPIOPD}$ | Fast GPIO internal pull-down resistor[f] | 13.0 | 20.5 | 35.5 | kΩ |
| $R_{GPIOPD4MA}$ | Fast GPIO PL6 and PL7 (4-mA only) pull-down resistor | 10 | 14.3 | 17 | kΩ |
| $I_{LKG+}$ | Fast GPIO input leakage current, 0 V ≤ $V_{IN}$ ≤ $V_{DD}$GPIO pins[g] | - | - | 400 | nA |
| | Fast GPIO input leakage current, 0 V < $V_{IN}$ ≤ $V_{DD}$, Fast GPIO pins configured as ADC or analog comparator inputs | - | - | 400 | nA |
| $I_{INJ-}$ | DC injection current, $V_{IN}$ ≤ 0 V | - | - | 60 | µA |
| $I_{MAXINJ-}$ | Max negative injection if not voltage protected[d] | - | - | -0.5 | mA |
| $T_{GPIOR}$ | Fast GPIO rise time, 2-mA drive[h] | - | 7.85 | 11.73 | ns |
| | Fast GPIO rise time, 4-mA drive[h] | | 4.15 | 6.35 | ns |
| | Fast GPIO rise time, 8-mA drive[h] | | 2.33 | 3.73 | ns |
| | Fast GPIO rise time, 8-mA drive with slew rate control[h] | | 3.77 | 5.76 | ns |
| | Fast GPIO rise time, 10-mA drive[h] | | 1.98 | 3.22 | ns |
| | Fast GPIO rise time, 12-mA drive[h] | | 1.75 | 2.9 | ns |

### Table 32-34. Fast GPIO Module Characteristics *(continued)*

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $T_{GPIOF}$ | Fast GPIO fall time, 2-mA drive[i] | - | 10.3 | 16.5 | ns |
| | Fast GPIO fall time, 4-mA drive[i] | | 5.15 | 8.29 | ns |
| | Fast GPIO fall time, 8-mA drive[i] | | 2.58 | 4.16 | ns |
| | Fast GPIO fall time, 8-mA drive with slew rate control[i] | | 3.54 | 5.55 | ns |
| | Fast GPIO fall time, 10-mA drive[i] | | 2.07 | 3.34 | ns |
| | Fast GPIO fall time, 12-mA drive[i] | | 1.73 | 2.78 | ns |

a. $V_{DD}$ must be within the range specified in Table 32-6 on page 2097.

b. Leakage and Injection current characteristics specified in this table also apply to XOSC0 and XOSC1 inputs.

c. Note that for the ADC's external reference inputs, care must be taken to avoid a current limiting resistor (refer to $I_{VREF}$ spec in Table 32-44 on page 2138)

d. I/O pads should be protected if at any point the IO voltage has a possibility of going outside the limits shown in the table. If the part is unpowered, the IO pad Voltage or Current must be limited (as shown in this table) to avoid powering the part through the IO pad, causing potential irreversible damage.

e. Refer to individual peripheral sections for specific loading information.

f. This value includes all GPIO except for port pins PL6 and PL7.

g. The leakage current is measured with $V_{IN}$ applied to the corresponding pin(s). The leakage of digital port pins is measured individually. The port pin is configured as an input and the pull-up/pull-down resistor is disabled.

h. Time measured from 20% to 80% of $V_{DD}$.

i. Time measured from 80% to 20% of $V_{DD}$.

### Table 32-35. Slow GPIO Module Characteristics[abc]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $C_{LGPIO}$ | Capacitive loading for measurements given in this table[d] | - | - | 50 | pF |
| $R_{GPIOPU}$ | Slow GPIO internal pull-up resistor | 13.8 | 20.0 | 31.4 | kΩ |
| $R_{GPIOPD}$ | Slow GPIO internal pull-down resistor | 13.0 | 20.5 | 35.5 | kΩ |
| $I_{LKG+}$ | Slow GPIO input leakage current, $0 V \le V_{IN} \le V_{DD}$ GPIO pins[e] | - | - | 3.25 | nA |
| | Slow GPIO input leakage current, $0 V < V_{IN} \le V_{DD}$, GPIO pins configured as ADC or analog comparator inputs | - | - | 3.25 | nA |
| $I_{INJ-}$ | DC injection current, $V_{IN} \le 0 V$ | - | - | 3.42 | µA |
| $T_{GPIOR}$ | Slow GPIO rise time, 2-mA drive[f] | - | 19.3 | 29.8 | ns |
| $T_{GPIOF}$ | Slow GPIO fall time, 2-mA drive[g] | - | 12.8 | 21.1 | ns |

a. $V_{DD}$ must be within the range specified in Table 32-6 on page 2097.

b. $V_{IN}$ must be within the range specified in Table 32-1 on page 2095. Leakage current outside of this maximum voltage is not guaranteed and can result in permanent damage of the device.

c. To avoid potential damage to the part, either the voltage or current on the non-Power, non-$\overline{WAKE}$ input/outputs should be limited externally as shown in this table.

d. Refer to individual peripheral sections for specific loading information.

e. The leakage current is measured with $V_{IN}$ applied to the corresponding pin(s). The leakage of digital port pins is measured individually. The port pin is configured as an input and the pull-up/pull-down resistor is disabled.

f. Time measured from 20% to 80% of $V_{DD}$.

g. Time measured from 80% to 20% of $V_{DD}$.

### 32.14.1 Types of I/O Pins and ESD Protection

**Caution – All device I/Os pins, except for PB1, are NOT 5V tolerant; voltages in excess of the limits shown in Table 32-6 on page 2097 can permanently damage the device. PB1 is used for the USB's USB0VBUS signal, which requires a 5-V input.**

#### 32.14.1.1 Hibernate $\overline{\text{WAKE}}$ pin

The Hibernate $\overline{\text{WAKE}}$ pin uses ESD protection, similar to the one shown in Figure 32-16 on page 2128. This ESD protection prevents a direct path between this pad and any power supply rails in the device. The $\overline{\text{WAKE}}$ pad input voltage should be kept inside the maximum ratings specified in Table 32-1 on page 2095 to ensure current leakage and current injections are within acceptable range. Current leakages and current injection for these pins are specified in Table 32-36 on page 2128.

**Figure 32-16. ESD Protection**

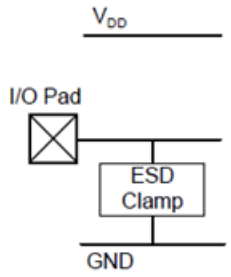



**Table 32-36. Pad Voltage/Current Characteristics for Hibernate $\overline{\text{WAKE}}$ Pin[ab]**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $I_{LKG+}$ | Positive IO leakage for $V_{DD} \le V_{IN} \le V_{BAT}$ + 0.3V | - | - | 300 | nA |
| $I_{LKG-}$ | Negative IO leakage for-0.3V ≤ $V_{IN}$ ≤0V[c] | - | - | 43.3 | µA |
| $I_{INJ+}$ | Max positive injection if not voltage protected[d] | - | - | 2 | mA |
| $I_{INJ-}$ | Max negative injection if not voltage protected[d] | - | - | -0.5 | mA |

a. $V_{IN}$ must be within the range specified in Table 32-1 on page 2095. Leakage current outside of this maximum voltage is not guaranteed and can result in permanent damage of the device.

b. VDD must be within the range specified in Table 32-6 on page 2097.

c. Leakage outside the minimum range (-0.3V) is unbounded and must be limited to $I_{INJ-}$ using an external resistor.

d. If the I/O pad is not voltage limited, it should be current limited (to $I_{INJ}$+ and $I_{INJ-}$) if there is any possibility of the pad voltage exceeding the $V_{IO}$ limits (including transient behavior during supply ramp up, or at any time when the part is unpowered).

#### 32.14.1.2 Non-Power I/O Pins

Most non-power I/Os (with the exception of the I/O pad for Hibernate $\overline{\text{WAKE}}$ input) have ESD protection as shown in Figure 32-17 on page 2129.

These I/Os have an ESD clamp to ground and a diode connection to the corresponding power supply rail. The voltage and current of these I/Os should follow the specifications in Table 32-37 on page 2129 to prevent potential damage to the device. In addition, it is recommended that the ADC external reference specifications in Table 32-44 on page 2138 be adhered to prevent any gain error.

**Figure 32-17. ESD Protection for Non-Power Pins (Except $\overline{\text{WAKE}}$ Signal)**

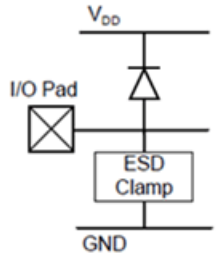



**Table 32-37. Non-Power I/O Pad Voltage/Current Characteristics** [abc]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{IO}$[d] | IO pad voltage limits if voltage protected | -0.3 | $V_{DD}$ | $V_{DD}$+0.3 | V |
| $I_{LKG+}$ | Positive IO leakage for $V_{DD} \le V_{IN} \le V_{IO}$ MAX[e] | - | - | 400 | nA |
| $I_{LKG-}$ | Negative IO leakage for $V_{IO}$ MIN $\le V_{IN} \le$ 0V[e] | - | - | 60 | µA |
| $I_{INJ+}$ | Max positive injection if not voltage protected[f] | - | - | 2 | mA |
| $I_{INJ-}$ | Max negative injection if not voltage protected[f] | - | - | -0.5 | mA |

a. To avoid potential damage to the part, either the voltage or current on the non-Power, non-$\overline{\text{WAKE}}$ input/outputs should be limited externally as shown in this table.

b. Note that for the ADC's external reference inputs, care must be taken to avoid a current limiting resistor (refer to IVREF spec in Table 32-44 on page 2138)

c. I/O pads should be protected if at any point the IO voltage has a possibility of going outside the limits shown in the table. If the part is unpowered, the IO pad Voltage or Current must be limited (as shown in this table) to avoid powering the part through the IO pad, causing potential irreversible damage.

d. The Hibernate XOSC pins are non-failsafe and should follow the limits for $V_{IO}$ with respect to both $V_{DD}$ and $V_{BAT}$. Thus $V_{IO}$ for the HIB XOSC pins should also fall within a MIN of -0.3 and a MAX of $V_{BAT}$ + 0.3.

e. MIN and MAX leakage current for the case when the I/O is voltage protected to $V_{IO}$ Min or $V_{IO}$ Max.

f. If the I/O pad is not voltage limited, it should be current limited (to $I_{INJ}$+ and $I_{INJ-}$) if there is any possibility of the pad voltage exceeding the $V_{IO}$ limits (including transient behavior during supply ramp up, or at any time when the part is unpowered).

## 32.15 External Peripheral Interface (EPI)

**Table 32-38. EPI Interface Load Conditions**

| Signals | Load Value ($C_L$) |
|---|---|
| EPI0S[35:0] SDRAM interface | 30 pF |
| EPI0S[35:0] General-Purpose interface | |
| EPI0S[35:0] Host-Bus interface | |
| EPI0S[35:0] PSRAM interface | 40 pF |

When the EPI module is in SDRAM mode, EPI CLK must be configured to 12 mA. The EPI data bus can be configured to 8 mA. Table 32-39 on page 2130 shows the rise and fall times in SDRAM mode. When the EPI module is in Host-Bus or General-Purpose mode, the values in “Input/Output Pin Characteristics” on page 2126 should be used.

**Table 32-39. EPI SDRAM Characteristics**

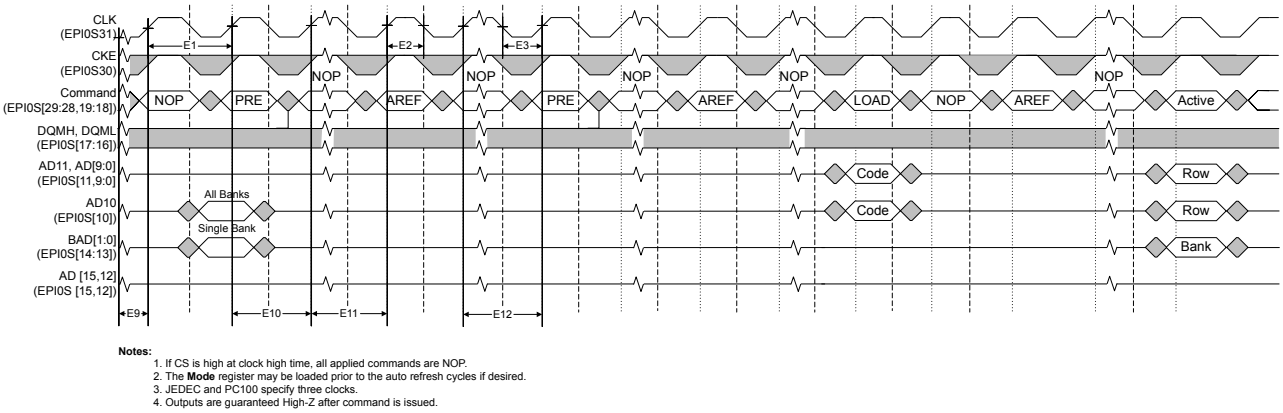
| Parameter | Parameter Name | Condition | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| $T_{SDRAMR}$ | EPI Rise Time (from 20% to 80% of $V_{DD}$) | 12-mA drive, $C_L$ = 30 pF | - | 2 | 3 | ns |
| $T_{SDRAMF}$ | EPI Fall Time (from 80% to 20% of $V_{DD}$) | 12-mA drive, $C_L$ = 30 pF | - | 2 | 3 | ns |

**Table 32-40. EPI SDRAM Interface Characteristics**[a]

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| E1 | $T_{CK}$ | SDRAM Clock period | 16.67 | - | - | ns |
| E2 | $T_{CH}$ | SDRAM Clock high time | 8.33 | - | - | ns |
| E3 | $T_{CL}$ | SDRAM Clock low time | 8.33 | - | - | ns |
| E4 | $T_{COV}$ | CLK to output valid | - | - | 4 | ns |
| E5 | $T_{COI}$ | CLK to output invalid | - | - | 4 | ns |
| E6 | $T_{COT}$ | CLK to output tristate | - | - | 4 | ns |
| E7 | $T_S$ | Input set up to CLK | 8.5 | - | - | ns |
| E8 | $T_H$ | CLK to input hold | 0 | - | - | ns |
| E9 | $T_{PU}$ | Power-up time | 100 | - | - | µs |
| E10 | $T_{RP}$ | Precharge all banks | 20 | - | - | ns |
| E11 | $T_{RFC}$ | Auto refresh | 66 | - | - | ns |
| E12 | $T_{MRD}$ | Program mode register | 2 | - | - | EPI CLK |

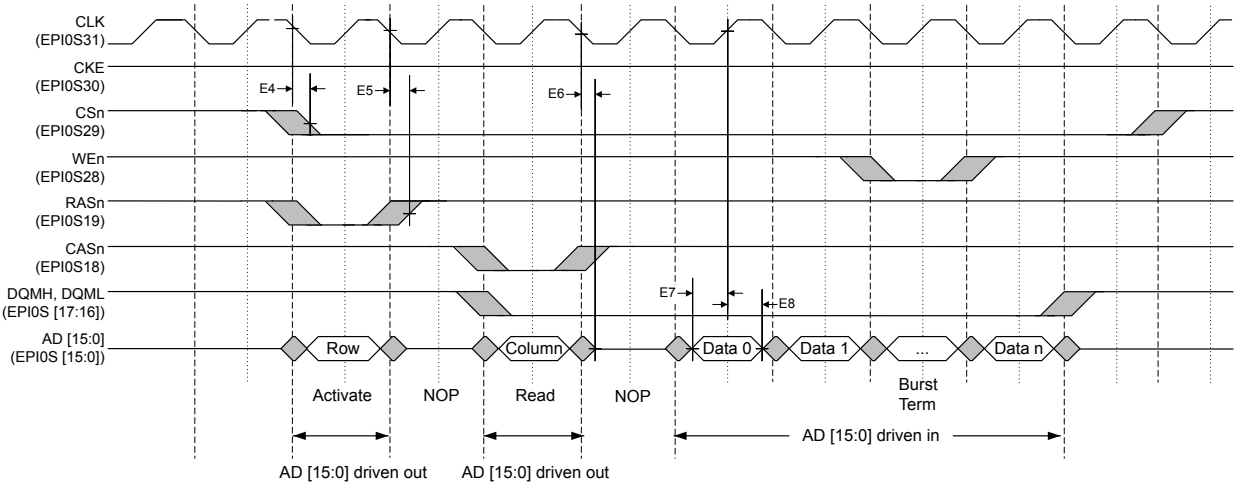
a. The EPI SDRAM interface must use 12-mA drive.

### Figure 32-18. SDRAM Initialization and Load Mode Register Timing

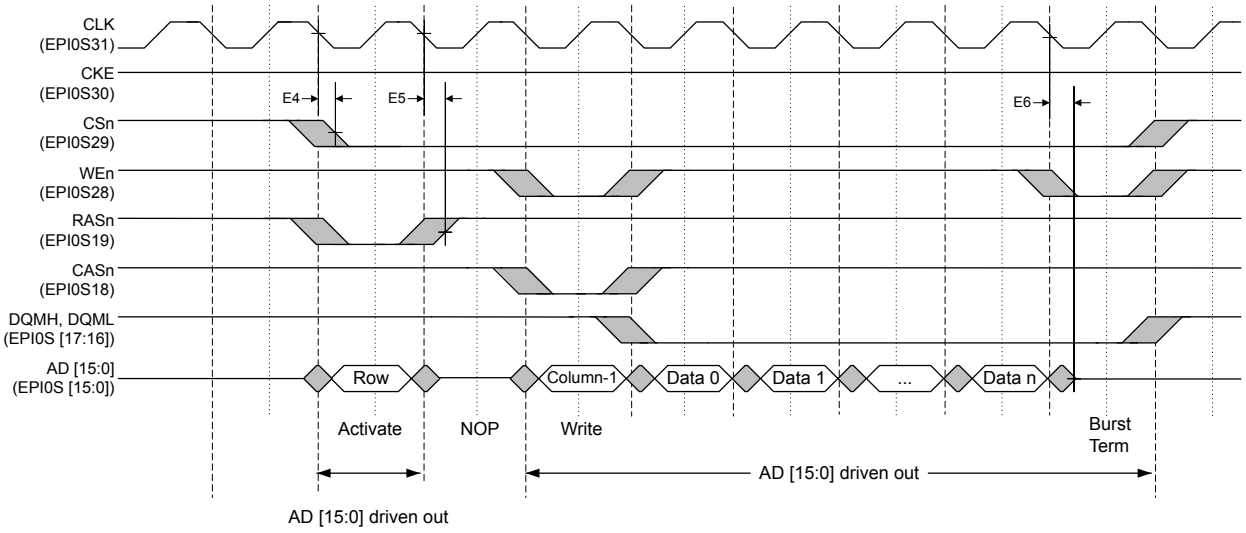



**Notes:**
1. If CS is high at clock high time, all applied commands are NOP.
2. The **Mode** register may be loaded prior to the auto refresh cycles if desired.
3. JEDEC and PC100 specify three clocks.
4. Outputs are guaranteed High-Z after command is issued.

### Figure 32-19. SDRAM Read Timing



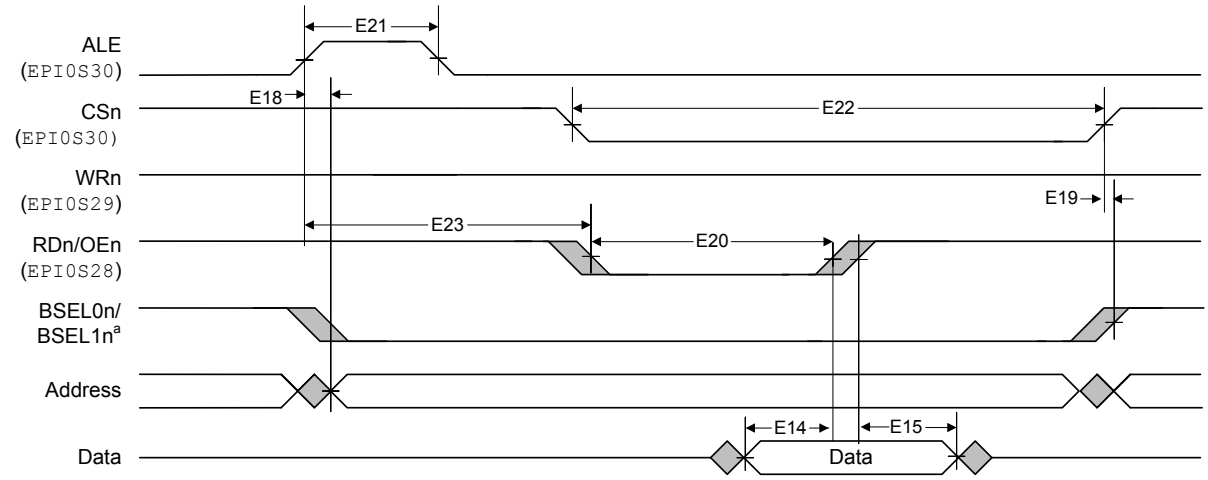
### Figure 32-20. SDRAM Write Timing




### Table 32-41. EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics

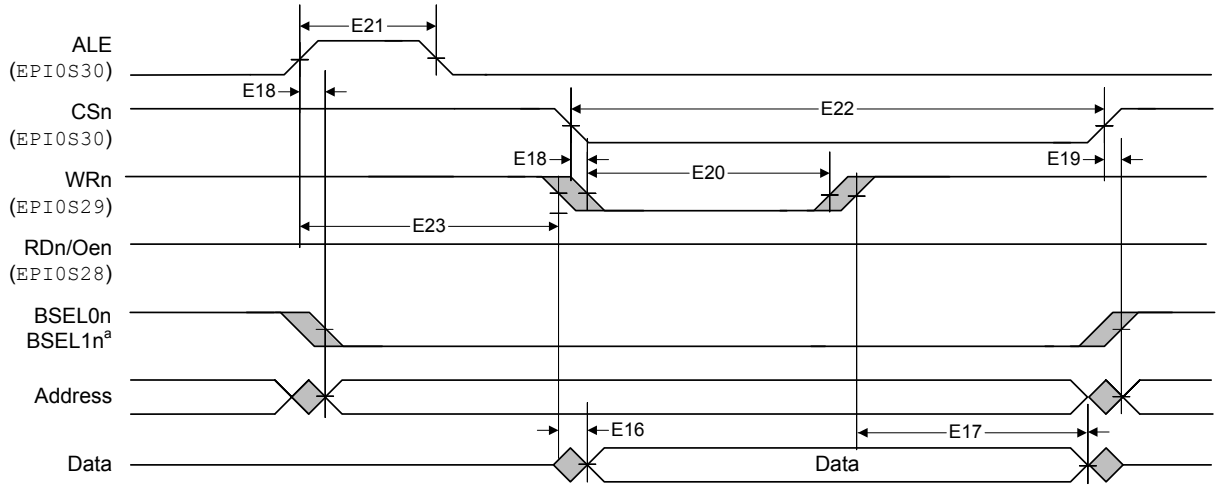
| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| E14 | $T_{ISU}$ | Read data set up time | 10 | - | - | ns |
| E15 | $T_{IH}$ | Read data hold time | 0 | - | - | ns |
| E16 | $T_{DV}$ | WRn to write data valid | - | - | 3.6 | ns |
| E17 | $T_{DI}$ | Data hold from WRn invalid | 1 | - | - | EPI Clocks |
| E18 | $T_{OV}$ | ALE/CSn to output valid | - | - | 4 | ns |
| E19 | $T_{OINV}$ | CSn to output invalid | - | - | 4 | ns |
| E20 | $T_{STLOW}$ | WRn / RDn strobe width low | 1 | - | - | EPI Clocks |
| E21 | $T_{ALEHIGH}$ | ALE width high | - | 1 | - | EPI Clocks |
| E22 | $T_{CSLOW}$ | CSn width low | 2 | - | - | EPI Clocks |
| E23 | $T_{ALEST}$ | ALE rising to WRn / RDn strobe falling | 2 | - | - | EPI Clocks |
| E24 | $T_{ALEADD}$ | ALE falling to Address tristate | 1 | - | - | EPI Clocks |

**Figure 32-21. Host-Bus 8/16 Asynchronous Mode Read Timing**



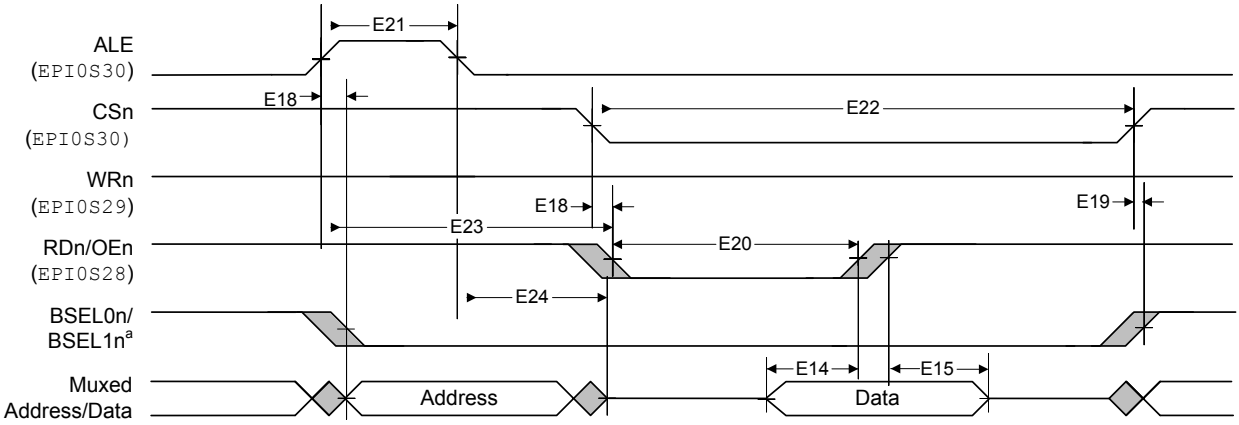

[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

**Figure 32-22. Host-Bus 8/16 Asynchronous Mode Write Timing**



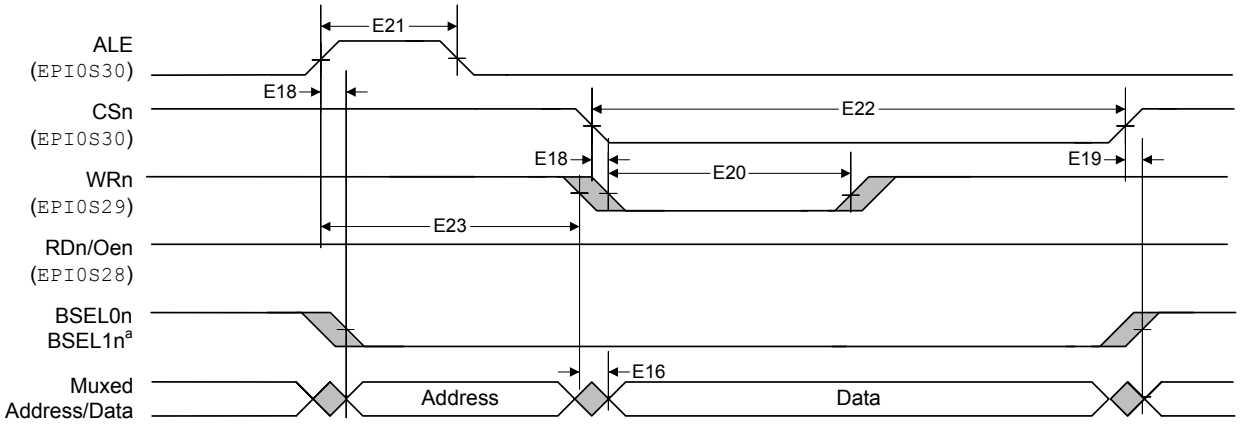

[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

### Figure 32-23. Host-Bus 8/16 Mode Asynchronous Muxed Read Timing




[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

### Figure 32-24. Host-Bus 8/16 Mode Asynchronous Muxed Write Timing

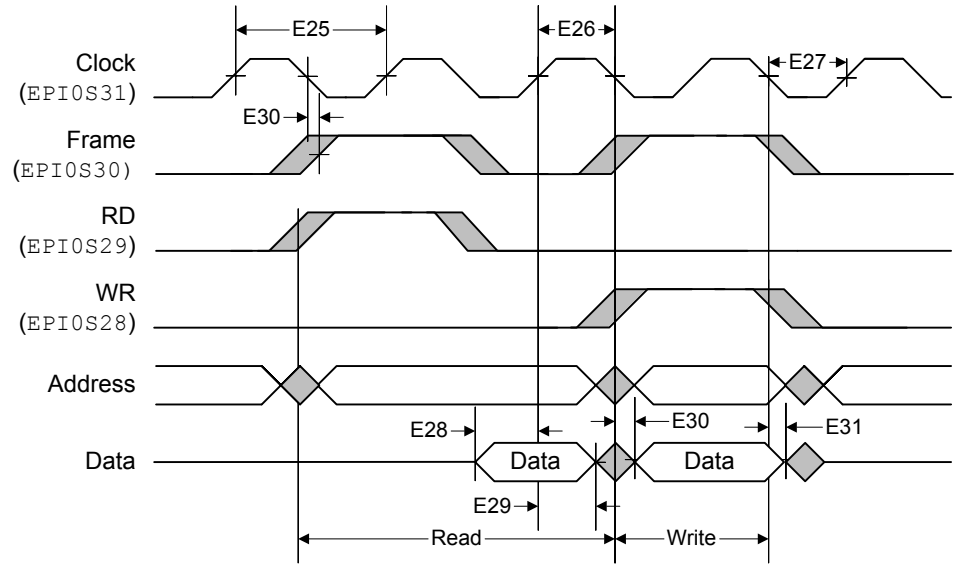



[a] BSEL0n and BSEL1n are available in Host-Bus 16 mode only.

### Table 32-42. EPI General-Purpose Interface Characteristics

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| E25 | $T_{CK}$ | General-Purpose Clock period | 16.67 | - | - | ns |
| E26 | $T_{CH}$ | General-Purpose Clock high time | 8.33 | - | - | ns |
| E27 | $T_{CL}$ | General-Purpose Clock low time | 8.33 | - | - | ns |
| E28 | $T_{ISU}$ | Input signal set up time to rising clock edge | 8.50 | - | - | ns |
| E29 | $T_{IH}$ | Input signal hold time from rising clock edge | 0 | - | - | ns |
| E30 | $T_{DV}$ | Falling clock edge to output valid | - | - | 4 | ns |
| E31 | $T_{DI}$ | Falling clock edge to output invalid | - | - | 4 | ns |
| E32 | $T_{RDYSU}$ | iRDY assertion or deassertion set up time to falling clock edge | 8.5 | - | - | ns |

**Figure 32-25. General-Purpose Mode Read and Write Timing**




**Note:** This figure illustrates accesses where the FRM50 bit is clear, the FRMCNT field is 0x0 and the WR2CYC bit is clear.

**Table 32-43. EPI PSRAM Interface Characteristics**

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| E33 | $T_{EPICLK}$ | EPI_CLK period | 20 | - | - | ns |
| E34 | $T_{RTFT}$ | EPI_CLK rise or fall time | - | - | 1.8 | ns |
| E35 | $T_{OV}$ | Falling EPI_CLK to Address/Write Data or Control output valid[a] | 4.5 | - | 20 | ns |
| E36 | $T_{HT}$ | Falling EPI_CLK to Address/Write Data or Control hold time[a] | 2 | - | - | ns |
| E37 | $T_{SUP}$ | Read data setup time from EPI_CLK rising | - | - | 9 | ns |
| E38 | $T_{DH}$ | Read data output hold from EPI_CLK rising | 0 | - | - | ns |
| E39 | $T_{IRV}$ | iRDY setup time | - | - | 9 | ns |
| E40 | $T_{IRH}$ | iRDY hold time | - | - | 9 | ns |

a. Control output includes WRn, RDn, OEn, BSELn, ALE, and CSn.

**Figure 32-26. PSRAM Single Burst Read**

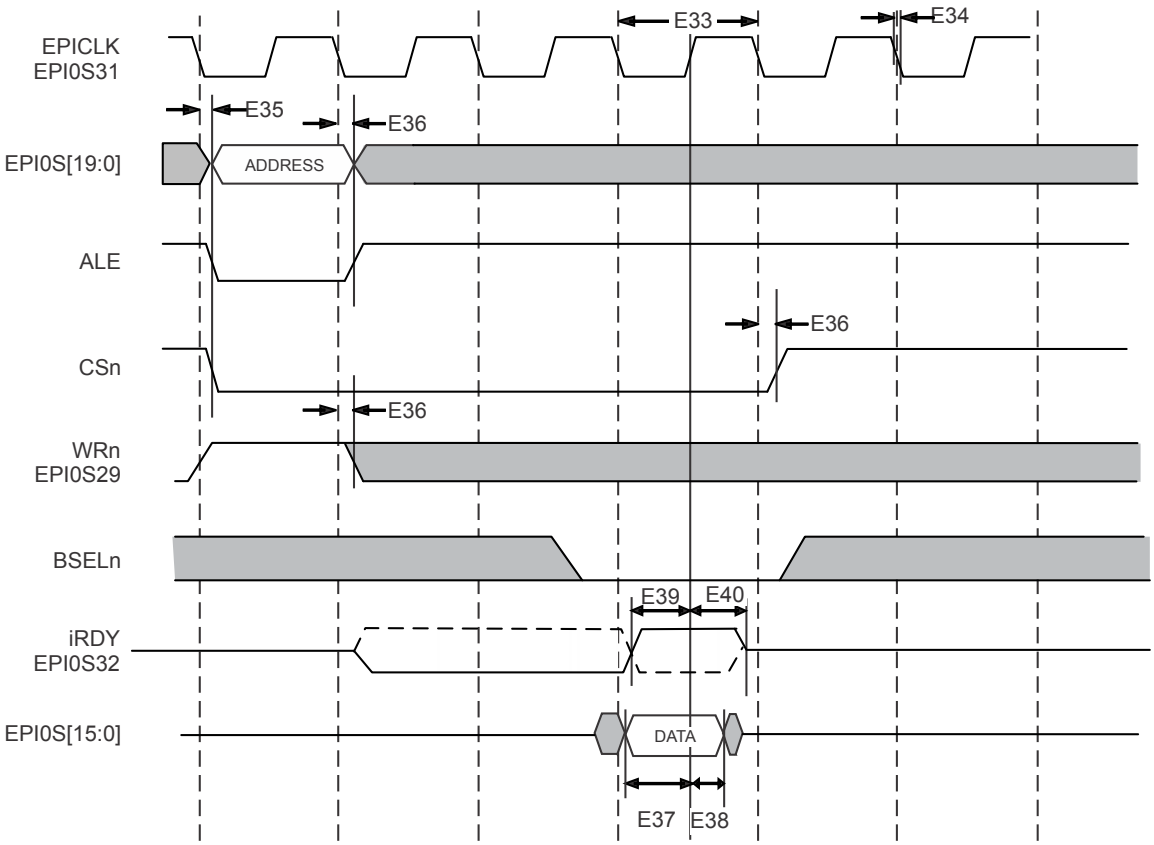
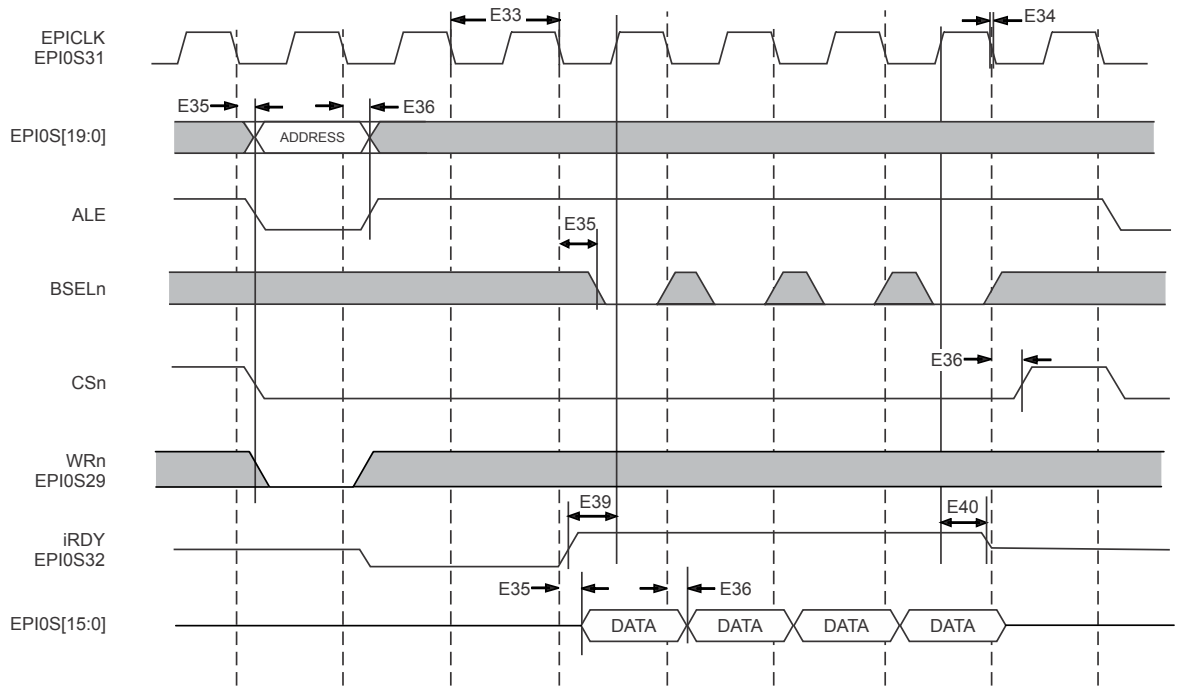


**Figure 32-27. PSRAM Single Burst Write**



## 32.16 Analog-to-Digital Converter (ADC)

**Table 32-44. ADC Electrical Characteristics for ADC at 1 Msps[ab]**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| POWER SUPPLY REQUIREMENTS | | | | | |
| $V_{DDA}$ | ADC supply voltage | 2.97 | 3.3 | 3.63 | V |
| GNDA | ADC ground voltage | - | 0 | - | V |
| VDDA / GNDA VOLTAGE REFERENCE | | | | | |
| $C_{REF}$ | Voltage reference decoupling capacitance | - | 1.0 // 0.01[c] | - | µF |
| EXTERNAL VOLTAGE REFERENCE INPUT | | | | | |
| $V_{REFA+}$ | Positive external voltage reference for ADC, when VREF field in the **ADCCTL** register is 0x1[d]- | 2.4 | $V_{DDA}$ | $V_{DDA}$ | V |
| $V_{REFA-}$ | Negative external voltage reference for ADC, when VREF field in the **ADCCTL** register is 0x1[d] | GNDA | GNDA | 0.3 | V |
| $I_{VREF}$ | Current on VREF+ input, using external $V_{REF+}$ = 3.3 V | - | 330.5 | 440 | µA |
| $I_{LVREF}$ | DC leakage current on VREF+ input when external VREF disabled | - | - | 2.0 | µA |
| $C_{REF}$ | External reference decoupling capacitance[d] | - | 1.0 // 0.01[c] | - | µF |
| ANALOG INPUT | | | | | |
| $V_{ADCIN}$ | Single-ended, full- scale analog input voltage, internal reference[ef] | 0 | - | $V_{DDA}$ | V |
| | Differential, full-scale analog input voltage, internal reference[eg] | $-V_{DDA}$ | - | $V_{VDDA}$ | V |
| | Single-ended, full-scale analog input voltage, external reference[df] | $V_{REFA-}$ | - | $V_{REFA+}$ | V |
| | Differential, full-scale analog input voltage, external reference[dh] | - ($V_{REFA+}$ - $V_{REFA-}$) | - | $V_{REFA+}$ - $V_{REFA-}$ | V |
| $VIN_{CM}$ | Input common mode voltage, differential mode[i] | - | - | [($V_{REFA+}$ + $V_{REFA-}$) / 2] ± 0.025 | V |
| $I_L$ | ADC input leakage current[j] | - | - | 2.0 | µA |
| $R_{ADC}$ | ADC equivalent input resistance[j] | - | - | 2.5 | kΩ |
| $C_{ADC}$ | ADC equivalent input capacitance[j] | - | - | 10 | pF |
| $R_S$ | Analog source resistance[j] | - | - | 500 | Ω |
| SAMPLING DYNAMICS | | | | | |
| $F_{ADC}$ | ADC conversion clock frequency[k] | - | 16 | - | MHz |
| $F_{CONV}$ | ADC conversion rate | | 1 | | Msps |
| $T_S$ | ADC sample time | - | 250 | - | ns |
| $T_C$ | ADC conversion time[l] | - | 1 | - | µs |
| $T_{LT}$ | Latency from trigger to start of conversion | - | 2 | - | ADC clocks |
| SYSTEM PERFORMANCE when using external reference[mn] | | | | | |
| N | Resolution | | 12 | | bits |
| INL | Integral nonlinearity error, over full input range | - | ±1.5 | ±3.0 | LSB |
| DNL | Differential nonlinearity error, over full input range | - | ±0.8 | +2.0/-1.0[o] | LSB |
| $E_O$ | Offset error | - | ±1.0 | ±3.0 | LSB |

**Table 32-44. ADC Electrical Characteristics for ADC at 1 Msps *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $E_G$ | Gain error[p] | - | ±2.0 | ±3.0 | LSB |
| $E_T$ | Total unadjusted error, over full input range[q] | - | ±2.5 | ±4.0 | LSB |
| SYSTEM PERFORMANCE when using internal reference | | | | | |
| N | Resolution | | 12 | | bits |
| INL | Integral nonlinearity error, over full input range | - | ±1.5 | ±3.0 | LSB |
| DNL | Differential nonlinearity error, over full input range | - | ±0.8 | +2.0/-1.0[o] | LSB |
| $E_O$ | Offset error | - | ±5.0 | ±15.0 | LSB |
| $E_G$ | Gain error[p] | - | ±10.0 | ±30.0 | LSB |
| $E_T$ | Total unadjusted error, over full input range[q] | - | ±10.0 | ±30.0 | LSB |
| DYNAMIC CHARACTERISTICS[rs] | | | | | |
| $SNR_D$ | Signal-to-noise-ratio, Differential input, $V_{ADCIN}$: -20dB FS, 1KHz [t] | 70 | 72 | - | dB |
| $SDR_D$ | Signal-to-distortion ratio, Differential input, $V_{ADCIN}$: -3dB FS, 1KHz[tuv] | 72 | 75 | - | dB |
| $SNDR_D$ | Signal-to-Noise+Distortion ratio, Differential input, $V_{ADCIN}$: -3dB FS, 1KHz[twx] | 68 | 70 | - | dB |
| $SNR_S$ | Signal-to-noise-ratio, Single-ended input, $V_{ADCIN}$: -20dB FS, 1KHz [y] | 60 | 65 | - | dB |
| $SDR_S$ | Signal-to-distortion ratio, Single-ended input, $V_{ADCIN}$: -3dB FS, 1KHz[uv] | 70 | 72 | - | dB |
| $SNDR_S$ | Signal-to-Noise+Distortion ratio, Single-ended input, $V_{ADCIN}$: -3dB FS, 1KHz[ywx] | 60 | 63 | - | dB |
| TEMPERATURE SENSOR | | | | | |
| $V_{TSENS}$ | Temperature sensor voltage, junction temperature 25 °C | - | 1.633 | - | V |
| $S_{TSENS}$ | Temperature sensor slope at:<br>-40°C to 85 °C ambient (industrial temperature part)<br>-40°C to 105 °C ambient (extended temperature part) | - | -13.3 | - | mV/°C |
| $E_{TSENS}$ | Temperature sensor accuracy at:[z]<br>-40°C to 85 °C ambient (industrial temperature part)<br>-40°C to 105 °C ambient (extended temperature part) | - | - | ±5 | °C |

a. Values are at $V_{REF+}$= 3.3V, $F_{ADC}$=16 MHz unless otherwise noted.

b. Best design practices suggest that static or quiet digital I/O signals be configured adjacent to sensitive analog inputs to reduce capacitive coupling and cross talk. Unexpected results can occur if a switching digital I/O is placed adjacent to an ADC input channel or voltage reference input. In addition, analog signals configured adjacent to ADC input channels or reference inputs must meet the $R_{ADC}$ equivalent input resistance given in this table and must be band-limited to 100 kHz or lower.

c. Two capacitors in parallel. Note that these capacitors should be as close to the die as possible.

d. Assumes external filtering network between VREFA+ and VREFA- as shown in Figure 32-28 on page 2143. External reference noise level must be under 12bit (-74 dB) of Full Scale input, over input bandwidth, measured at VREFA+ - VREFA-.

e. Internal reference is connected directly between $V_{DDA}$ and GNDA (VREFi = $V_{DDA}$ - GNDA). In this mode, $E_O$, $E_G$, $E_T$, and dynamic specifications are adversely affected due to internal voltage drop and noise on $V_{DDA}$ and GNDA. Internal reference voltage is selected when VREF field in the **ADCCTL** register is 0x0.

f. $V_{ADCIN} = V_{INP} - V_{INN}$

g. With signal common mode as $V_{DDA}/2$.

h. With signal common mode as $(V_{REF+} + V_{REF-})/2$.

i. This parameter is defined as the average of the differential inputs.

j. As shown in Figure 32-29 on page 2143, $R_{ADC}$ is the total equivalent resistance in the input line all the way up to the sampling node at the input of the ADC.

k. See “System Clock Specification with ADC Operation” on page 2119 for full ADC clock frequency specification.

l. ADC conversion time (Tc) includes the ADC sample time (Ts).

m. Low noise environment is assumed in order to obtain values close to spec. Board must have good ground isolation between analog and digital grounds, a clean reference voltage is assumed, and input signal must be bandlimited to Nyquist bandwidth. No anti-aliasing filter is provided internally.

n. ADC static measurements taken by averaging over several samples. At least 20-sample averaging is assumed to obtain expected typical or maximum spec values.

o. 12-bit DNL

p. Gain error is measured at max code after compensating for offset. Gain error is equivalent to "Full Scale Error." It can be given in % of slope error, or in LSB, as done here.

q. Total Unadjusted Error is the maximum error at any one code versus the ideal ADC curve. It includes all other errors (offset error, gain error and INL) at any given ADC code.

r. A low noise environment is assumed in order to obtain values close to spec. The board must have good ground isolation between analog and digital grounds and a clean reference voltage. The input signal must be band-limited to Nyquist bandwidth. No anti-aliasing filter is provided internally.

s. ADC dynamic characteristics are measured using low-noise board design, with low-noise reference voltage ( < -74dB noise level in signal BW) and low-noise analog supply voltage. Board noise and ground bouncing couple into the ADC and affect dynamic characteristics. Clean external reference must be used to achieve shown specs.

t. Differential signal with correct common mode, applied between two ADC inputs.

u. SDR = -THD in dB.

v. For higher frequency inputs, degradation in SDR should be expected.

w. SNDR = S/(N+D) = SINAD (in dB)

x. Effective number of bits (ENOB) can be calculated from SNDR: ENOB = (SNDR - 1.76) / 6.02.

y. Single ended inputs are more sensitive to board and trace noise than differential inputs; SNR and SNDR measurements on single-ended inputs are highly dependent on how clean the test set-up is. If the input signal is not well-isolated on the board, higher noise than specified could potentially be seen at the ADC output.

z. Note that this parameter does not include ADC error.

### Table 32-45. ADC Electrical Characteristics for ADC at 2 Msps[ab]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| POWER SUPPLY REQUIREMENTS | | | | | |
| $V_{DDA}$ | ADC supply voltage | 2.97 | 3.3 | 3.63 | V |
| GNDA | ADC ground voltage | - | 0 | - | V |
| VDDA / GNDA VOLTAGE REFERENCE | | | | | |
| $C_{REF}$ | Voltage reference decoupling capacitance | - | 1.0 // 0.01[c] | - | µF |
| EXTERNAL VOLTAGE REFERENCE INPUT | | | | | |
| $V_{REFA+}$ | Positive external voltage reference for ADC, when VREF field in the **ADCCTL** register is 0x1[d]- | 2.4 | $V_{DDA}$ | $V_{DDA}$ | V |
| $V_{REFA-}$ | Negative external voltage reference for ADC, when VREF field in the **ADCCTL** register is 0x1[d] | GNDA | GNDA | 0.3 | V |
| $I_{VREF}$ | Current on VREF+ input, using external $V_{REF+}$ = 3.3 V | - | 330.5 | 440 | µA |
| $I_{LVREF}$ | DC leakage current on VREF+ input when external VREF disabled | - | - | 2.0 | µA |
| $C_{REF}$ | External reference decoupling capacitance[d] | - | 1.0 // 0.01[c] | - | µF |

**Table 32-45. ADC Electrical Characteristics for ADC at 2 Msps *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| ANALOG INPUT | | | | | |
| $V_{ADCIN}$ | Single-ended, full- scale analog input voltage, internal reference[ef] | 0 | - | $V_{DDA}$ | V |
| | Differential, full-scale analog input voltage, internal reference[eg] | $-V_{DDA}$ | - | $V_{VDDA}$ | V |
| | Single-ended, full-scale analog input voltage, external reference[df] | $V_{REFA-}$ | - | $V_{REFA+}$ | V |
| | Differential, full-scale analog input voltage, external reference[dh] | $-(V_{REFA+} - V_{REFA-})$ | - | $V_{REFA+} - V_{REFA-}$ | V |
| $VIN_{CM}$ | Input common mode voltage, differential mode[i] | - | - | $[(V_{REFA+} + V_{REFA-}) / 2] \pm 0.025$ | V |
| $I_L$ | ADC input leakage current[j] | - | - | 2.0 | µA |
| $R_{ADC}$ | ADC equivalent input resistance[j] | - | - | 2.5 | kΩ |
| $C_{ADC}$ | ADC equivalent input capacitance[j] | - | - | 10 | pF |
| $R_S$ | Analog source resistance[j] | - | - | 250 | Ω |
| SAMPLING DYNAMICS | | | | | |
| $F_{ADC}$ | ADC conversion clock frequency[k] | - | 32 | - | MHz |
| $F_{CONV}$ | ADC conversion rate | | 2 | | Msps |
| $T_S$ | ADC sample time | - | 125 | - | ns |
| $T_C$ | ADC conversion time[l] | - | 0.5 | - | µs |
| $T_{LT}$ | Latency from trigger to start of conversion | - | 2 | - | ADC clocks |
| SYSTEM PERFORMANCE when using external reference[mn] | | | | | |
| N | Resolution | | 12 | | bits |
| INL | Integral nonlinearity error, over full input range | - | ±1.5 | ±3.0 | LSB |
| DNL | Differential nonlinearity error, over full input range | - | ±0.8 | +2.0/-1.0[o] | LSB |
| $E_O$ | Offset error | - | ±1.0 | ±3.0 | LSB |
| $E_G$ | Gain error[p] | - | ±2.0 | ±3.0 | LSB |
| $E_T$ | Total unadjusted error, over full input range[q] | - | ±2.5 | ±4.0 | LSB |
| SYSTEM PERFORMANCE when using internal reference | | | | | |
| N | Resolution | | 12 | | bits |
| INL | Integral nonlinearity error, over full input range | - | ±1.5 | ±3.0 | LSB |
| DNL | Differential nonlinearity error, over full input range | - | ±0.8 | +2.0/-1.0[o] | LSB |
| $E_O$ | Offset error | - | ±5.0 | ±15.0 | LSB |
| $E_G$ | Gain error[p] | - | ±10.0 | ±30.0 | LSB |
| $E_T$ | Total unadjusted error, over full input range[q] | - | ±10.0 | ±30.0 | LSB |
| DYNAMIC CHARACTERISTICS[rs] | | | | | |
| $SNR_D$ | Signal-to-noise-ratio, Differential input, $V_{ADCIN}$: -20dB FS, 1KHz [t] | 68 | 72 | - | dB |
| $SDR_D$ | Signal-to-distortion ratio, Differential input, $V_{ADCIN}$: -3dB FS, 1KHz[tuv] | 70 | 75 | - | dB |

**Table 32-45. ADC Electrical Characteristics for ADC at 2 Msps *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $SNDR_D$ | Signal-to-Noise+Distortion ratio, Differential input, $V_{ADCIN}$: -3dB FS, 1KHz[twx] | 65 | 70 | - | dB |
| $SNR_S$ | Signal-to-noise-ratio, Single-ended input, $V_{ADCIN}$: -20dB FS, 1KHz[y] | 58 | 65 | - | dB |
| $SDR_S$ | Signal-to-distortion ratio, Single-ended input, $V_{ADCIN}$: -3dB FS, 1KHz[uv] | 68 | 72 | - | dB |
| $SNDR_S$ | Signal-to-Noise+Distortion ratio, Single-ended input, $V_{ADCIN}$: -3dB FS, 1KHz[ywx] | 58 | 63 | - | dB |

a. Values are at $V_{REF+}$= 3.3V, $F_{ADC}$=32 MHz unless otherwise noted.

b. Best design practices suggest that static or quiet digital I/O signals be configured adjacent to sensitive analog inputs to reduce capacitive coupling and cross talk. Unexpected results can occur if a switching digital I/O is placed adjacent to an ADC input channel or voltage reference input. In addition, analog signals configured adjacent to ADC input channels or reference inputs must meet the $R_{ADC}$ equivalent input resistance given in this table and must be band-limited to 100 kHz or lower.

c. Two capacitors in parallel. Note that these capacitors should be as close to the die as possible.

d. Assumes external filtering network between VREFA+ and VREFA- as shown in Figure 32-28 on page 2143. External reference noise level must be under 12bit (-74 dB) of Full Scale input, over input bandwidth, measured at VREFA+ - VREFA-.

e. Internal reference is connected directly between $V_{DDA}$ and GNDA (VREFi = $V_{DDA}$ - GNDA). In this mode, $E_O$, $E_G$, $E_T$, and dynamic specifications are adversely affected due to internal voltage drop and noise on $V_{DDA}$ and GNDA. Internal reference voltage is selected when VREF field in the **ADCCTL** register is 0x0.

f. $V_{ADCIN} = V_{INP} - V_{INN}$

g. With signal common mode as $V_{DDA}/2$.

h. With signal common mode as $(V_{REF+} + V_{REF-})/2$.

i. This parameter is defined as the average of the differential inputs.

j. As shown in Figure 32-29 on page 2143, $R_{ADC}$ is the total equivalent resistance in the input line all the way up to the sampling node at the input of the ADC.

k. See “System Clock Specification with ADC Operation” on page 2119 for full ADC clock frequency specification.

l. ADC conversion time (Tc) includes the ADC sample time (Ts).

m. Low noise environment is assumed in order to obtain values close to spec. Board must have good ground isolation between analog and digital grounds, a clean reference voltage is assumed, and input signal must be bandlimited to Nyquist bandwidth. No anti-aliasing filter is provided internally.

n. ADC static measurements taken by averaging over several samples. At least 20-sample averaging is assumed to obtain expected typical or maximum spec values.

o. 12-bit DNL

p. Gain error is measured at max code after compensating for offset. Gain error is equivalent to "Full Scale Error." It can be given in % of slope error, or in LSB, as done here.

q. Total Unadjusted Error is the maximum error at any one code versus the ideal ADC curve. It includes all other errors (offset error, gain error and INL) at any given ADC code.

r. A low noise environment is assumed in order to obtain values close to spec. The board must have good ground isolation between analog and digital grounds and a clean reference voltage. The input signal must be band-limited to Nyquist bandwidth. No anti-aliasing filter is provided internally.

s. ADC dynamic characteristics are measured using low-noise board design, with low-noise reference voltage ( < -74dB noise level in signal BW) and low-noise analog supply voltage. Board noise and ground bouncing couple into the ADC and affect dynamic characteristics. Clean external reference must be used to achieve shown specs.

t. Differential signal with correct common mode, applied between two ADC inputs.

u. SDR = -THD in dB.

v. For higher frequency inputs, degradation in SDR should be expected.

w. SNDR = S/(N+D) = SINAD (in dB)

x. Effective number of bits (ENOB) can be calculated from SNDR: ENOB = (SNDR - 1.76) / 6.02.

y. Single ended inputs are more sensitive to board and trace noise than differential inputs; SNR and SNDR measurements on single-ended inputs are highly dependent on how clean the test set-up is. If the input signal is not well-isolated on the board, higher noise than specified could potentially be seen at the ADC output.

**Figure 32-28. ADC External Reference Filtering**

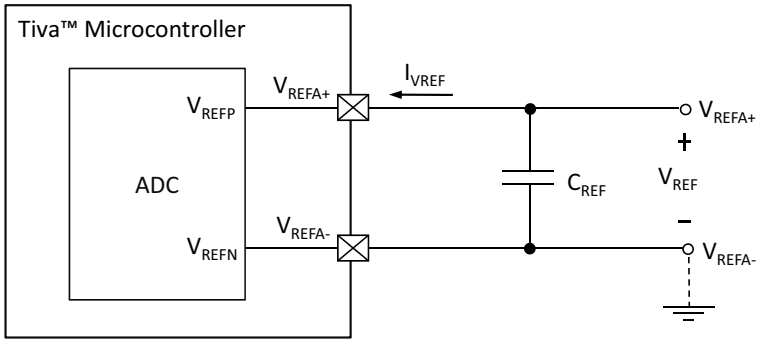
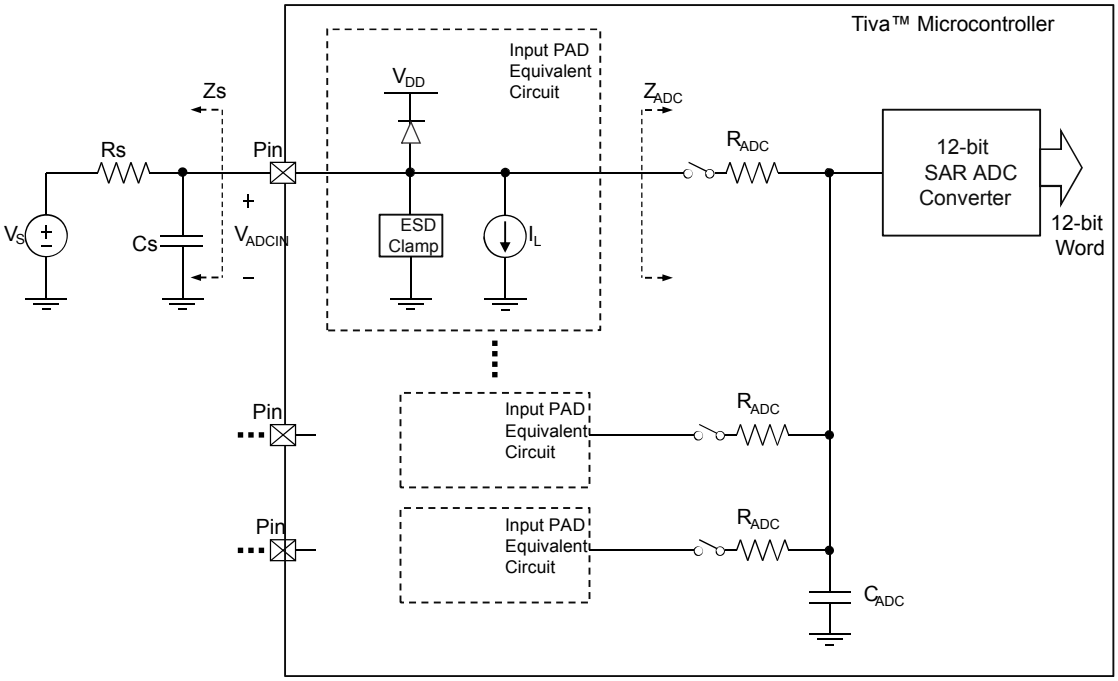



**Figure 32-29. ADC Input Equivalency**



# 32.17 Synchronous Serial Interface (SSI)

**Table 32-46. SSI Characteristics**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| S1 | $T_{CLK_PER}$ | SSIClk cycle time, as master[a] | 16.67 | - | - | ns |
| | | SSIClk cycle time, as slave[b] | 100 | - | - | ns |
| S2 | $T_{CLK_HIGH}$ | SSIClk high time, as master | 8.33 | - | - | ns |
| | | SSIClk high time, as slave | 50 | - | - | ns |
| S3 | $T_{CLK_LOW}$ | SSIClk low time, as master | 8.33 | - | - | ns |
| | | SSIClk low time, as slave | 50 | - | - | ns |
| S4 | $T_{CLKR}$ | SSIClk rise time[c] | 1.25 | - | - | ns |
| S5 | $T_{CLKF}$ | SSIClk fall time[c] | 1.25 | - | - | ns |
| S6 | $T_{TXDMOV}$ | Master Mode: Master Tx Data Output (to slave) Valid Time from edge of SSIClk | - | - | 4.00 | ns |
| S7 | $T_{TXDMOH}$ | Master Mode: Master Tx Data Output (to slave) Hold Time after next SSIClk | 0.60 | - | - | ns |
| S8 | $T_{RXDMS}$ | Master Mode: Master Rx Data In (from slave) setup time | 7.89 | - | - | ns |
| S9 | $T_{RXDMH}$ | Master Mode: Master Rx Data In (from slave) hold time | 0 | - | - | ns |
| S10 | $T_{TXDSOV}$ | Slave Mode: Master Tx Data Output (to Master) Valid Time from edge of SSIClk | - | - | 47.60[d] | ns |
| S11 | $T_{TXDSOH}$ | Slave Mode: Slave Tx Data Output (to Master) Hold Time from next SSIClk | 37.4[e] | - | - | ns |
| S13 | $T_{RXDSSU}$ | Slave Mode: Rx Data In (from master) setup time | 0 | - | - | ns |
| S14 | $T_{RXDSH}$ | Slave Mode: Rx Data In (from master) hold time | 37.03[f] | - | - | ns |

a. In master mode, the system clock must be at least twice as fast as the SSIClk.

b. In slave mode, the system clock must be at least 12 times faster than the SSIClk.

c. Note that the delays shown are using 12-mA drive strength.

d. This MAX value is for the minimum slave mode $T_{SYSCLK}$ period (8.33 ns). To find the MAX $T_{TXDSOV}$ value for a larger $T_{SYSCLK}$, use the equation: $4*T_{SYSCLK}+14.25$.

e. This MIN value is for the minimum slave mode $T_{SYSCLK}$ (8.33 ns). To find the MIN $T_{TXDSOH}$ value for a larger $T_{SYSCLK}$, use the equation: $4*T_{SYSCLK}+4.08$.

f. This MIN value is for the minimum slave mode $T_{SYSCLK}$ (8.33 ns). To find the MIN $T_{TXDSH}$ value for a larger $T_{SYSCLK}$, use the equation: $4*T_{SYSCLK}+3.70$.

**Figure 32-30. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement**

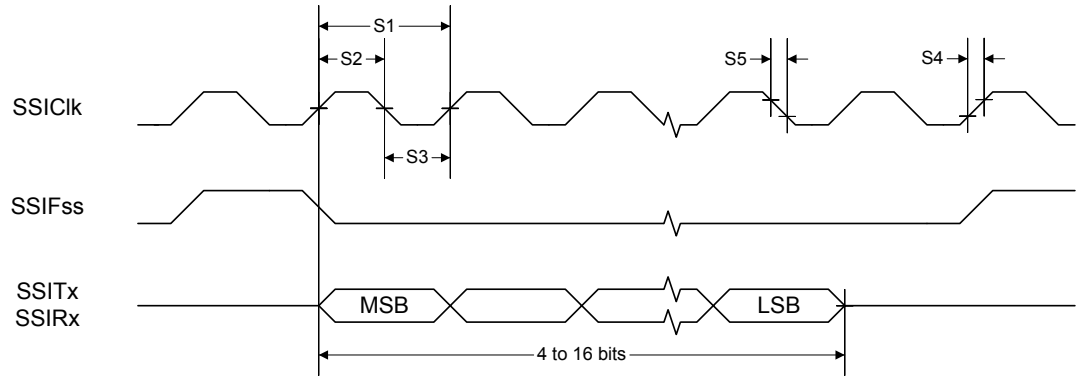



**Figure 32-31. Master Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1**

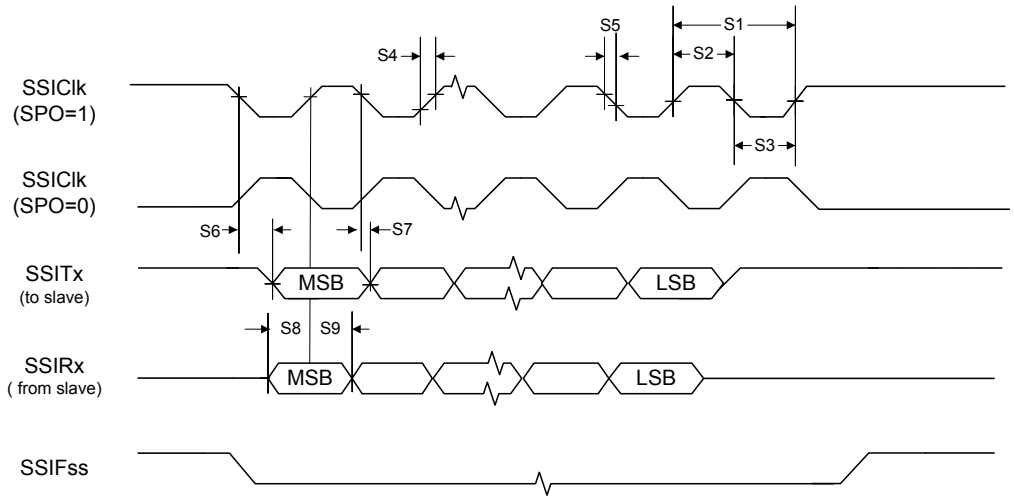


**Figure 32-32. Slave Mode SSI Timing for SPI Frame Format (FRF=00), with SPH=1**

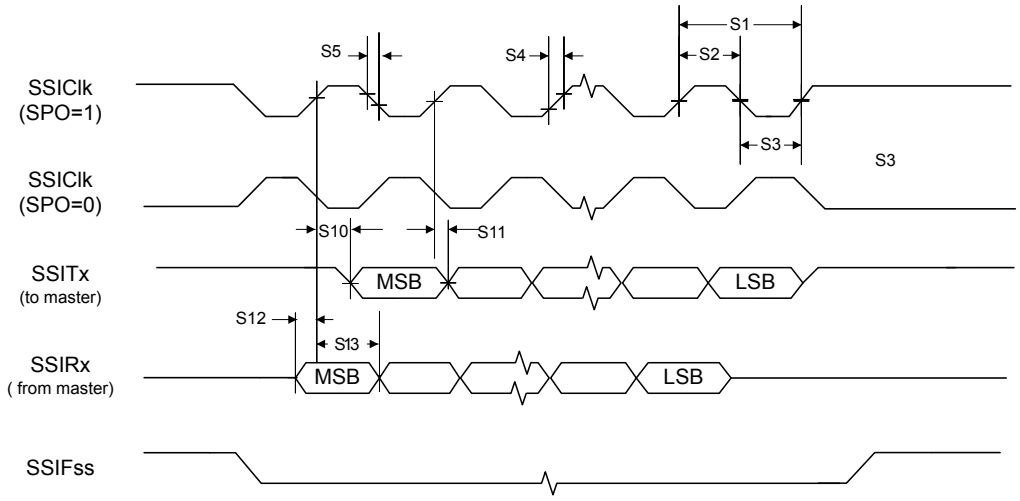



**Table 32-47. Bi- and Quad-SSI Characteristics**[a]

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| S15 | $T_{CLK_PER}$ | SSIClk cycle time, as master[b] | 16.67 | - | - | ns |
| S16 | $T_{CLK_HIGH}$ | SSIClk high time, as master | 8.33 | - | - | ns |
| S17 | $T_{CLK_LOW}$ | SSIClk low time, as master | 8.33 | - | - | ns |
| S18 | $T_{CLKR}$ | SSIClk rise time[c] | 1.25 | - | - | ns |
| S19 | $T_{CLKF}$ | SSIClk fall time[c] | 1.25 | - | - | ns |
| S20 | $T_{TXDMOV}$ | Master Mode: Master SSInXDATn Data Output (to slave) Valid Time from edge of SSIClk | - | - | 4.04 | ns |
| S21 | $T_{TXDMOH}$ | Master Mode: Master SSInXDATn Data Output (to slave) Hold Time after next SSIClk | 0.60 | - | - | ns |
| S22 | $T_{RXDMS}$ | Master Mode: Master SSInXDATn Data In (from slave) setup time | 5.78 | - | - | ns |
| S23 | $T_{RXDMH}$ | Master Mode: Master SSInXDATn Data In (from slave) hold time | 0 | - | - | ns |

a. Parameters S15 through S23 correspond to parameters S1 through S9 in Figure 32-30 and Figure 32-31.

b. In master mode, the system clock must be at least twice as fast as the SSIClk.

c. Note that the delays shown are using 12-mA drive strength.

## 32.18 Inter-Integrated Circuit ($I^2C$) Interface

### Table 32-48. $I^2C$ Characteristics

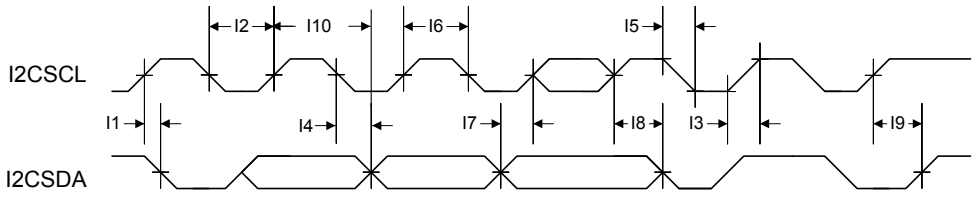
| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| I1[a] | $T_{SCH}$ | Start condition hold time | 36 | - | - | system clocks |
| I2[a] | $T_{LP}$ | Clock Low period | 36 | - | - | system clocks |
| I3[b] | $T_{SRT}$ | I2CSCL/I2CSDA rise time ($V_{IL}$ =0.5 V to $V_{IH}$ =2.4 V) | - | - | (see note b) | ns |
| I4 | $T_{DH}$ | Data hold time (slave) | - | 2 | - | system clocks |
| | | Data hold time (master) | - | 7 | - | system clocks |
| I5[c] | $T_{SFT}$ | I2CSCL/I2CSDA fall time ($V_{IH}$ =2.4 V to $V_{IL}$ =0.5 V) | - | 9 | 10 | ns |
| I6[a] | $T_{HT}$ | Clock High time | 24 | - | - | system clocks |
| I7 | $T_{DS}$ | Data setup time | 18 | - | - | system clocks |
| I8[a] | $T_{SCSR}$ | Start condition setup time (for repeated start condition only) | 36 | - | - | system clocks |
| I9[a] | $T_{SCS}$ | Stop condition setup time | 24 | - | - | system clocks |
| I10 | $T_{DV}$ | Data Valid (slave) | - | 2 | - | system clocks |
| | | Data Valid (master) | - | (6 * (1 + TPR)) + 1 | - | system clocks |

a. Values depend on the value programmed into the TPR bit in the **$I^2C$ Master Timer Period (I2CMTPR)** register; a TPR programmed for the maximum I2CSCL frequency (TPR=0x2) results in a minimum output timing as shown in the table above. The $I^2C$ interface is designed to scale the actual data transition time to move it to the middle of the I2CSCL Low period. The actual position is affected by the value programmed into the TPR; however, the numbers given in the above values are minimum values.

b. Because I2CSCL and I2CSDA operate as open-drain-type signals, which the controller can only actively drive low, the time I2CSCL or I2CSDA takes to reach a high level depends on external signal capacitance and pull-up resistor values.

c. Specified at a nominal 50 pF load.

### Figure 32-33. $I^2C$ Timing



# 32.19 Ethernet Controller

## 32.19.1 DC Characteristics

The parameters listed in Table 32-49 on page 2148, with the exception of $R_{BIAS}$, apply to transmit pins of the Ethernet PHY, which are generally the EN0TXOP and EN0TX0N signals during standard operation but can also be the EN0RXIN EN0RXIP signals if Auto-MDIX is enabled.

**Table 32-49. Ethernet PHY DC Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $R_{BIAS}$ | Value of the pull-down resistor on the RBIAS pin | 4.82 | 4.87 | 4.92 | kΩ |
| $V_{TPTD_100}$ | 100M transmit voltage | 0.95 | 1 | 1.05 | V |
| $V_{TPTDSYM}$ | 100M transmit voltage symmetry | -2 | - | 2 | % |
| $V_{OVRSHT}$ | Output overshoot | - | - | 5 | % |
| $V_{TPTD_10}$ | 10M transmit voltage | 2.2 | 2.5 | 2.8 | V |
| $V_{TH1}$ | 10Base-T Receive threshold | - | 200 | - | mV |

## 32.19.2 Clock Characteristics

**Table 32-50. MOSC 25-MHz Crystal Specification[a]**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N1 | $F_{MOSC25}$ | Frequency | - | 25 | - | MHz |
| - | $F_{TOL}$ | Frequency tolerance at operational temperature | 0 | - | ±50 | ppm |
| - | $T_{STA}$ | Frequency stability at 1-year aging | - | - | ±5 | ppm |

a. Refer to Table 32-23 on page 2115 for additional MOSC requirements.

**Figure 32-34. MOSC Crystal Characteristics for Ethernet**

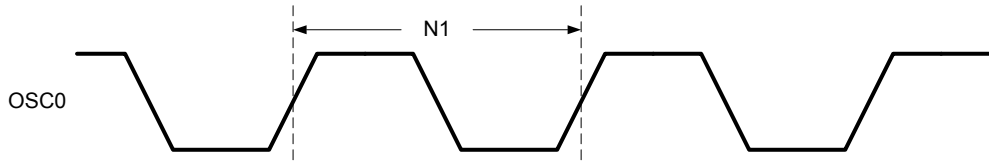



**Table 32-51. MOSC Single-Ended 25-MHz Oscillator Specification[a]**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N4 | $F_{OSC}$ | Frequency | - | 25 | - | MHz |
| - | $F_{TOL}$ | Frequency tolerance at operational temperature | 0 | - | ±50 | ppm |
| - | $T_{STA}$ | Frequency stability at 1-year aging | - | - | ±50 | ppm |
| N5 | $T_{RF}$ | Frequency rise and fall time | - | - | 1 | ns |

**Table 32-51. MOSC Single-Ended 25-MHz Oscillator Specification[a] *(continued)***

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| - | $T_J$ | Jitter (cycle-to-cycle) | - | 50 | - | ps |
| | | Jitter (over 10 ms) | - | - | 1 | ns |
| - | DC | Duty cycle | 40 | - | 60 | % |

**Figure 32-35. Single-Ended MOSC Characteristics for Ethernet**

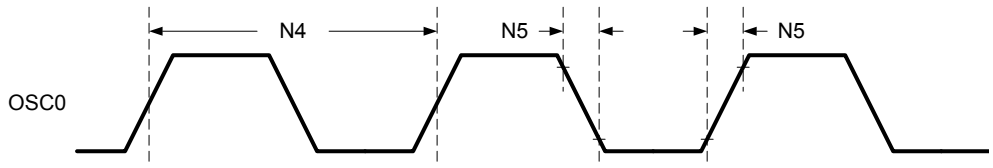
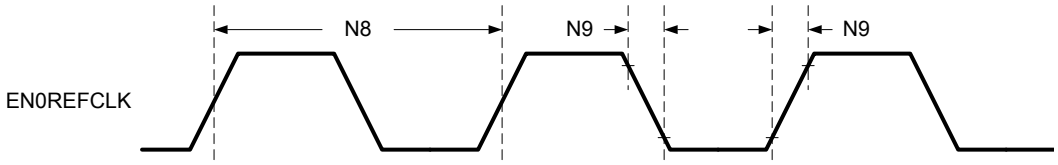



**Table 32-52. EN0RREF_CLK 50-MHz Oscillator Specification**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N8 | $F_{OSC}$ | Frequency | - | 50 | - | MHz |
| - | $F_{TOL}$ | Frequency tolerance at operational temperature | 0 | - | ±50 | ppm |
| - | $T_{STA}$ | Frequency stability at 1-year aging | - | - | ±50 | ppm |
| N9 | $T_{RF}$ | Frequency rise and fall time | - | - | 1.5 | ns |
| - | $T_J$ | Jitter (cycle-to-cycle) | - | 50 | - | ps |
| | | Jitter (over 10 ms) | - | - | 1 | ns |
| - | DC | Duty cycle | 40 | - | 60 | % |

**Figure 32-36. EN0RREF_CLK 50-MHz Oscillator Characteristics**




## 32.19.3 AC Characteristics

**Table 32-53. Ethernet Controller Enable and Software Reset Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N16 | $T_{EN}$ | Time from the System Control enable of the PHY to energy on the PMD output pin[ab] | 45 | - | - | µs |

**Table 32-53. Ethernet Controller Enable and Software Reset Timing *(continued)***

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N17 | $T_{SWRST}$ | Time from software reset of the PHY to energy on the PMD output pin | 110 | - | - | ns |

a. The PHY is enabled through System Control by setting the P0 bit in the **PCEPHY** register and the R0 bit in the **RCGCPHY** register.

b. This minimum timing assumes the PHYHOLD bit in the **EMACPC** register is not set.

**Figure 32-37. Reset Timing**

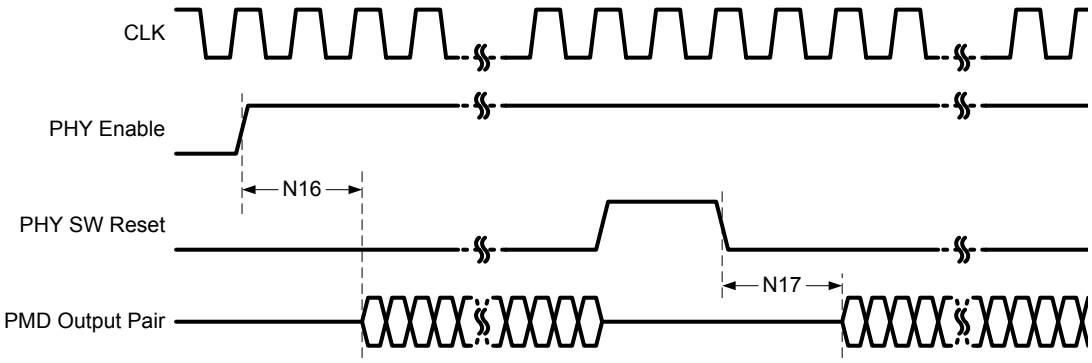



**Table 32-54. MII Serial Management Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N20 | $F_{MDC}$ | EN0MDC frequency | - | - | 2.5 | MHz |
| N21 | $T_{MDIO_DLY}$ | EN0MDC to EN0MDIO (output) delay time | 0 | - | 150 | ns |
| N22 | $T_{MDIO_SU}$ | EN0MDIO (input) to EN0MDC setup time | 10 | - | - | ns |
| N23 | $T_{MDIO_HLD}$ | EN0MDIO (input) to EN0MDC hold time | 10 | - | - | ns |

**Figure 32-38. Station Management Write and Read Timing**

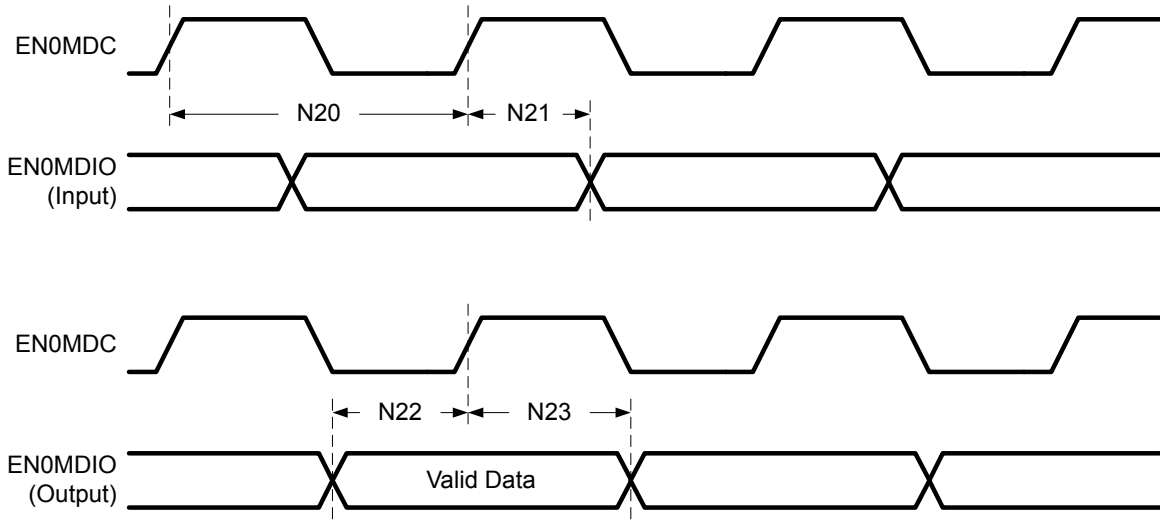



**Table 32-55. 100 Mb/s MII Transmit Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N26 | $T_{TXCK_HI}$ | EN0TXCK high time | 16 | 20 | 24 | ns |
| N27 | $T_{TXCK_LO}$ | EN0TXCK low time | 16 | 20 | 24 | ns |
| N28 | $T_{TX_DLY}$ | EN0TXCK to EN0TXDn, EN0TXEN delay | 0 | - | 25 | ns |

**Figure 32-39. 100 Mb/s MII Transmit Timing**

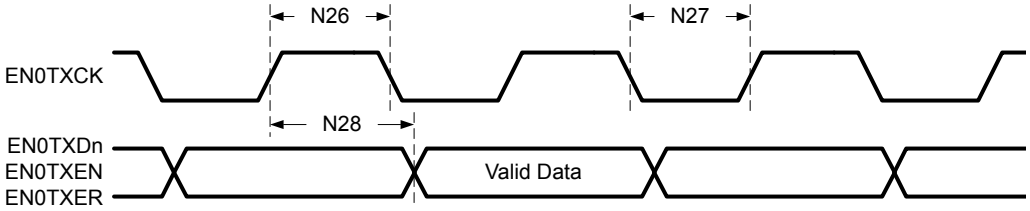



**Table 32-56. 100 Mb/s MII Receive Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N30 | $T_{RXCK_HI}$ | EN0RXCK high time | 16 | 20 | 24 | ns |
| N31 | $T_{RXCK_LO}$ | EN0RXCK low time | 16 | 20 | 24 | ns |
| N32 | $T_{RXCK_SU}$ | EN0RXDn, EN0RXDV, EN0RXER data setup to EN0RXCK | 10 | - | - | ns |
| N33 | $T_{RXCK_HLD}$ | EN0RXDn, EN0RXDV, EN0RXER data hold to EN0RXCK | 0 | - | - | ns |

**Figure 32-40. 100 Mb/s MII Receive Timing**

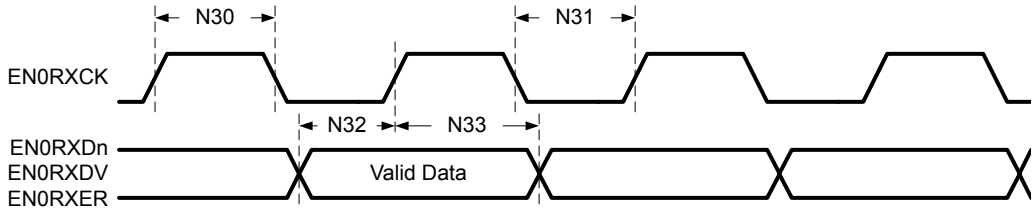



**Table 32-57. 100Base-TX Transmit Timing ($t_{R/F}$ and Jitter)**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N38 | $T_{RF}$ | 100 Mb/s PMD output pair $t_R$ and $t_F$[a] | 3 | 4 | 5 | ns |
| | $T_{RF_MM}$ | 100 Mb/s $t_R$ and $t_F$ symmetry[bc] | - | - | 500 | ps |
| N39 | $T_{RF_JTTR}$ | 100 Mb/s PMD Output Pair Transmit Jitter | - | - | 1.4 | ns |

a. Rise and fall times taken at 10% and 90% of the +1 or -1 amplitude.

b. Normal mismatch is the difference between the maximum and minimum of all rise and fall times

c. Choice of Ethernet transformer magnetics can affect this parameter.

**Figure 32-41. 100 Base-TX Transmit Timing**

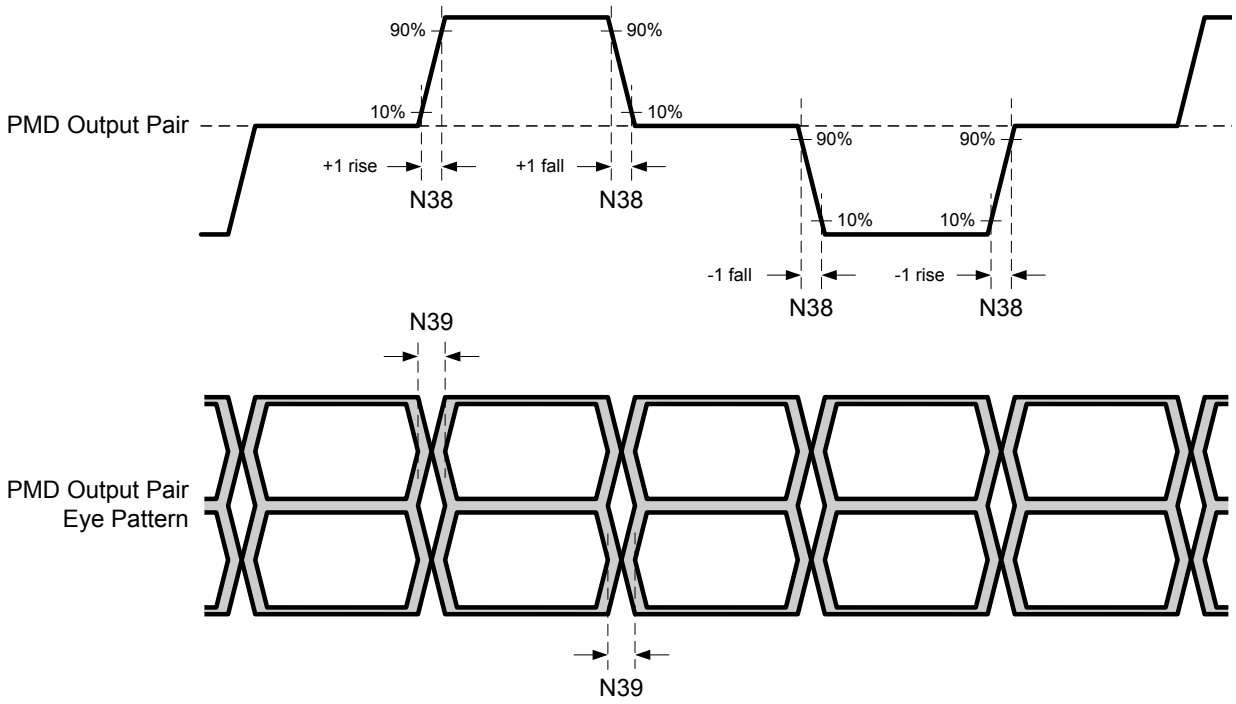


**Table 32-58. 10 Mb/s MII Transmit Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N45 | $T_{TXCK_HI}$ | EN0TXCK High Time | 190 | 200 | 210 | ns |
| N46 | $T_{TXCK_LO}$ | EN0TXCK Low Time | 190 | 200 | 210 | ns |
| N47 | $T_{TX_DLY}$ | EN0TXCK to EN0TXDn, EN0TXEN Delay | 0 | - | 25 | ns |

**Figure 32-42. 10 Mb/s MII Transmit Timing**

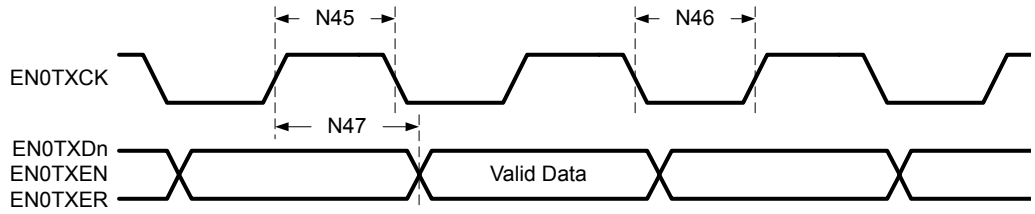



**Table 32-59. 10 Mb/s MII Receive Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N50 | $T_{RXCK_HI}$ | EN0RXCK High Time | 160 | 200 | 240 | ns |
| N51 | $T_{RXCK_LO}$ | EN0RXCK Low Time | 160 | 200 | 240 | ns |
| N52 | $T_{RX_SU}$ | EN0RXCK to EN0RXDn, EN0RXDV, EN0RXER setup | 100 | - | - | ns |
| N53 | $T_{RX_HLD}$ | EN0RXCK to EN0RXDn, EN0RXDV, EN0RXER hold | 100 | - | - | ns |

**Figure 32-43. 10 Mb/s MII Receive Timing**

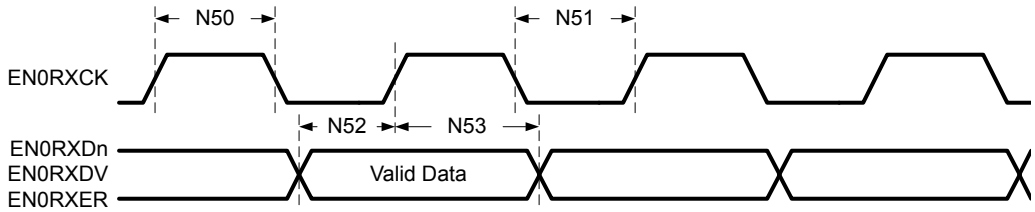
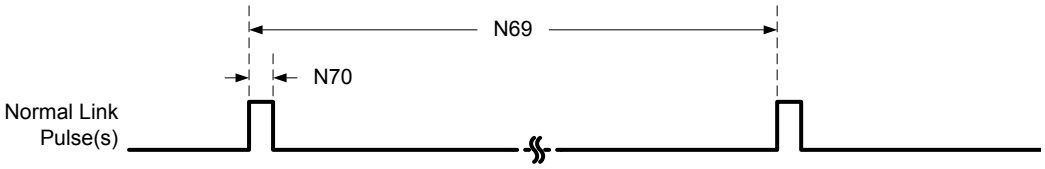



**Table 32-60. 10Base-T Normal Link Pulse Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N69 | $T_{LP_PER}$ | Link pulse period | - | 76 | - | ms |
| N70 | $T_{LP_WID}$ | Link pulse width | - | 100 | - | µs |

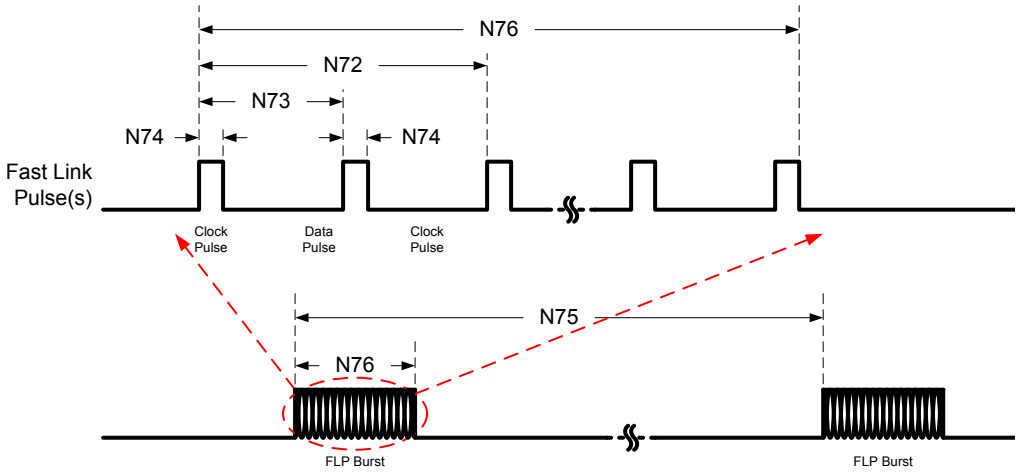
### Figure 32-44. 10Base-TX Normal Link Pulse Timing




### Table 32-61. Auto-Negotiation Fast Link Pulse (FLP) Timing

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N72 | $T_{CLKP}$ | Clock pulse to clock pulse period | - | 125 | - | µs |
| N73 | $T_{CLKDP}$ | Clock Pulse to Data Pulse Period | - | 62 | - | µs |
| N74 | $T_{PUL}$ | Clock, Data Pulse Width | - | 110 | - | ns |
| N75 | $T_{BRSTP}$ | FLP Burst to FLP Burst Period | - | 16 | - | ms |
| N76 | $T_{BRSTW}$ | Burst Width | - | 2 | - | ms |

### Figure 32-45. Auto-Negotiation Fast Link Pulse Timing




### Table 32-62. 100Base-TX Signal Detect Timing

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N79 | $T_{ON}$ | SD Internal Turn-on Time | - | - | 100 | µs |
| N80 | $T_{OFF}$ | Internal Turn-off Time | - | - | 200 | µs |

**Figure 32-46. 100Base-TX Signal Detect Timing**

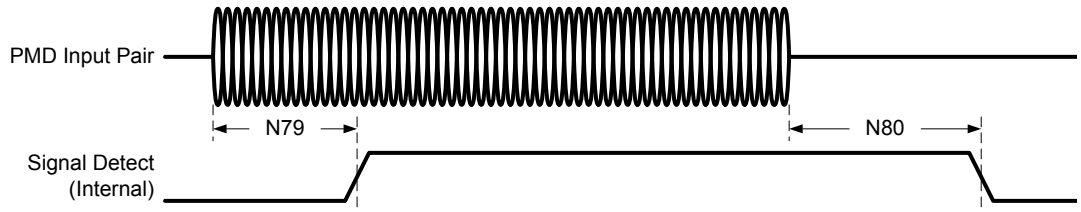



**Table 32-63. RMII Transmit Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N86 | $F_{REFCLK}$ | EN0RREF_CLK frequency | - | 50 | - | MHz |
| - | $DC_{REFCLK}$ | EN0RREF_CLK duty cycle | 40 | - | 60 | % |
| N87 | $T_{TX_DLY}$ | EN0RREF_CLK to EN0TXDn, EN0TXEN delay | 2 | - | 14 | ns |

**Figure 32-47. RMII Transmit Timing**

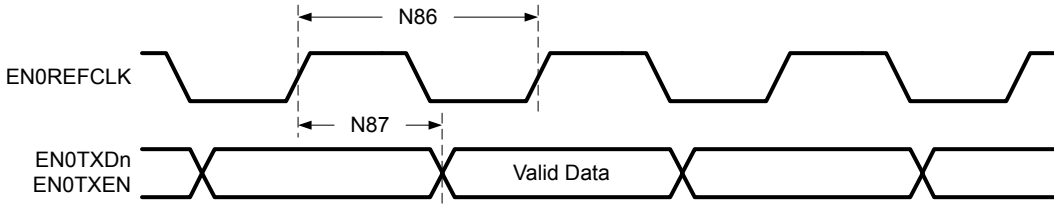
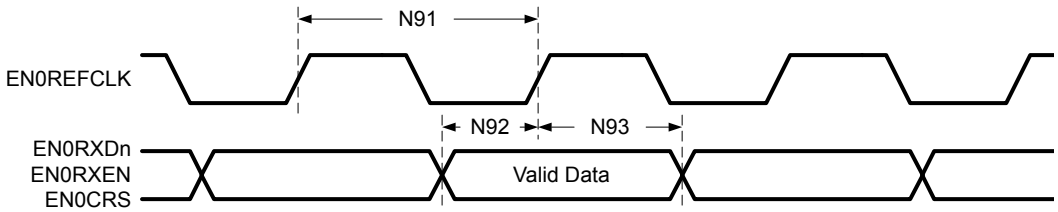



**Table 32-64. RMII Receive Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| N91 | $F_{REFCLK}$ | EN0RREF_CLK frequency | - | 50 | - | MHz |
| - | $DC_{REFCLK}$ | EN0RREF_CLK duty cycle | 40 | - | 60 | % |
| N92 | $T_{RX_SU}$ | EN0RXDn, EN0RXEN, EN0CRS setup time to EN0RREF_CLK | 4 | - | - | ns |
| N93 | $T_{RX_HLD}$ | EN0RXDn, EN0RXEN, EN0CRS hold time from EN0RREF_CLK | 2 | - | - | ns |

**Figure 32-48. RMII Receive Timing**



## 32.20 Universal Serial Bus (USB) Controller

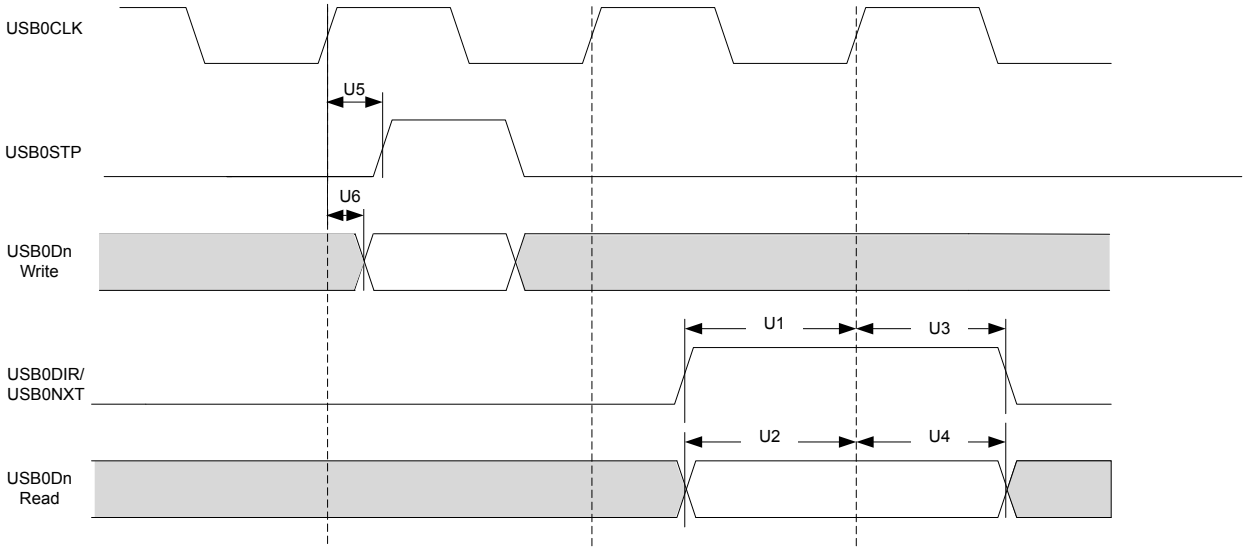
The Tiva™ C Series USB controller electrical specifications are compliant with the *Universal Serial Bus Specification Rev. 2.0* (full-speed and low-speed support) and the *On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0*. Some components of the USB system are integrated within the TM4C129XKCZAD microcontroller and specific to the Tiva™ C Series microcontroller design.

**Note:** GPIO pin, PB1, which can be configured as the USB0VBUS signal, is the only pin which is 5-V tolerant on the device.

**Table 32-65. ULPI Interface Timing**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| Timings with respect to external clock source input to USB0CLK | | | | | | |
| U1 | $T_{SUC}$ | Setup time (control in) USB0DIR, USB0NXT | 4.8 | - | - | ns |
| U2 | $T_{SUD}$ | Setup Time (data in) USB0Dn | 3.5 | - | - | ns |
| U3 | $T_{HTC}$ | Hold Time (control in) USB0DIR, USB0NXT | 0 | - | - | ns |
| U4 | $T_{HTD}$ | Hold Time (data in) USB0Dn | 0 | - | - | ns |
| U5 | $T_{ODC}$ | Output Delay (control out) USB0STP | 3.7 | - | 9.5 | ns |
| U6 | $T_{ODD}$ | Output Delay (data out) USB0Dn | 3.7 | - | 9.5 | ns |
| Timings with USB0CLK as clock output | | | | | | |
| U1 | $T_{SUC}$ | Setup time (control in) USB0DIR, USB0NXT | 6.0 | - | - | ns |
| U2 | $T_{SUD}$ | Setup Time (data in) USB0Dn | 4.6 | - | - | ns |
| U3 | $T_{HTC}$ | Hold Time (control in) USB0DIR, USB0NXT | 0 | - | - | ns |
| U4 | $T_{HTD}$ | Hold Time (data in) USB0Dn | 0 | - | - | ns |
| U5 | $T_{ODC}$ | Output Delay (control out) USB0STP | 4.0 | - | 10.6 | ns |
| U6 | $T_{ODD}$ | Output Delay (data out) USB0Dn | 4.0 | - | 10.6 | ns |

**Figure 32-49. ULPI Interface Timing Diagram**



## 32.21 LCD Controller

The LCD controller consists of two independent controllers, the raster controller and the LCD interface display driver (LIDD) controller. Each controller operates independently from the other and only one of them is active at any given time.

- The LIDD controller supports an asynchronous LCD interface. It provides full-timing programmability of control signals (CS, WE, OE, ALE) and output data. It also supports synchronous interface in Motorola 6800 mode and Intel 8080 mode through the use of the external LCDMCLK output.
- The raster controller handles the synchronous LCD interface. It provides timing and data for constant graphics refresh to a passive display. It supports a wide variety of monochrome and full-color display types and sizes by use of programmable timing controls, a built-in palette, and a gray-scale and serializer. Graphics data is processed and stored in frame buffers. A frame buffer is a contiguous memory block in the system. The µDMA supplies the graphics data to the raster engine which, in turn, outputs to the external LCD device.

The maximum resolution for the LCD controller is 2048 x 2048 pixels. The maximum frame rate is determined by the image size in combination with the pixel clock rate.

The LCD signals must be configured with a drive strength of 8 mA with slew rate control.

**Table 32-66. LCD Controller Load Capacitance Limits**

| Parameter | Parameter Name | Condition | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| $C_{LOAD}$ | LCD Output Load Capacitance | LIDD mode | 5 | - | 50 | pF |
| | | Raster mode | 3 | - | 30 | pF |

**Note:** A 25-pF load is assumed for LCD timings.

### 32.21.1 LCD Interface Display Driver (LIDD Mode)

In LIDD mode, the **LIDDCS0CFG** register allows for full programmability of the read and write strobes, data and enables. In the **LCDCS0CFG** register, the following parameters can be configured with respect to the internal MCLK:

- WRSU (bits[31:27]): Number of MCLK cycles that the LCDDATA bus, output enable, ADE, DIR and CS0 signals have to be ready before the write strobe. The minimum value is 0x0.
- WRDUR (bits[26:21]): Number of MCLK cycles for which the write strobe is held active when performing a write access. The minimum value is 0x1.
- WRHOLD (bits[20:17]): Number of MCLK cycles for which the LCDDATA bus, output enable, ALE, DIR and CS0 signals are held after the write strobe is deasserted when performing a write access. The minimum value is 0x1.
- RDSU (bits[16:12]): When performing a read access, this field defines the number of MCLK cycles that the LCDDATA bus, output enable, ALE, DIR and CS0 signals have to be ready before the read strobe is asserted.
- RDDUR (bits[11:6]): Number of MCLK cycles for which the read strobe is held active when performing a read access. The minimum value is 0x1.

- RDHOLD (bits[5:2]): Number of MCLK cycles for which the LCDDATA bus, output enable, ALE, DIR and CS0 signals are held after the read strobe is deasserted when performing a read access. The minimum value is 0x1.
- GAP (bits[1:0]): Number of MCLK cycles (GAP + 1) between the end of one CS0 device access and the start of another CS0 devices access unless the two accesses are both reads. In this case, this delay is not incurred. The minimum value is 0x0.

Essentially, output valid/hold and input setup/hold times in LIDD mode are programmed by these fields. In addition, the inherent clock delay from the internal MCLK transition to the signal output adds or subtracts to these programmed times as indicated in the following tables and figures.

**Table 32-67. LCD Switching Characteristics**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| L1 | $T_{CYC}$ | Internal MCLK cycle time | - | 50 | - | ns |
| L2 | $T_{CYCH}$ | Internal MCLK pulse duration high | - | 25 | - | ns |
| L3 | $T_{CYCL}$ | Internal MCLK pulse duration low | - | 25 | - | ns |
| L4 | $T_{DLYVAL}$ | Delay time from internal MCLK high to LCDDATA[15:0] valid (write) | - | - | 11.6 | ns |
| L5 | $T_{DLYINV}$ | Delay time from internal MCLK high to LCDDATA[15:0] invalid (write) | - | - | 4.0 | ns |
| L6 | $T_{DLYHAC}$ | Delay time, internal MCLK high to LCDAC | - | - | 11.19 | ns |
| L7 | $T_{TRANAC}$ | LCDAC transition time | - | - | 5.9 | ns |
| L8 | $T_{DLYFP}$ | Delay time from internal MCLK high to LCDFP | - | - | 10.5 | ns |
| L9 | $T_{TRANFP}$ | LCDFP transition time | - | - | 5.9 | ns |
| L10 | $T_{DLYLP}$ | Delay time internal MCLK high to LCDLP | - | - | 11.0 | ns |
| L11 | $T_{TRANLP}$ | LCDLP transition time | - | - | 5.9 | ns |
| L12 | $T_{DLYCP}$ | Delay time of internal MCLK to LCDCP | - | - | 8.4 | ns |
| L13 | $T_{TRANCP}$ | LCDCP transition time | - | - | 5.9 | ns |
| L14 | $T_{DLYDZ}$ | Delay time from internal MCLK high to LCDDATA[15:0] high-Z (read cycle) | - | - | 11.7 | ns |
| L15 | $T_{DLYDD}$ | Delay time from internal MCLK high to LCDDATA[15:0] active (read cycle) | - | - | 11.0 | ns |
| L19 | $T_{TRANMCLK}$ | Internal MCLK transition time | - | - | 5.9 | ns |
| L20 | $T_{TRANPKT}$ | LCDDATA transition time from one data packet to next. | - | - | 5.9 | ns |

**Table 32-68. Timing Requirements for LCDDATA in LIDD Mode**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| L16 | $T_{STDATA}$ | LCDDATA[15:0] setup time (read) before internal MCLK high | 20 | - | - | ns |
| L17 | $T_{HTDATA}$ | LCDDATA[15:0] hold time (read) after internal MCLK high | 0 | - | - | ns |
| L18 | $T_{TRANRDATA}$ | LCDDATA read transition time | 5.9 | - | - | ns |

### 32.21.1.1 Hitachi Mode

Hitachi mode is an asynchronous mode which does not use an external LCDMCLK and allows for full programmability of the read and write strobes, data and enables through the **LIDDCS0CFG** register. Note that in addition to the parameter delays configured in the **LIDDCS0CFG** register, the delay times in the associated figures shown below that are from the internal MCLK to the signal output must be added to the programmed delays for full timing information. All of the parameter values associated with the following figures can be found in Table 32-67 on page 2159 and Table 32-68 on page 2159.

In Figure 32-50 on page 2160, Figure 32-51 on page 2161, and Figure 32-52 on page 2161 the second LCDMCLK is shown as E1 and can be used as the enable strobe for a second display in Hitachi mode. The primary enable strobe is the LCDAC signal.

**Note:** The acronyms WRSU, WRDUR, WRHOLD and GAP in the following figures correspond to the bitfields of the **LIDDCS0CFG** register described in "LCD Interface Display Driver (LIDD Mode)" on page 2158.

**Figure 32-50. Command Write in Hitachi Mode**

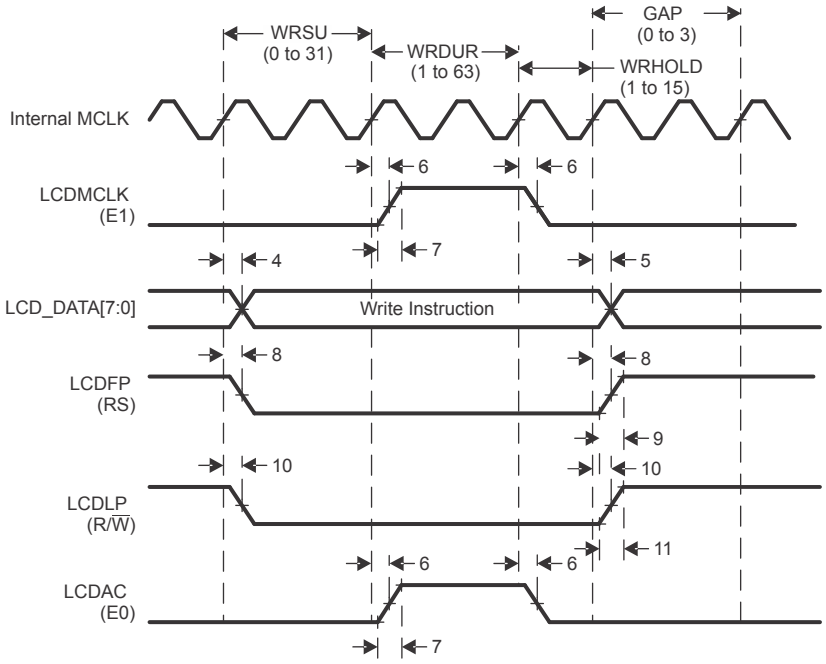


**Figure 32-51. Data Write in Hitachi Mode**

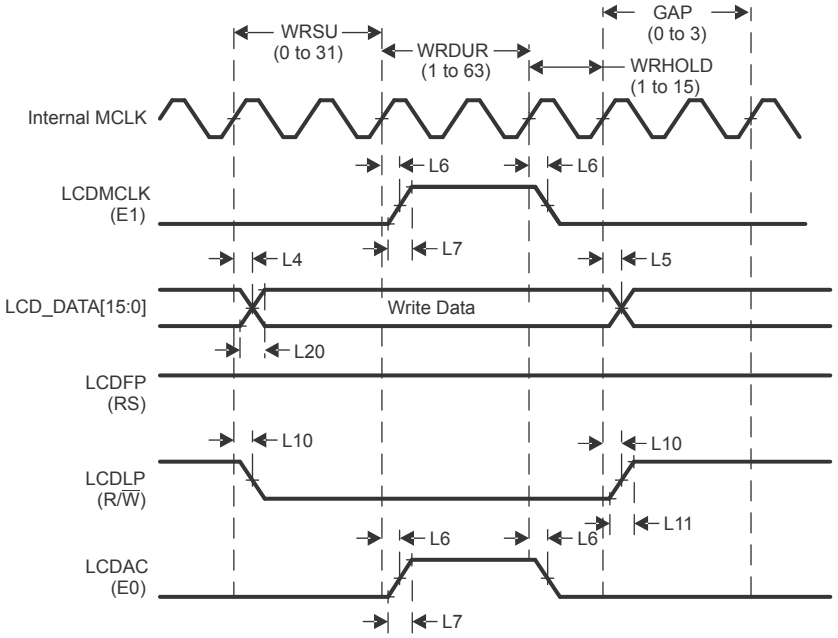



**Figure 32-52. Command Read in Hitachi Mode**

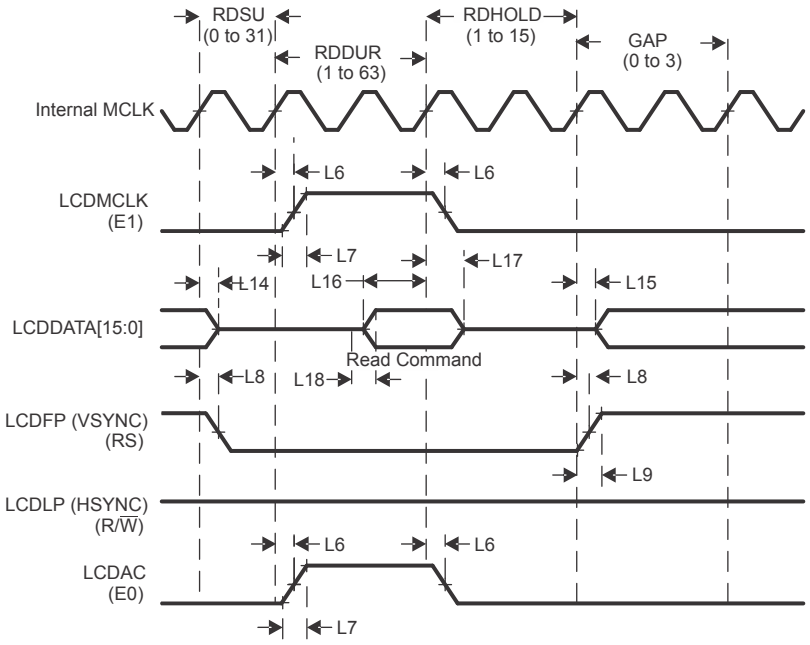
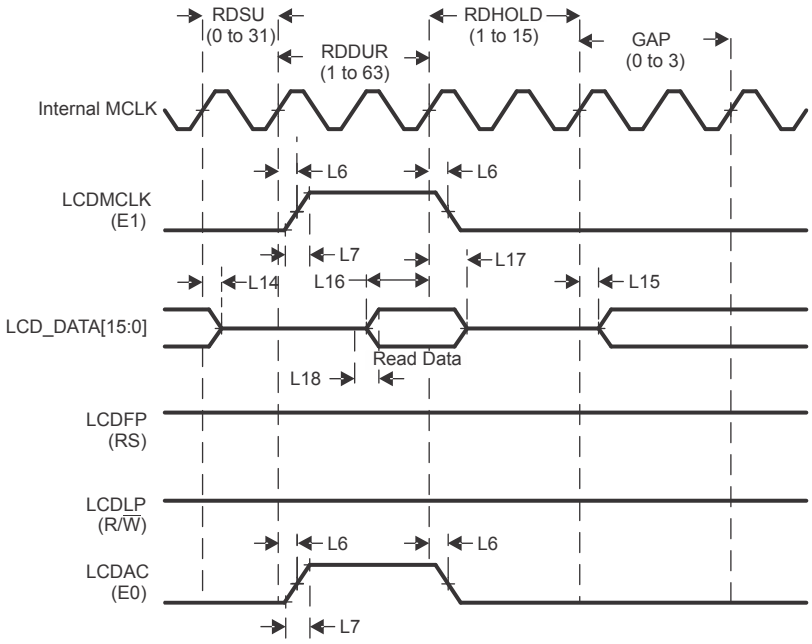


**Figure 32-53. Data Read in Hitachi Mode**




### 32.21.1.2 Motorola 6800 Mode

Motorola 6800 mode can be configured to perform asynchronous operations or synchronous operations. When configured in asynchronous mode, LCDMCLK is not required, so it performs the CS1 function. When configured in asynchronous mode, LCDMCLK is not required, so it performs the CS1 function. When configured in synchronous mode, MCLK is available externally through the signal LCDMCLK. Note that in asynchronous mode the internal MCLK shown represents the internal clock that sequences the other signals. All of the parameter values associated with the following figures can be found in Table 32-67 on page 2159 and Table 32-68 on page 2159.

**Note:** In asynchronous Motorola 6800 mode, LCDCP functions as the read and write strobe. In synchronous mode

**Note:** The acronyms WRSU, WRDUR, WRHOLD and GAP in the following figures correspond to the bitfields of the **LIDDCS0CFG** register described in “LCD Interface Display Driver (LIDD Mode)” on page 2158.

**Figure 32-54. Motorola 6800 Graphic Display Mode Write (Synchronous & Asynchronous Operation)**

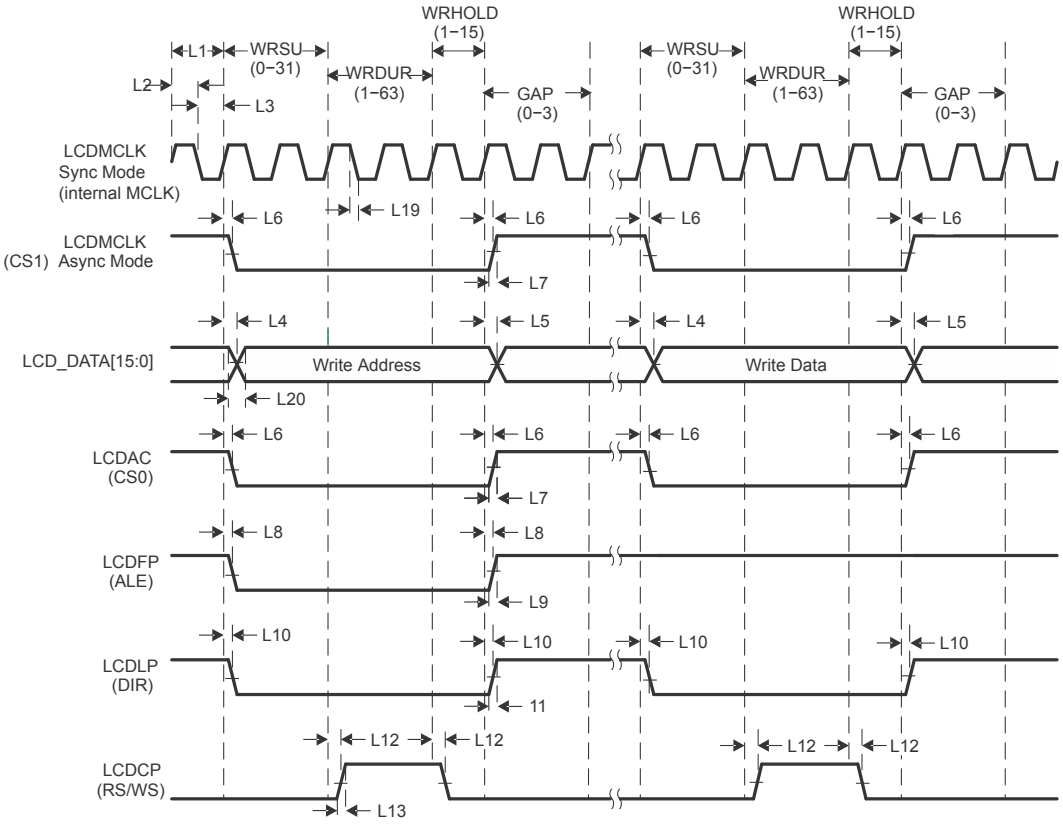


**Figure 32-55. Motorola 6800 Graphic Display Mode Read (Synchronous & Asynchronous Operation)**

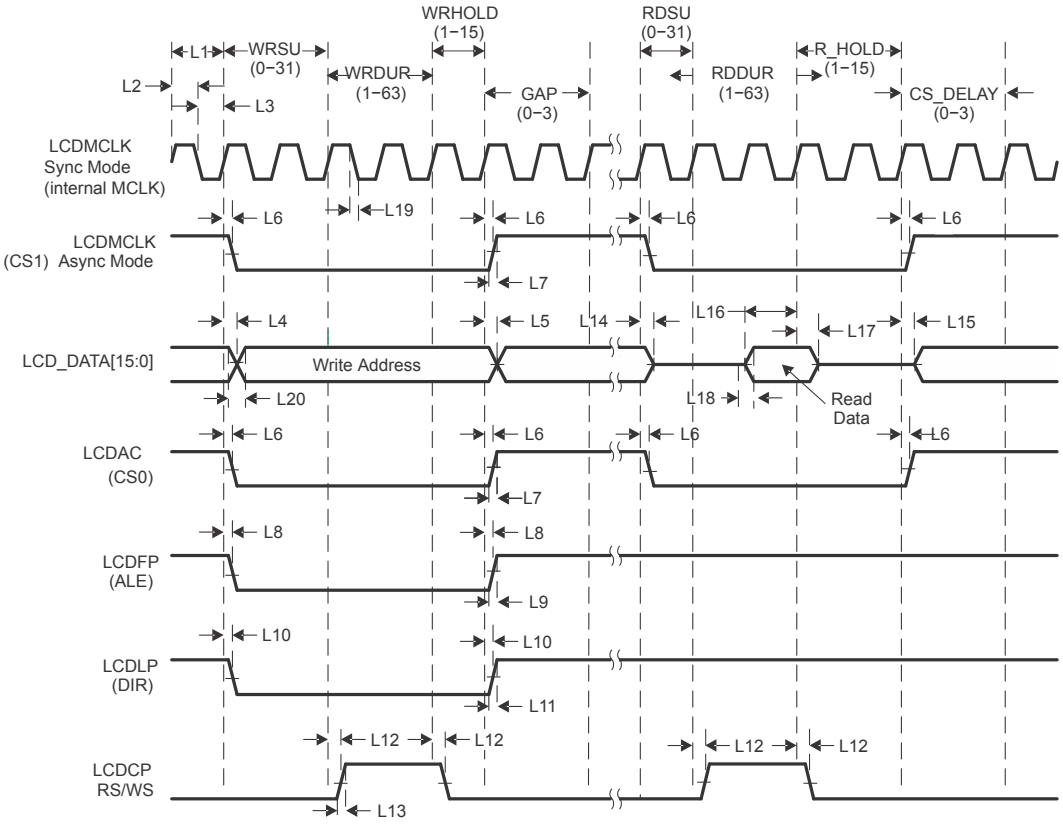
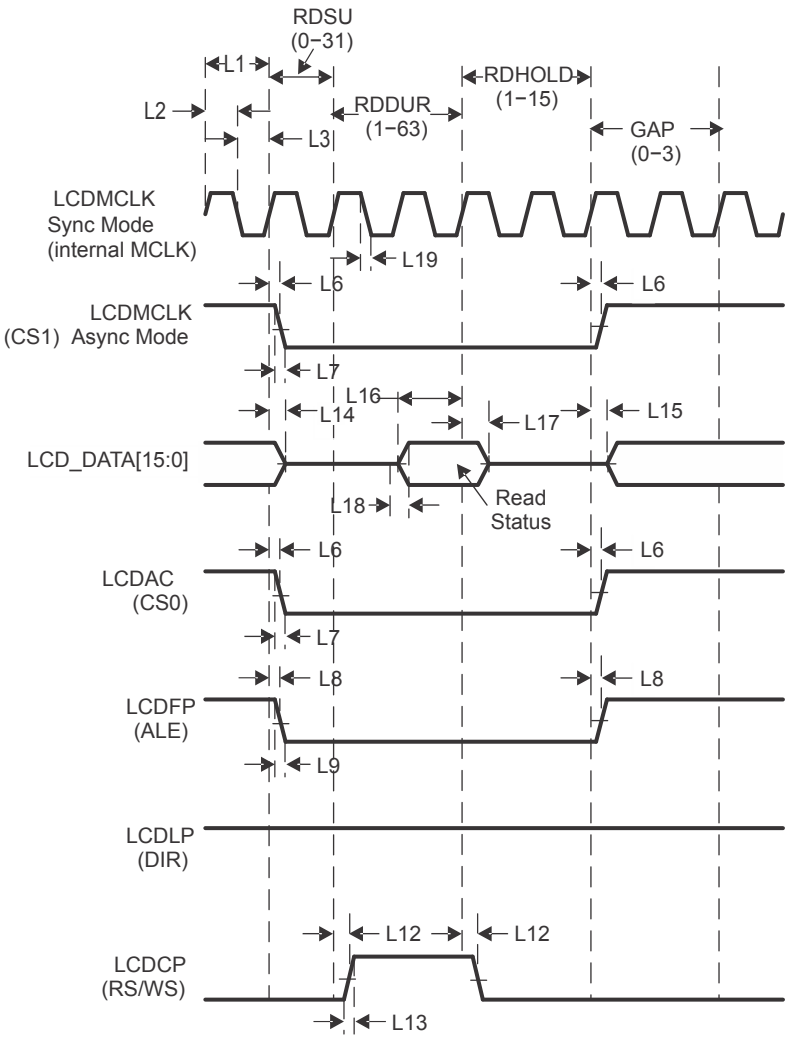


**Figure 32-56. Motorola 6800 Graphic Display Mode Status (Synchronous & Asynchronous Operation)**




### 32.21.1.3 Intel 8080 Mode

Intel mode can be configured to perform asynchronous operations or synchronous operations. When configured in asynchronous mode, LCDMCLK is not required, so it performs the CS1 function. When configured in synchronous mode, MCLK is available externally through the signal LCDMCLK. Note that in asynchronous mode the internal MCLK shown represents the internal clock that sequences the other signals. All of the parameter values associated with the following figures can be found in Table 32-67 on page 2159 and Table 32-68 on page 2159.

**Note:** The acronyms WRSU, WRDUR, WRHOLD and GAP in the following figures correspond to the bitfields of the **LIDDCS0CFG** register described in “LCD Interface Display Driver (LIDD Mode)” on page 2158.

**Figure 32-57. Micro-Interface Graphic Display Intel Write**

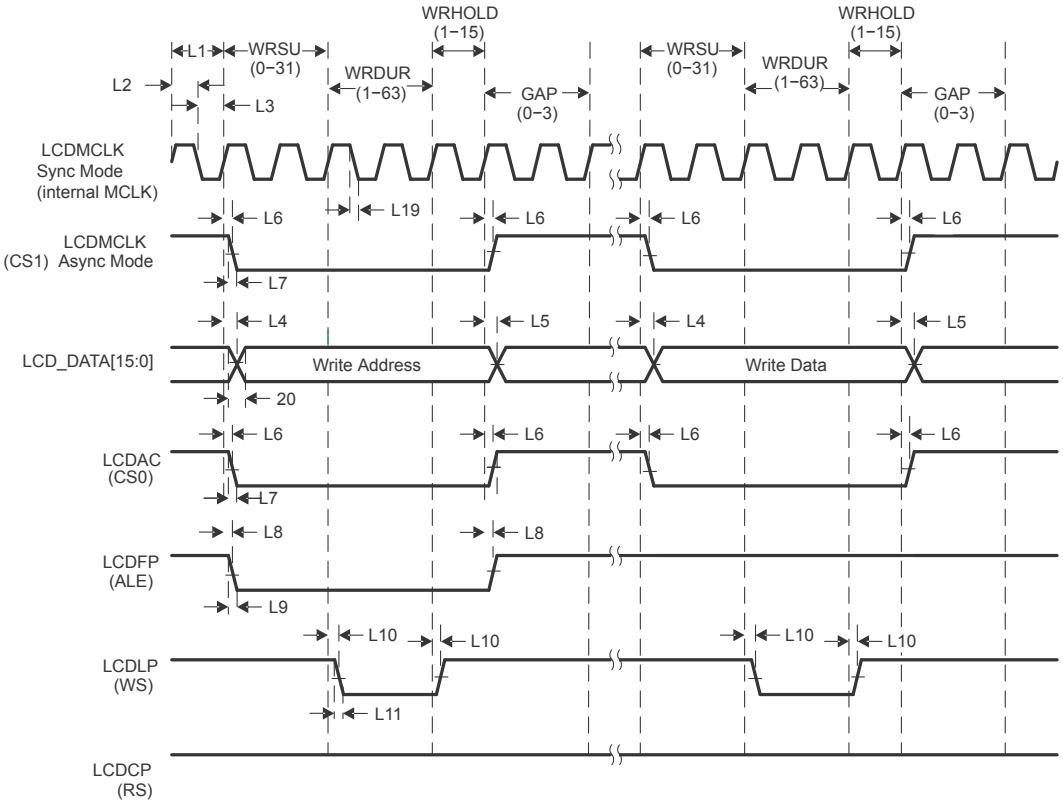


**Figure 32-58. Micro-Interface Graphic Display Intel Read**

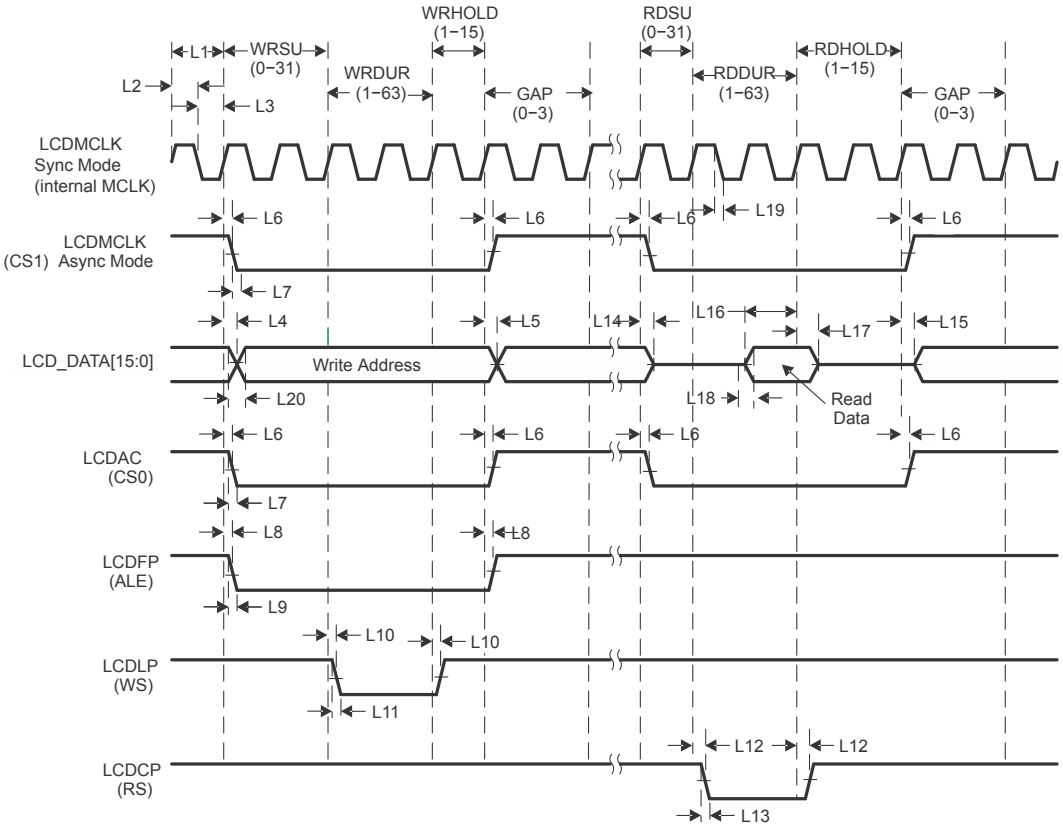
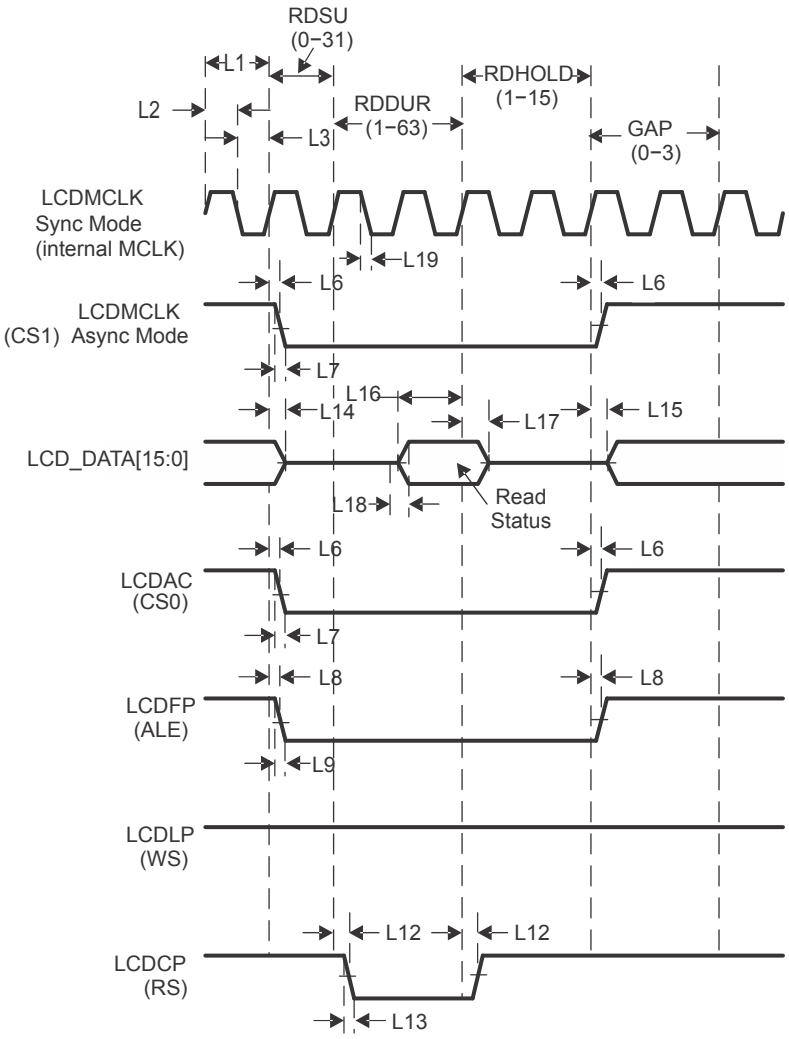


**Figure 32-59. Micro-Interface Graphic Display Intel Status**




## 32.21.2 LCD Raster Mode

The following table depicts the LCD Raster Mode Characteristics.

**Table 32-69. Switching Characteristics Over Recommended Operating Characteristics for LCD Raster Mode**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| L21 | $T_{CYC}$ | LCDCP cycle time | 16.67 | - | - | ns |
| L22 | $T_{PH}$ | LCDCP pulse width high | 8.33 | - | - | ns |
| L23 | $T_{PL}$ | LCDCP pulse width low | 8.33 | - | - | ns |
| L24 | $T_{DLYVAL}$ | Delay time from LCDCP to LCDDATA[23:0] valid (write) | 2.0 | - | 7.3 | ns |

**Table 32-69. Switching Characteristics Over Recommended Operating Characteristics for LCD Raster Mode *(continued)***

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| L25 | $T_{DLYINV}$ | Delay time from LCDCP to LCDDATA[23:0] invalid (write) | 2.0 | - | 7.3 | ns |
| L26 | $T_{DLYHAC}$ | Delay time, LCDCP to LCDAC | 1.9 | - | 7.0 | ns |
| L27 | $T_{TRANAC}$ | LCDAC transition time | 0.5 | - | 3.3 | ns |
| L28 | $T_{DLYFP}$ | Delay time from LCDCP high to LCDFP | 1.7 | - | 6.5 | ns |
| L29 | $T_{TRANFP}$ | LCDFP transition time | 0.5 | - | 3.3 | ns |
| L30 | $T_{DLYLP}$ | Delay time from LCDCP high to LCDLP | 2.0 | - | 6.8 | ns |
| L31 | $T_{TRANLP}$ | LCDLP transition time | 0.5 | - | 3.3 | ns |
| L32 | $T_{TRANCP}$ | LCDCP transition time | 0.5 | - | 3.3 | ns |
| L33 | $T_{TRANDATA}$ | LCDDATA transition time | 0.5 | - | 3.3 | ns |

Frame-to-frame timing is derived through the following parameters in the **LCD Raster Timing 1 (LCDRASTRTIM1)** register:

- Vertical front porch (VFP)
- Vertical sync pulse width (VSW)
- Vertical back porch (VBP)
- Lines per panel (MSBLPP + LPP)

Line-to-line timing is derived through the following parameters in the **LCD Raster Timing 0 (LCDRASTRTIM0)** register:

- Horizontal front porch (HFP)
- Horizontal sync pulse width (HSW)
- Horizontal back porch (HBP)
- Pixels per panel (PPLMSB + PPLLSB)

LCDAC timing is derived through the following parameter in the **LCD Raster Timing 2 (LCDRASTRTIM2)** register:

- AC bias frequency (ACBF)

The display format produced in raster mode is shown in Figure 32-60 on page 2170. An entire frame is delivered one line at a time. The first line delivered starts at data pixel (1, 1) and ends at data pixel (P, 1). The last line delivered starts at data pixel (1, L) and ends at data pixel (P, L). The beginning of each new frame is denoted by the activation of IO signal LCDFP (VSYNC). The beginning of each new line is denoted by the activation of IO signal LCDLP (HSYNC).

**Figure 32-60. LCD Raster-Mode Display Format**

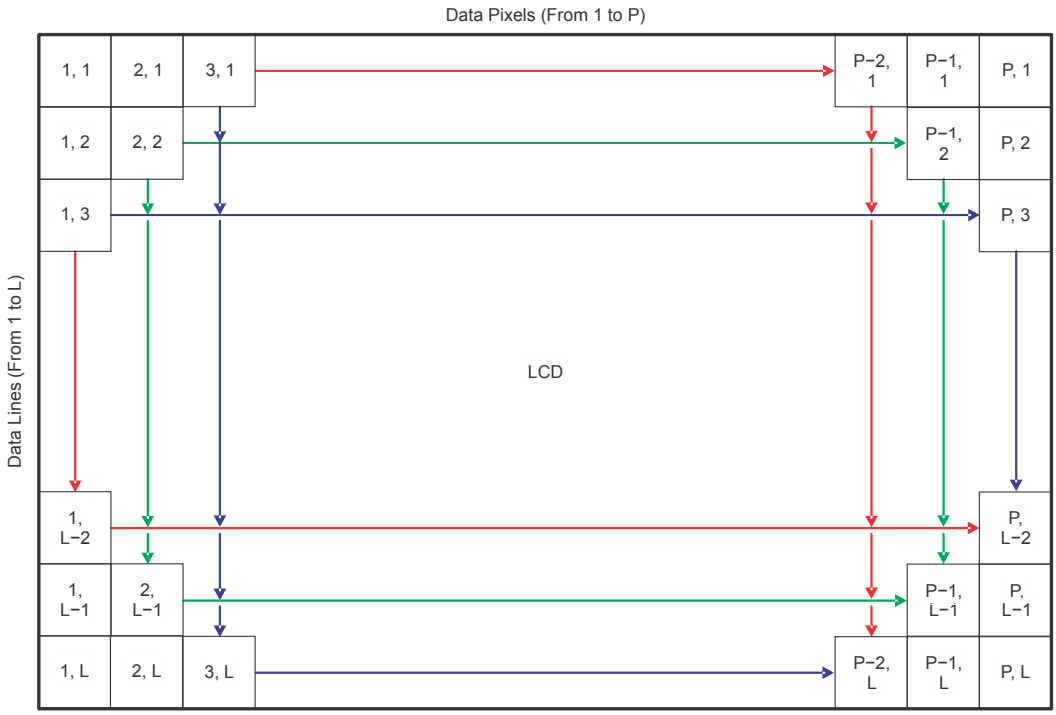


**Figure 32-61. LCD Raster-Mode Active**

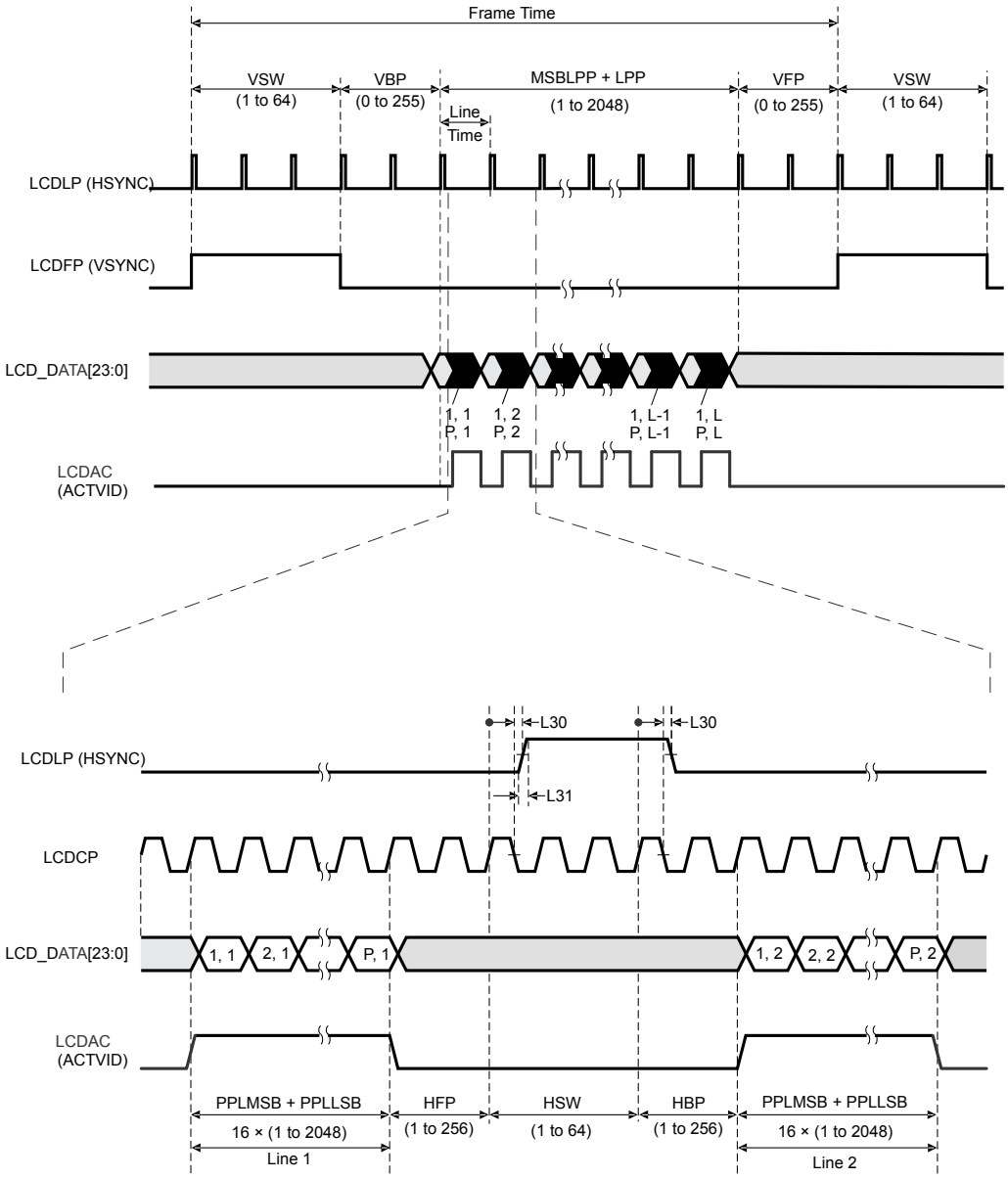


**Figure 32-62. LCD Raster-Mode Passive**

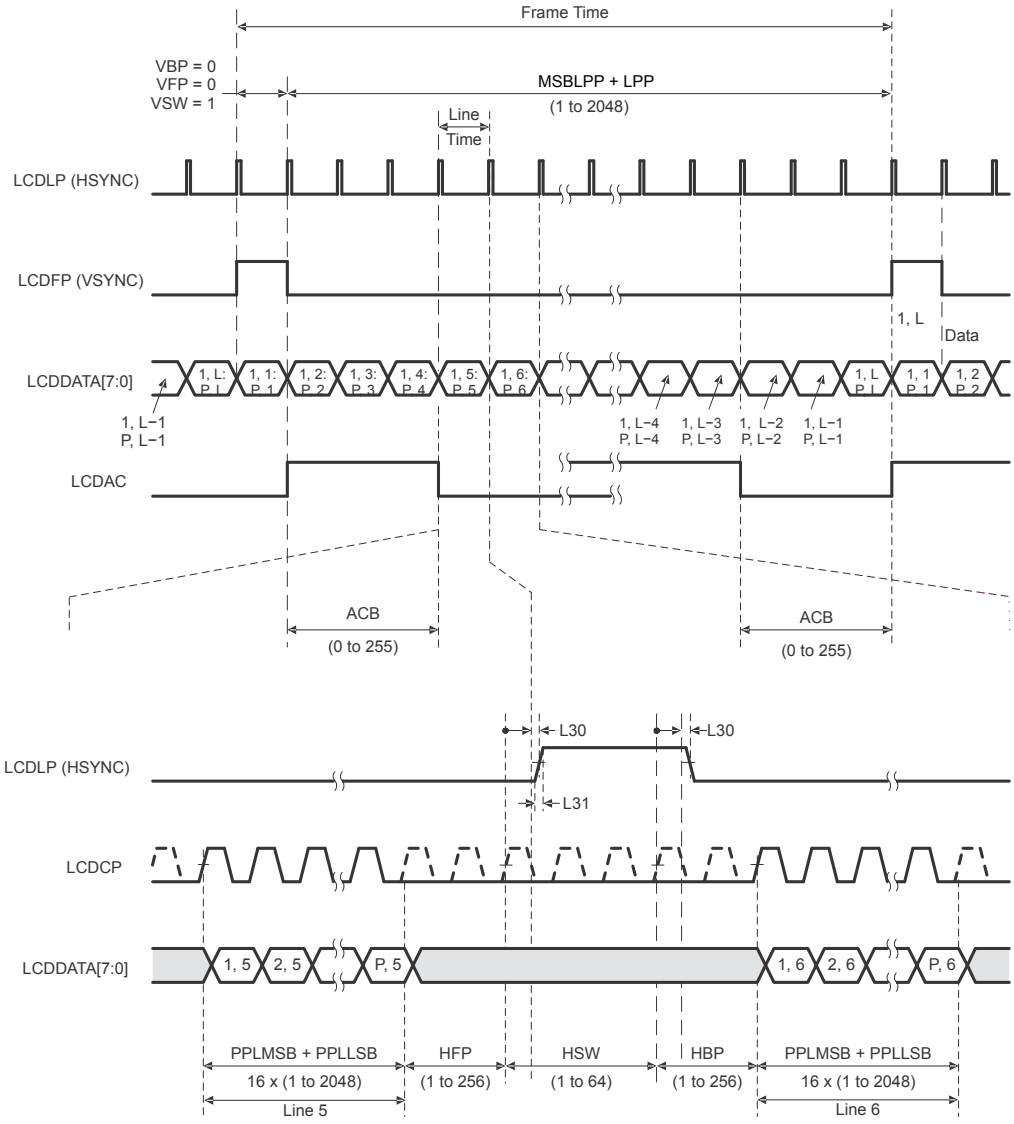


**Figure 32-63. LCD Raster-Mode Control Signal Activation**

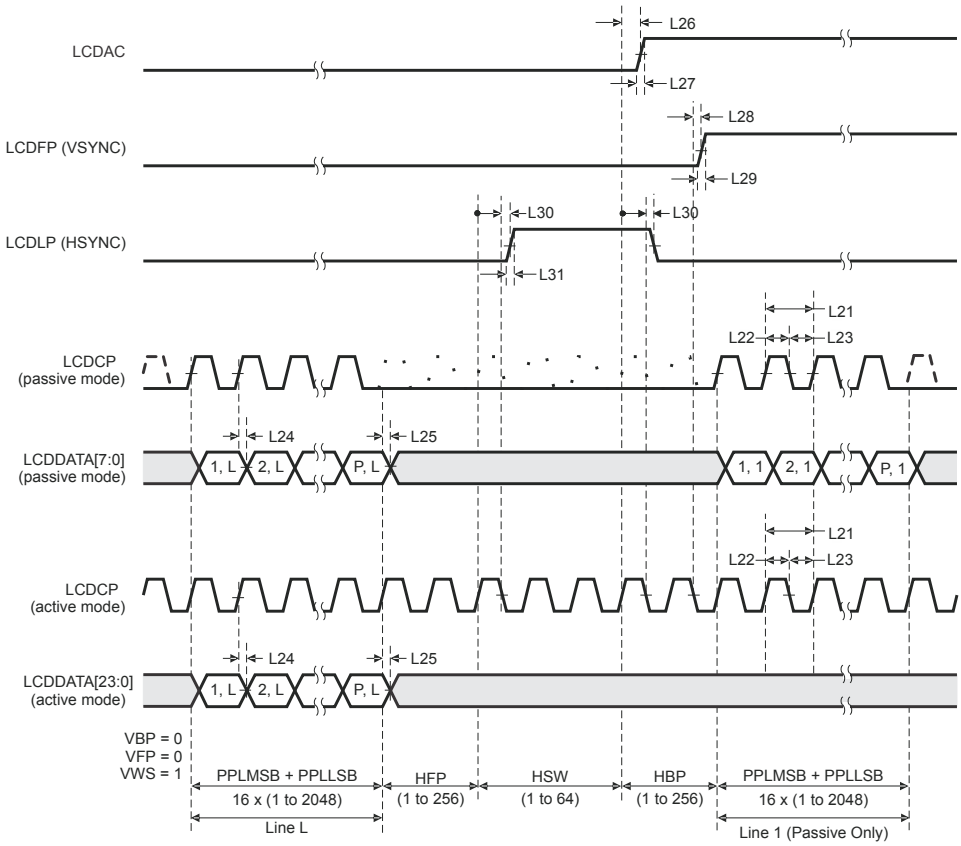
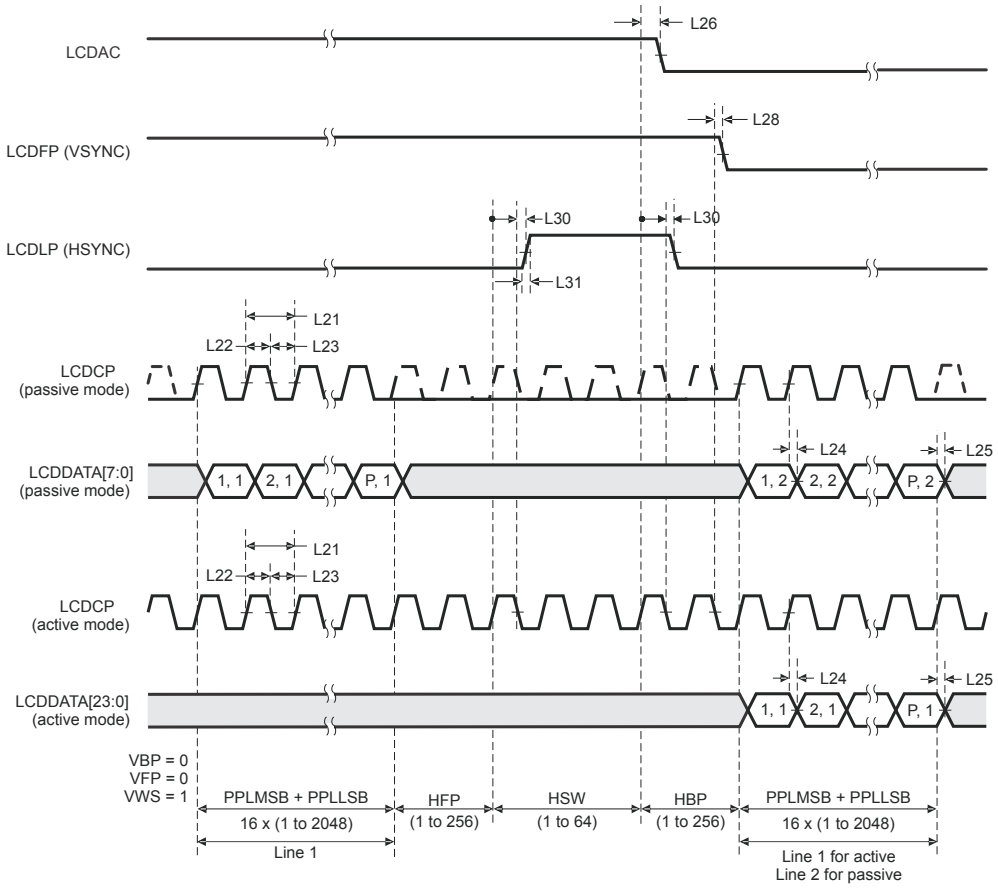


**Figure 32-64. LCD Raster-Mode Control Signal Deactivation**



## 32.22 Analog Comparator

### Table 32-70. Analog Comparator Characteristics[ab]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{INP}$,$V_{INN}$[c] | Input voltage range | GNDA | - | $V_{DDA}$ | V |
| $V_{CM}$ | Input common mode voltage range | GNDA | - | $V_{DDA}$ | V |
| $V_{OS}$ | Input offset voltage | - | ±10 | ±50[d] | mV |
| $I_{INP}$,$I_{INN}$ | Input leakage current over full voltage range | - | - | 2.0 | µA |
| $C_{MRR}$ | Common mode rejection ratio | - | 50 | - | dB |
| $T_{RT}$ | Response time | - | - | 1.0[e] | µs |
| $T_{MC}$ | Comparator mode change to Output Valid | - | - | 10 | µs |

a. Best design practices suggest that static or quiet digital I/O signals be configured adjacent to sensitive analog inputs to reduce capacitive coupling and cross talk.

b. To achieve best analog results, the source resistance driving the analog inputs, $V_{INP}$ and $V_{INN}$, should be kept low.

c. The external voltage inputs to the Analog Comparator are designed to be highly sensitive and can be affected by external noise on the board. For this reason, $V_{INP}$ and $V_{INN}$ must be set to different voltage levels during idle states to ensure the analog comparator triggers are not enabled. If an internal voltage reference is used, it should be set to a mid-supply level. When operating in Sleep/Deep-Sleep modes, the Analog Comparator module should be disabled or the external voltage inputs set to different levels (greater than the input offset voltage) to achieve minimum current draw.

d. Measured at VREF=100 mV.

e. Measured at external VREF=100 mV, input signal switching from 75 mV to 125 mV.

### Table 32-71. Analog Comparator Voltage Reference Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $R_{HR}$ | Resolution in high range | - | $V_{DDA}$/29.4 | - | V |
| $R_{LR}$ | Resolution in low range | - | $V_{DDA}$/22.12 | - | V |
| $A_{HR}$ | Absolute accuracy high range | - | - | ±$R_{HR}$/2 | V |
| $A_{LR}$ | Absolute accuracy low range | - | - | ±$R_{LR}$/2 | V |

### Table 32-72. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 0

| VREF Value | $V_{IREF}$ Min | Ideal $V_{IREF}$ | $V_{IREF}$ Max | Unit |
|---|---|---|---|---|
| 0x0 | 0.731 | 0.786 | 0.841 | V |
| 0x1 | 0.843 | 0.898 | 0.953 | V |
| 0x2 | 0.955 | 1.010 | 1.065 | V |
| 0x3 | 1.067 | 1.122 | 1.178 | V |
| 0x4 | 1.180 | 1.235 | 1.290 | V |
| 0x5 | 1.292 | 1.347 | 1.402 | V |
| 0x6 | 1.404 | 1.459 | 1.514 | V |
| 0x7 | 1.516 | 1.571 | 1.627 | V |
| 0x8 | 1.629 | 1.684 | 1.739 | V |
| 0x9 | 1.741 | 1.796 | 1.851 | V |
| 0xA | 1.853 | 1.908 | 1.963 | V |
| 0xB | 1.965 | 2.020 | 2.076 | V |
| 0xC | 2.078 | 2.133 | 2.188 | V |

**Table 32-72. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 0 *(continued)***

| VREF Value | $V_{IREF}$ Min | Ideal $V_{IREF}$ | $V_{IREF}$ Max | Unit |
|---|---|---|---|---|
| 0xD | 2.190 | 2.245 | 2.300 | V |
| 0xE | 2.302 | 2.357 | 2.412 | V |
| 0xF | 2.414 | 2.469 | 2.525 | V |

**Table 32-73. Analog Comparator Voltage Reference Characteristics, $V_{DDA}$ = 3.3V, EN= 1, and RNG = 1**

| VREF Value | $V_{IREF}$ Min | Ideal $V_{IREF}$ | $V_{IREF}$ Max | Unit |
|---|---|---|---|---|
| 0x0 | 0.000 | 0.000 | 0.074 | V |
| 0x1 | 0.076 | 0.149 | 0.223 | V |
| 0x2 | 0.225 | 0.298 | 0.372 | V |
| 0x3 | 0.374 | 0.448 | 0.521 | V |
| 0x4 | 0.523 | 0.597 | 0.670 | V |
| 0x5 | 0.672 | 0.746 | 0.820 | V |
| 0x6 | 0.822 | 0.895 | 0.969 | V |
| 0x7 | 0.971 | 1.044 | 1.118 | V |
| 0x8 | 1.120 | 1.193 | 1.267 | V |
| 0x9 | 1.269 | 1.343 | 1.416 | V |
| 0xA | 1.418 | 1.492 | 1.565 | V |
| 0xB | 1.567 | 1.641 | 1.715 | V |
| 0xC | 1.717 | 1.790 | 1.864 | V |
| 0xD | 1.866 | 1.939 | 2.013 | V |
| 0xE | 2.015 | 2.089 | 2.162 | V |
| 0xF | 2.164 | 2.238 | 2.311 | V |

## 32.23 Pulse-Width Modulator (PWM)

**Table 32-74. PWM Timing Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $T_{FLTW}$ | Minimum Fault Pulse Width | 2 | - | - | PWM clock periods |
| $T_{FLTMAX}$ | MnFAULTn Assertion to PWM Inactive[a] | - | - | 24 + (1 PWM clock) | ns |
| $T_{FLTMIN}$ | MnFAULTn De-Assertion to PWM Active[b] | 5 | - | - | ns |

a. This parameter value can vary depending on the PWM clock frequency which is controlled by the System Clock and a programmable divider field in the **PWMCC** register.

b. The latch and minimum fault period functions that can be enabled in the **PWMnCTL** register can change the timing of this parameter.

## 32.24 Current Consumption

Table 32-76 on page 2182 shows the amount of current consumption that specific peripherals contribute to the Run mode current consumption numbers shown in Table 32-75 on page 2178. If these peripherals are not powered, then the peripheral current consumption values can be subtracted from the Run mode numbers displayed in Table 32-75 on page 2178.

**Table 32-75. Current Consumption**[ab]

| Parameter | Parameter Name | Conditions | System Clock | | Nom | | | | Max | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Frequency | Clock Source | -40°C | 25°C | 85°C | 105°C[c] | 85°C | 105°C[c] | |
| $I_{DD_RUN}$ | Run mode (Flash loop) | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON including MAC and PHY | 120 MHz | MOSC with PLL | 96.4 | 105.3 | 107.2 | 108.7 | 129.9 | 140.0 | mA |
| | | | 60 MHz | MOSC with PLL | 67.4 | 76.6 | 78.6 | 79.9 | 100.3 | 112.5 | mA |
| | | | 16 MHz | PIOSC | 11.9 | 24.4 | 25.5 | 26.7 | 45.0 | 56.4 | mA |
| | | | 1 MHz | PIOSC | 5.75 | 10.9 | 12.1 | 13.3 | 31.3 | 42.6 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON including MAC but not PHY | 120 MHz | MOSC with PLL | 69.9 | 77.8 | 79.6 | 80.8 | 98.8 | 108.4 | mA |
| | | | 60 MHz | MOSC with PLL | 40.9 | 49.2 | 50.9 | 52.1 | 69.2 | 80.8 | mA |
| | | | 16 MHz | PIOSC | 11.3 | 23.6 | 25.0 | 26.2 | 43.1 | 54.3 | mA |
| | | | 1 MHz | PIOSC | 5.10 | 10.1 | 11.5 | 12.7 | 29.3 | 40.5 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON except MAC/PHY | 120 MHz | MOSC with PLL | 68.1 | 76.0 | 77.6 | 78.6 | 96.6 | 106.0 | mA |
| | | | 60 MHz | MOSC with PLL | 40.0 | 48.2 | 49.8 | 50.8 | 67.9 | 79.2 | mA |
| | | | 16 MHz | PIOSC | 11.1 | 23.3 | 24.6 | 25.6 | 42.5 | 53.3 | mA |
| | | | 1 MHz | PIOSC | 5.07 | 10.1 | 11.3 | 12.3 | 29.0 | 39.8 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF | 120 MHz | MOSC with PLL | 35.2 | 39.1 | 40.4 | 41.5 | 55.8 | 65.3 | mA |
| | | | 60 MHz | MOSC with PLL | 23.2 | 29.4 | 30.7 | 31.7 | 45.8 | 55.5 | mA |
| | | | 16 MHz | PIOSC | 7.38 | 17.9 | 19.0 | 20.0 | 34.5 | 44.1 | mA |
| | | | 1 MHz | PIOSC | 4.12 | 9.13 | 10.3 | 11.4 | 25.7 | 35.5 | mA |
| | Run mode (SRAM loop) | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON including MAC and PHY | 120 MHz | MOSC with PLL | 93.8 | 103.6 | 111.6 | 113.2 | 133.4 | 144.6 | mA |
| | | | 60 MHz | MOSC with PLL | 66.9 | 76.7 | 78.7 | 80.0 | 100.0 | 111.9 | mA |
| | | | 16 MHz | PIOSC | 12.6 | 19.0 | 20.1 | 21.3 | 39.1 | 50.3 | mA |
| | | | 1 MHz | PIOSC | 5.73 | 10.6 | 11.7 | 12.8 | 30.9 | 42.2 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON including MAC but not PHY | 120 MHz | MOSC with PLL | 67.2 | 76.1 | 84.0 | 85.4 | 102.3 | 113.0 | mA |
| | | | 60 MHz | MOSC with PLL | 40.3 | 49.2 | 50.9 | 52.2 | 68.9 | 80.2 | mA |
| | | | 16 MHz | PIOSC | 11.9 | 18.2 | 19.6 | 20.8 | 37.2 | 48.2 | mA |
| | | | 1 MHz | PIOSC | 5.08 | 9.79 | 11.2 | 12.3 | 28.9 | 40.1 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V | 120 MHz | MOSC with PLL | 65.4 | 74.3 | 82.0 | 83.2 | 100.1 | 110.6 | mA |

**Table 32-75. Current Consumption *(continued)***

| Parameter | Parameter Name | Conditions | System Clock | | Nom | | | | Max | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Frequency | Clock Source | -40°C | 25°C | 85°C | 105°C[c] | 85°C | 105°C[c] | |
| | | Peripherals =All ON except MAC/PHY | 60 MHz | MOSC with PLL | 39.4 | 48.2 | 49.8 | 50.9 | 67.6 | 78.6 | mA |
| | | | 16 MHz | PIOSC | 11.7 | 17.9 | 19.2 | 20.2 | 36.6 | 47.2 | mA |
| | | | 1 MHz | PIOSC | 5.05 | 9.75 | 11.0 | 11.9 | 28.6 | 39.4 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF | 120 MHz | MOSC with PLL | 35.4 | 43.3 | 44.7 | 45.8 | 59.8 | 69.0 | mA |
| | | | 60 MHz | MOSC with PLL | 23.4 | 29.4 | 30.7 | 31.7 | 45.5 | 54.9 | mA |
| | | | 16 MHz | PIOSC | 7.08 | 12.4 | 13.6 | 14.6 | 28.7 | 38.0 | mA |
| | | | 1 MHz | PIOSC | 4.60 | 8.78 | 10.0 | 11.0 | 25.3 | 34.9 | mA |
| $I_{DD_SLEEP}$ | Sleep mode (FLASHPM = 0x0) | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON including MAC and PHY<br>LDO = 1.2 V | 120 MHz | MOSC with PLL | 82.8 | 94.8 | 96.8 | 98.1 | 117.9 | 129.1 | mA |
| | | | 60 MHz | MOSC with PLL | 60.8 | 69.2 | 71.2 | 72.3 | 91.8 | 102.9 | mA |
| | | | 16 MHz | PIOSC[d] | 11.2 | 16.8 | 18.1 | 19.1 | 35.4 | 45.9 | mA |
| | | | 1 MHz | PIOSC[d] | 5.10 | 10.3 | 11.5 | 12.6 | 28.9 | 39.6 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON including MAC but not PHY<br>LDO = 1.2 V | 120 MHz | MOSC with PLL | 56.2 | 67.4 | 69.1 | 70.3 | 87.1 | 97.8 | mA |
| | | | 60 MHz | MOSC with PLL | 34.4 | 41.9 | 43.4 | 44.5 | 60.7 | 71.6 | mA |
| | | | 16 MHz | PIOSC[d] | 10.6 | 16.2 | 17.5 | 18.5 | 34.5 | 45.1 | mA |
| | | | 1 MHz | PIOSC[d] | 4.47 | 9.60 | 10.9 | 12.0 | 28.0 | 38.7 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON except MAC/PHY<br>LDO = 1.2 V | 120 MHz | MOSC with PLL | 54.4 | 65.6 | 67.1 | 68.1 | 84.9 | 95.4 | mA |
| | | | 60 MHz | MOSC with PLL | 33.5 | 40.9 | 42.3 | 43.2 | 59.4 | 70.0 | mA |
| | | | 16 MHz | PIOSC[d] | 10.4 | 15.9 | 17.1 | 17.9 | 33.9 | 44.1 | mA |
| | | | 1 MHz | PIOSC[d] | 4.44 | 9.56 | 10.7 | 11.6 | 27.7 | 38.0 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF<br>LDO = 1.2 V | 120 MHz | MOSC with PLL | 22.0 | 28.6 | 29.8 | 30.7 | 44.1 | 53.1 | mA |
| | | | 60 MHz | MOSC with PLL | 16.3 | 22.0 | 23.2 | 24.1 | 37.5 | 46.6 | mA |
| | | | 16 MHz | PIOSC[d] | 5.37 | 10.4 | 11.5 | 12.4 | 26.1 | 35.1 | mA |
| | | | 1 MHz | PIOSC[d] | 4.37 | 8.60 | 9.71 | 10.6 | 24.6 | 33.9 | mA |
| | Sleep mode (FLASHPM = 0x2) | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON including MAC and PHY<br>LDO = 1.2 V | 120 MHz | MOSC with PLL | 86.5 | 89.0 | 91.2 | 92.5 | 112.1 | 123.5 | mA |
| | | | 60 MHz | MOSC with PLL | 61.6 | 63.4 | 65.6 | 66.7 | 86.0 | 97.2 | mA |
| | | | 16 MHz | PIOSC[d] | 10.4 | 11.1 | 12.4 | 13.5 | 29.8 | 40.4 | mA |
| | | | 1 MHz | PIOSC[d] | 4.45 | 4.49 | 5.83 | 6.98 | 23.4 | 34.2 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V | 120 MHz | MOSC with PLL | 59.9 | 61.7 | 63.4 | 64.7 | 81.3 | 92.1 | mA |
| | | | 60 MHz | MOSC with PLL | 35.1 | 36.1 | 37.8 | 38.9 | 54.9 | 66.0 | mA |

**Table 32-75. Current Consumption *(continued)***

| Parameter | Parameter Name | Conditions | System Clock | | Nom | | | | Max | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Frequency | Clock Source | -40°C | 25°C | 85°C | 105°C[c] | 85°C | 105°C[c] | |
| | | Peripherals = All ON including MAC but not PHY<br>LDO = 1.2 V | 16 MHz | PIOSC[d] | 9.75 | 10.4 | 11.8 | 12.9 | 28.9 | 39.6 | mA |
| | | | 1 MHz | PIOSC[d] | 3.82 | 3.82 | 5.25 | 6.38 | 22.5 | 33.4 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON except MAC/PHY<br>LDO = 1.2 V | 120 MHz | MOSC with PLL | 58.1 | 59.9 | 61.4 | 62.5 | 79.1 | 89.7 | mA |
| | | | 60 MHz | MOSC with PLL | 34.2 | 35.1 | 36.7 | 37.6 | 53.6 | 64.4 | mA |
| | | | 16 MHz | PIOSC[d] | 9.50 | 10.1 | 11.4 | 12.3 | 28.3 | 38.6 | mA |
| | | | 1 MHz | PIOSC[d] | 3.79 | 3.78 | 5.06 | 5.96 | 22.2 | 32.7 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF<br>LDO = 1.2 V | 120 MHz | MOSC with PLL | 22.0 | 22.8 | 24.1 | 25.1 | 38.2 | 47.4 | mA |
| | | | 60 MHz | MOSC with PLL | 15.7 | 16.2 | 17.5 | 18.5 | 31.7 | 40.9 | mA |
| | | | 16 MHz | PIOSC[d] | 4.50 | 4.60 | 5.80 | 6.80 | 20.5 | 29.8 | mA |
| | | | 1 MHz | PIOSC[d] | 3.00 | 2.80 | 4.10 | 5.20 | 19.1 | 28.7 | mA |
| $I_{DD_DEEPSLEEP}$[e] | Deep-Sleep mode (FLASHPM = 0x2) | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON<br>LDO = 1.2 V | 16 MHz | PIOSC | 9.74 | 9.78 | 10.8 | 11.6 | 24.1 | 32.1 | mA |
| | | | 30 kHz | LFIOSC | 2.60 | 2.83 | 3.83 | 4.60 | 17.1 | 25.3 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF<br>LDO = 1.2 V | 16 MHz | PIOSC | 4.53 | 4.05 | 4.88 | 5.53 | 15.9 | 22.7 | mA |
| | | | 30 kHz | LFIOSC | 0.614 | 0.762 | 1.69 | 2.46 | 13.3 | 20.7 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON<br>LDO = 0.9 V[f] | 16 MHz | PIOSC | 5.21 | 7.33 | 7.97 | 8.48 | 15.3 | 20.1 | mA |
| | | | 30 kHz | LFIOSC | 2.02 | 2.16 | 2.79 | 3.29 | 10.0 | 14.9 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF<br>LDO = 0.9 V[f] | 16 MHz | PIOSC | 1.08 | 3.10 | 3.61 | 4.01 | 9.50 | 13.4 | mA |
| | | | 30 kHz | LFIOSC | 0.367 | 0.454 | 0.954 | 1.36 | 6.86 | 10.8 | mA |
| $I_{DDA_RUN}$, $I_{DDA_SLEEP}$ | All Run modes<br>All Sleep modes | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON | 120 MHz | MOSC with PLL | 2.61 | 2.66 | 2.68 | 2.66 | 3.03 | 3.35 | mA |
| | | | 60 MHz | MOSC with PLL | 2.61 | 2.66 | 2.68 | 2.66 | 3.04 | 3.10 | mA |
| | | | 16 MHz | PIOSC | 2.45 | 2.49 | 2.50 | 2.48 | 2.85 | 2.95 | mA |
| | | | 1 MHz | PIOSC | 2.45 | 2.48 | 2.50 | 2.48 | 2.84 | 2.90 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF | 120 MHz | MOSC with PLL | 0.227 | 0.229 | 0.270 | 0.250 | 0.559 | 0.650 | mA |
| | | | 60 MHz | MOSC with PLL | 0.229 | 0.232 | 0.267 | 0.250 | 0.579 | 0.600 | mA |
| | | | 16 MHz | PIOSC | 0.228 | 0.229 | 0.265 | 0.251 | 0.545 | 0.575 | mA |
| | | | 1 MHz | PIOSC | 0.227 | 0.227 | 0.267 | 0.247 | 0.549 | 0.555 | mA |

**Table 32-75. Current Consumption *(continued)***

| Parameter | Parameter Name | Conditions | System Clock | | Nom | | | | Max | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Frequency | Clock Source | -40°C | 25°C | 85°C | 105°C[c] | 85°C | 105°C[c] | |
| $I_{DDA_DEEPSLEEP}$ | Deep-Sleep mode (FLASHPM = 0x2) | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON<br>LDO = 1.2 V | 16 MHz | PIOSC | 2.45 | 2.48 | 2.50 | 2.48 | 2.84 | 2.90 | mA |
| | | | 30 kHz | LFIOSC | 2.45 | 2.48 | 2.50 | 2.48 | 2.85 | 2.90 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF<br>LDO = 1.2 V | 16 MHz | PIOSC | 0.226 | 0.227 | 0.265 | 0.249 | 0.558 | 0.635 | mA |
| | | | 30 kHz | LFIOSC | 0.228 | 0.227 | 0.272 | 0.247 | 0.558 | 0.600 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All ON<br>LDO = 0.9 V[f] | 16 MHz | PIOSC | 2.14 | 2.42 | 2.44 | 2.42 | 2.78 | 2.88 | mA |
| | | | 30 kHz | LFIOSC | 2.44 | 2.42 | 2.44 | 2.42 | 2.86 | 2.88 | mA |
| | | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>Peripherals = All OFF<br>LDO = 0.9 V[f] | 16 MHz | PIOSC | 0.216 | 0.166 | 0.209 | 0.193 | 0.563 | 0.580 | mA |
| | | | 30 kHz | LFIOSC | 0.223 | 0.167 | 0.209 | 0.189 | 0.508 | 0.580 | mA |
| $I_{HIB_NORTC}$ | Hibernate mode (external wake, RTC disabled) | $V_{BAT}$ = 3.0 V<br>$V_{DD}$ = 0 V<br>$V_{DDA}$ = 0 V<br>System Clock = OFF<br>Hibernate Module = 32.768 kHz | - | - | 1.04 | 1.20 | 1.44 | 1.69 | 1.62 | 2.14 | µA |
| $I_{HIB_RTC}$ | Hibernate mode (RTC enabled) | $V_{BAT}$ = 3.0 V<br>$V_{DD}$ = 0 V<br>$V_{DDA}$ = 0 V<br>System Clock = OFF<br>Hibernate Module = 32.768 kHz | - | - | 1.12 | 1.29 | 1.54 | 1.82 | 1.75 | 2.33 | µA |
| $I_{HIB_VDD3ON}$ | Hibernate mode (VDD3ON mode, Tamper enabled) | $V_{BAT}$ = 3.0 V<br>$V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>System Clock = OFF<br>Hibernate Module = 32.768 kHz | - | - | 6.78 | 7.99 | 17.0 | 22.1 | 31.0 | 46.2 | µA |
| | Hibernate mode (VDD3ON mode, Tamper disabled) | $V_{BAT}$ = 3.0 V<br>$V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V<br>System Clock = OFF<br>Hibernate Module = 32.768 kHz | - | - | 5.42 | 6.39 | 15.4 | 17.8 | 28.9 | 32.0 | µA |

a. Total current in RUN, SLEEP and DEEPSLEEP modes is the sum of $I_{DDPHY}$, $I_{DD}$and $I_{DDA}$.

b. For Peripherals = All OFF, the clocks to all peripherals are turned off and the peripherals are powered down, if capable (see the section called “Peripheral Power Control” on page 254).

c. Applicable to extended temperature devices only.

d. Note that if the MOSC is the source of the Run-mode system clock and is powered down in Sleep mode, wake time is increased by $T_{MOSC_SETTLE}$.

e. To achieve the lowest possible Deep-Sleep current, one or more wait states must be configured in the **MEMTIM0** register. If there are no wait states applied in Run mode, then lowest possible Deep-Sleep current is not achieved.

f. See the section called “LDO Power Control” on page 255 for information on lowering the LDO voltage to 0.9 V.

### Table 32-76. Peripheral Current Consumption

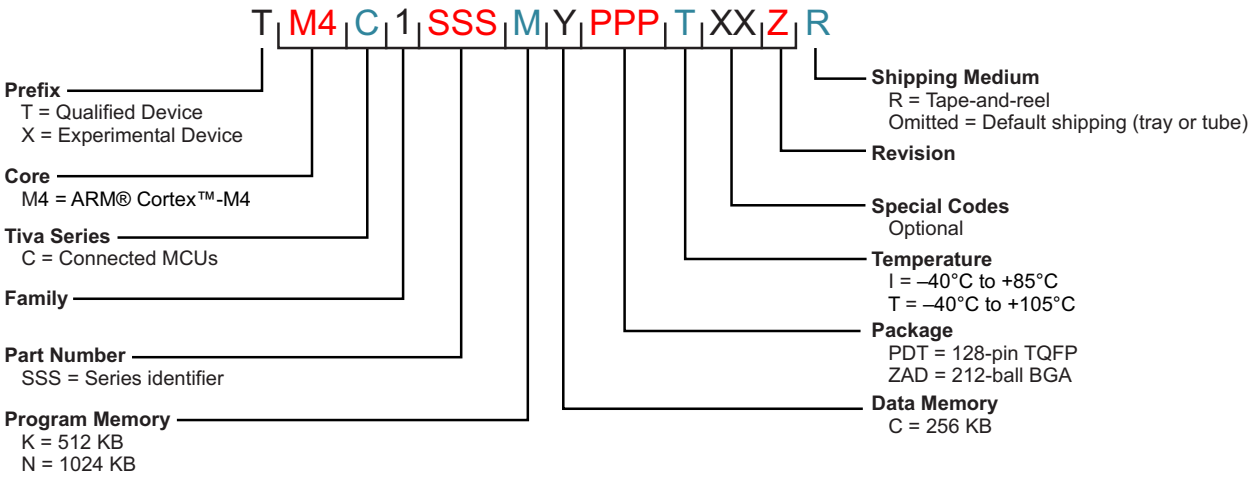
| Parameter | Parameter Name | Conditions | System Clock | Nom | Units |
|---|---|---|---|---|---|
| $I_{DDUSB}$ | USB (including USB PHY) run mode current | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V | 120 MHz (MOSC with PLL) | 4.0 | mA |
| $I_{DDEMAC}$ | Ethernet MAC run mode current | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V | 120 MHz (MOSC with PLL) | 1.9 | mA |
| $I_{DDEMACPHY}$ | Ethernet MAC and PHY run mode current | $V_{DD}$ = 3.3 V<br>$V_{DDA}$ = 3.3 V | 120 MHz (MOSC with PLL) | 30 | mA |

# A Package Information

## A.1 Orderable Devices

The figure below defines the full set of orderable part numbers for the TM4C129x Series. See the Package Option Addendum for the complete list of valid orderable part numbers for the TM4C129XKCZAD microcontroller.

**Figure A-1. Key to Part Numbers**




## A.2 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all microcontroller (MCU) devices. Each Tiva™ C Series family member has one of two prefixes: XM4C or TM4C. These prefixes represent evolutionary stages of product development from engineering prototypes (XM4C) through fully qualified production devices (TM4C).

Device development evolutionary flow:

- XM4C — Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.
- TM4C — Production version of the silicon die that is fully qualified.

XM4C devices are shipped against the following disclaimer:

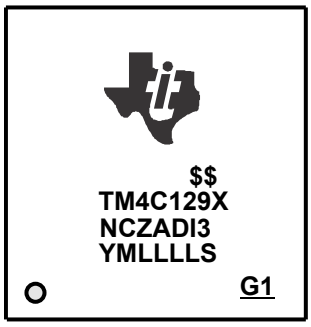
"Developmental product is intended for internal evaluation purposes."

TM4C devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (XM4C) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

## A.3 Device Markings

The figure below shows an example of the Tiva™ microcontroller package symbolization.




This identifying number contains the following information:

- **Lines 1 and 5:** Internal tracking numbers
- **Lines 2 and 3:** Part number

  For example, TM4C129X on the second line followed by NCZADI3 on the third line indicates orderable part number TM4C129XNCZADI3. Note that the first letter in the part number indicates the product status. A T indicates the part is fully qualified and released to production; an X indicates the part is experimental (pre-production) and requires a waiver. The silicon revision number is the last number in the part number, in this example, 3. The **DID0** register identifies the version of the microcontroller, as shown in the table below. Combined, the MAJOR and MINOR bit fields indicate the die revision and part revision numbers.

| MAJOR Bitfield Value | MINOR Bitfield Value | Die Revision | Part Revision |
|---|---|---|---|
| 0x0 | 0x0 | A0 | 1 |
| 0x0 | 0x1 | A1 | 2 |
| 0x0 | 0x2 | A2 | 3 |

- **Line 4:** Date code

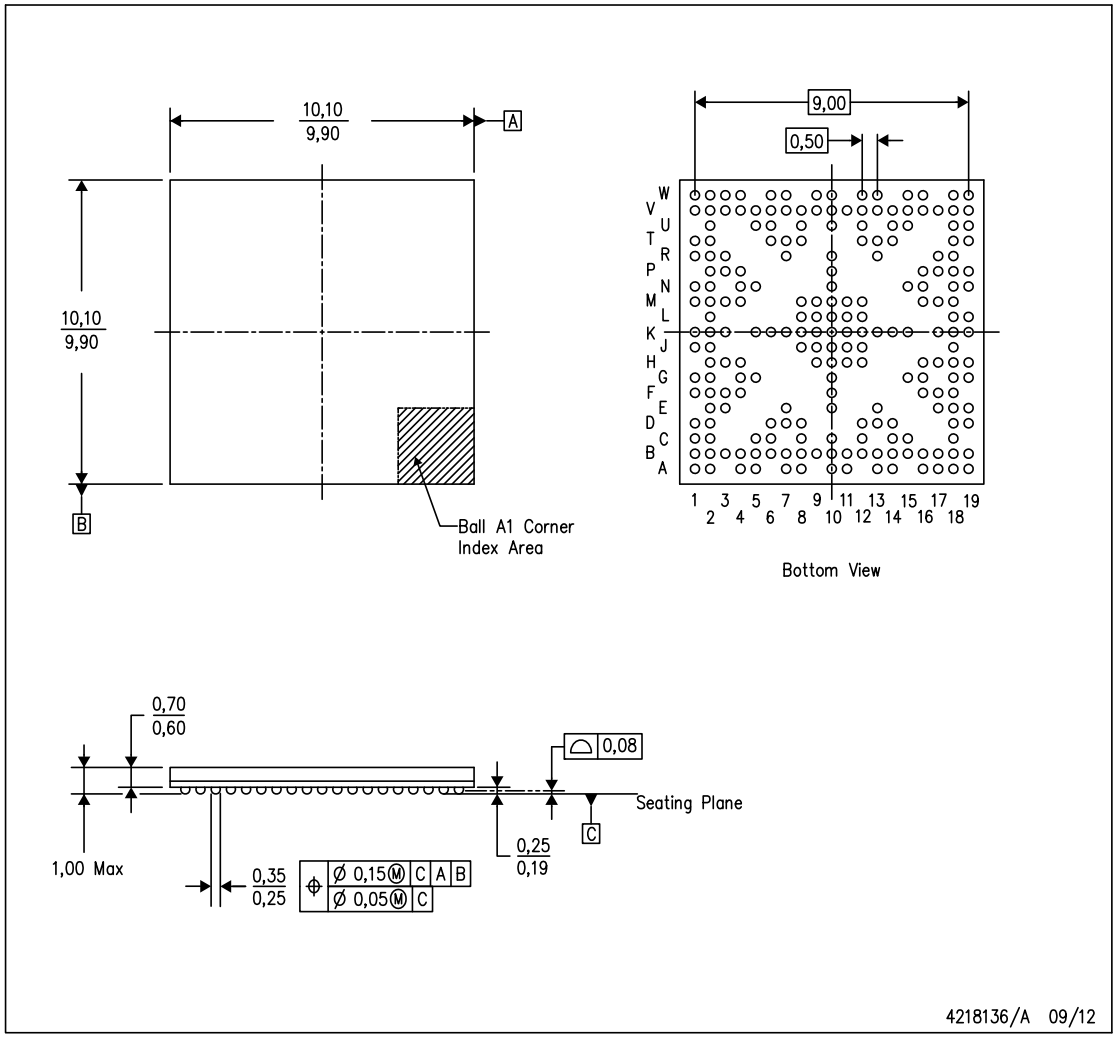
  The first two characters on the fourth line indicate the date code, followed by internal tracking numbers. The two-digit date code YM indicates the last digit of the year, then the month. For example, a 34 for the first two digits of the fourth line indicates a date code of April 2013.

# A.4 Packaging Diagram

**Figure A-2. TM4C129XKCZAD 212-Ball BGA Package Diagram**

**MECHANICAL DATA**

ZAD (S–PBGA–N212) PLASTIC BALL GRID ARRAY




NOTES:
A. All linear dimensions are in millimeters. Dimensioning and tolerancing per ASME Y14.5M–1994.
B. This drawing is subject to change without notice.
C. This is a Pb–free solder ball design.

## PACKAGING INFORMATION

| Orderable part number | Status (1) | Material type (2) | Package \| Pins | Package qty \| Carrier | RoHS (3) | Lead finish/ Ball material (4) | MSL rating/ Peak reflow (5) | Op temp (°C) | Part marking (6) |
|---|---|---|---|---|---|---|---|---|---|
| TM4C129XKCZADI3 | Active | Production | NFBGA (ZAD) \| 212 | 184 \| JEDEC TRAY (10+1) | Yes | SNAGCU | Level-3-260C-168 HR | -40 to 85 | TM4C129 XKCZADI3 |
| TM4C129XKCZADI3.B | Active | Production | NFBGA (ZAD) \| 212 | 184 \| JEDEC TRAY (10+1) | Yes | SNAGCU | Level-3-260C-168 HR | -40 to 85 | TM4C129 XKCZADI3 |
| TM4C129XKCZADI3R | Obsolete | Production | NFBGA (ZAD) \| 212 | - | - | Call TI | Call TI | -40 to 85 | TM4C129 XKCZADI3 |
| TM4C129XKCZADT3 | Active | Production | NFBGA (ZAD) \| 212 | 184 \| JEDEC TRAY (10+1) | Yes | SNAGCU | Level-3-260C-168 HR | -40 to 105 | TM4C129 XKCZADT3 |
| TM4C129XKCZADT3.B | Active | Production | NFBGA (ZAD) \| 212 | 184 \| JEDEC TRAY (10+1) | Yes | SNAGCU | Level-3-260C-168 HR | -40 to 105 | TM4C129 XKCZADT3 |

(1) **Status:** For more details on status, see our product life cycle.

(2) **Material type:** When designated, preproduction parts are prototypes/experimental devices, and are not yet approved or released for full production. Testing and final process, including without limitation quality assurance, reliability performance testing, and/or process qualification, may not yet be complete, and this item is subject to further changes or possible discontinuation. If available for ordering, purchases will be subject to an additional waiver at checkout, and are intended for early internal evaluation purposes only. These items are sold without warranties of any kind.

(3) **RoHS values:** Yes, No, RoHS Exempt. See the TI RoHS Statement for additional information and value definition.

(4) **Lead finish/Ball material:** Parts may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead finish/Ball material values may wrap to two lines if the finish value exceeds the maximum column width.

(5) **MSL rating/Peak reflow:** The moisture sensitivity level ratings and peak solder (reflow) temperatures. In the event that a part has multiple moisture sensitivity ratings, only the lowest level per JEDEC standards is shown. Refer to the shipping label for the actual reflow temperature that will be used to mount the part to the printed circuit board.

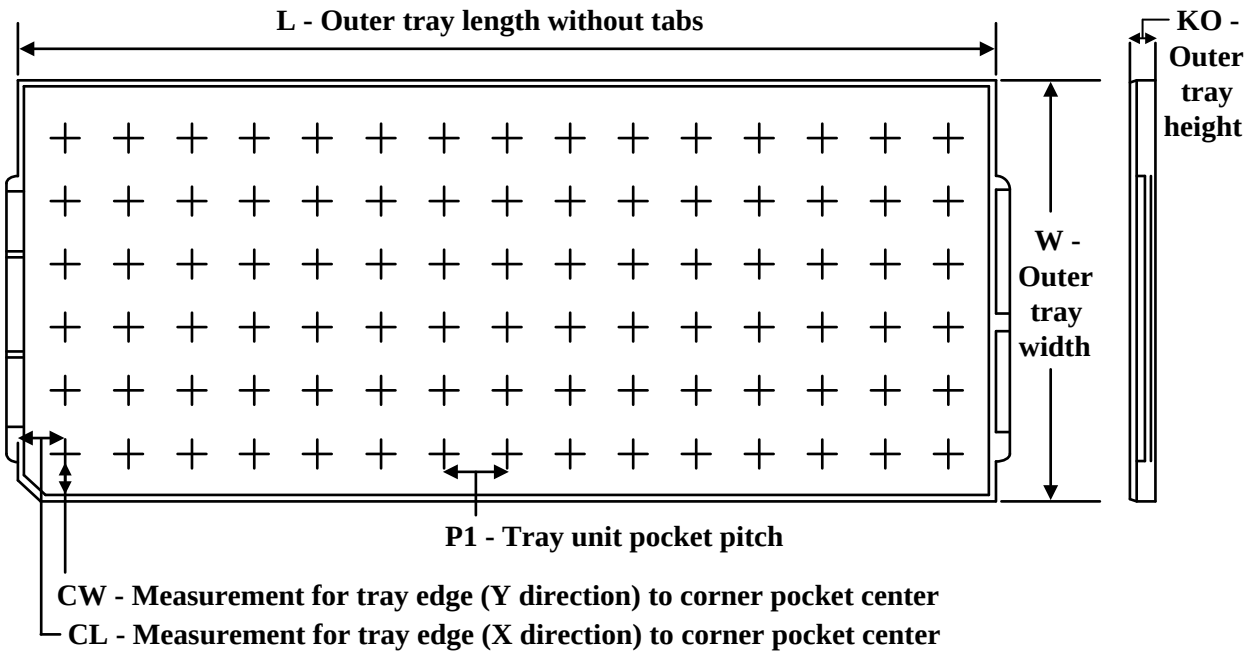
(6) **Part marking:** There may be an additional marking, which relates to the logo, the lot trace code information, or the environmental category of the part.

Multiple part markings will be inside parentheses. Only one part marking contained in parentheses and separated by a "~" will appear on a part. If a line is indented then it is a continuation of the previous line and the two combined represent the entire part marking for that device.

**Important Information and Disclaimer:**The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

## TRAY




Chamfer on Tray corner indicates Pin 1 orientation of packed units.

*All dimensions are nominal

| Device | Package Name | Package Type | Pins | SPQ | Unit array matrix | Max temperature (°C) | L (mm) | W (mm) | K0 (µm) | P1 (mm) | CL (mm) | CW (mm) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TM4C129XKCZADI3 | ZAD | NFBGA | 212 | 184 | 8 X 23 | 150 | 315 | 135.9 | 7620 | 13.4 | 10.1 | 19.65 |
| TM4C129XKCZADI3.B | ZAD | NFBGA | 212 | 184 | 8 X 23 | 150 | 315 | 135.9 | 7620 | 13.4 | 10.1 | 19.65 |
| TM4C129XKCZADT3 | ZAD | NFBGA | 212 | 184 | 8 X 23 | 150 | 315 | 135.9 | 7620 | 13.4 | 10.1 | 19.65 |
| TM4C129XKCZADT3.B | ZAD | NFBGA | 212 | 184 | 8 X 23 | 150 | 315 | 135.9 | 7620 | 13.4 | 10.1 | 19.65 |

# IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025