# THE DATASHEET OF

# LM3S1G58-IBZ80-A1

www.win-source.net 0086-755-83957316

# Stellaris® LM3S1G58 Microcontroller

## DATA SHEET

DS-LM3S1G58-9970

Copyright © 2007-2011
Texas Instruments Incorporated

# Copyright

Copyright © 2007-2011 Texas Instruments Incorporated All rights reserved. Stellaris and StellarisWare® are registered trademarks of Texas Instruments Incorporated. ARM and Thumb are registered trademarks and Cortex is a trademark of ARM Limited. Other names and brands may be claimed as 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/stellaris
http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm

# Table of Contents

**Revision History** ..... **26**

**About This Document** ..... **27**

Audience ..... 27
About This Manual ..... 27
Related Documents ..... 27
Documentation Conventions ..... 28

**1 Architectural Overview** ..... **30**

1.1 Functional Overview ..... 32
1.1.1 ARM Cortex-M3 ..... 32
1.1.2 On-Chip Memory ..... 34
1.1.3 Serial Communications Peripherals ..... 35
1.1.4 System Integration ..... 38
1.1.5 Analog ..... 43
1.1.6 JTAG and ARM Serial Wire Debug ..... 44
1.1.7 Packaging and Temperature ..... 45
1.2 Target Applications ..... 45
1.3 High-Level Block Diagram ..... 45
1.4 Hardware Details ..... 47

**2 The Cortex-M3 Processor** ..... **48**

2.1 Block Diagram ..... 49
2.2 Overview ..... 50
2.2.1 System-Level Interface ..... 50
2.2.2 Integrated Configurable Debug ..... 50
2.2.3 Trace Port Interface Unit (TPIU) ..... 51
2.2.4 Cortex-M3 System Component Details ..... 51
2.3 Programming Model ..... 52
2.3.1 Processor Mode and Privilege Levels for Software Execution ..... 52
2.3.2 Stacks ..... 52
2.3.3 Register Map ..... 53
2.3.4 Register Descriptions ..... 54
2.3.5 Exceptions and Interrupts ..... 67
2.3.6 Data Types ..... 67
2.4 Memory Model ..... 67
2.4.1 Memory Regions, Types and Attributes ..... 69
2.4.2 Memory System Ordering of Memory Accesses ..... 69
2.4.3 Behavior of Memory Accesses ..... 69
2.4.4 Software Ordering of Memory Accesses ..... 70
2.4.5 Bit-Banding ..... 71
2.4.6 Data Storage ..... 74
2.4.7 Synchronization Primitives ..... 74
2.5 Exception Model ..... 75
2.5.1 Exception States ..... 76
2.5.2 Exception Types ..... 76
2.5.3 Exception Handlers ..... 79
2.5.4 Vector Table ..... 79

2.5.5 Exception Priorities ..... 80
2.5.6 Interrupt Priority Grouping ..... 81
2.5.7 Exception Entry and Return ..... 81
2.6 Fault Handling ..... 83
2.6.1 Fault Types ..... 83
2.6.2 Fault Escalation and Hard Faults ..... 84
2.6.3 Fault Status Registers and Fault Address Registers ..... 85
2.6.4 Lockup ..... 85
2.7 Power Management ..... 85
2.7.1 Entering Sleep Modes ..... 85
2.7.2 Wake Up from Sleep Mode ..... 86
2.8 Instruction Set Summary ..... 87
**3 Cortex-M3 Peripherals ..... 90**
3.1 Functional Description ..... 90
3.1.1 System Timer (SysTick) ..... 90
3.1.2 Nested Vectored Interrupt Controller (NVIC) ..... 91
3.1.3 System Control Block (SCB) ..... 93
3.1.4 Memory Protection Unit (MPU) ..... 93
3.2 Register Map ..... 98
3.3 System Timer (SysTick) Register Descriptions ..... 100
3.4 NVIC Register Descriptions ..... 104
3.5 System Control Block (SCB) Register Descriptions ..... 117
3.6 Memory Protection Unit (MPU) Register Descriptions ..... 146
**4 JTAG Interface ..... 156**
4.1 Block Diagram ..... 157
4.2 Signal Description ..... 157
4.3 Functional Description ..... 158
4.3.1 JTAG Interface Pins ..... 158
4.3.2 JTAG TAP Controller ..... 160
4.3.3 Shift Registers ..... 160
4.3.4 Operational Considerations ..... 161
4.4 Initialization and Configuration ..... 163
4.5 Register Descriptions ..... 164
4.5.1 Instruction Register (IR) ..... 164
4.5.2 Data Registers ..... 166
**5 System Control ..... 168**
5.1 Signal Description ..... 168
5.2 Functional Description ..... 169
5.2.1 Device Identification ..... 169
5.2.2 Reset Control ..... 169
5.2.3 Non-Maskable Interrupt ..... 174
5.2.4 Power Control ..... 175
5.2.5 Clock Control ..... 175
5.2.6 System Control ..... 182
5.3 Initialization and Configuration ..... 184
5.4 Register Map ..... 184
5.5 Register Descriptions ..... 186

**6 Hibernation Module ..... 263**
6.1 Block Diagram ..... 264
6.2 Signal Description ..... 264
6.3 Functional Description ..... 265
6.3.1 Register Access Timing ..... 265
6.3.2 Hibernation Clock Source ..... 266
6.3.3 Battery Management ..... 268
6.3.4 Real-Time Clock ..... 268
6.3.5 Non-Volatile Memory ..... 268
6.3.6 Power Control Using $\overline{\text{HIB}}$ ..... 269
6.3.7 Power Control Using VDD3ON Mode ..... 269
6.3.8 Initiating Hibernate ..... 269
6.3.9 Interrupts and Status ..... 269
6.4 Initialization and Configuration ..... 270
6.4.1 Initialization ..... 270
6.4.2 RTC Match Functionality (No Hibernation) ..... 271
6.4.3 RTC Match/Wake-Up from Hibernation ..... 271
6.4.4 External Wake-Up from Hibernation ..... 271
6.4.5 RTC or External Wake-Up from Hibernation ..... 272
6.4.6 Register Reset ..... 272
6.5 Register Map ..... 272
6.6 Register Descriptions ..... 273
**7 Internal Memory ..... 290**
7.1 Block Diagram ..... 290
7.2 Functional Description ..... 290
7.2.1 SRAM ..... 291
7.2.2 ROM ..... 291
7.2.3 Flash Memory ..... 293
7.3 Register Map ..... 298
7.4 Flash Memory Register Descriptions (Flash Control Offset) ..... 299
7.5 Memory Register Descriptions (System Control Offset) ..... 311
**8 Micro Direct Memory Access (µDMA) ..... 335**
8.1 Block Diagram ..... 336
8.2 Functional Description ..... 336
8.2.1 Channel Assignments ..... 337
8.2.2 Priority ..... 338
8.2.3 Arbitration Size ..... 338
8.2.4 Request Types ..... 338
8.2.5 Channel Configuration ..... 339
8.2.6 Transfer Modes ..... 341
8.2.7 Transfer Size and Increment ..... 349
8.2.8 Peripheral Interface ..... 349
8.2.9 Software Request ..... 349
8.2.10 Interrupts and Errors ..... 350
8.3 Initialization and Configuration ..... 350
8.3.1 Module Initialization ..... 350
8.3.2 Configuring a Memory-to-Memory Transfer ..... 351
8.3.3 Configuring a Peripheral for Simple Transmit ..... 352

8.3.4 Configuring a Peripheral for Ping-Pong Receive ..... 354
8.3.5 Configuring Channel Assignments ..... 356
8.4 Register Map ..... 356
8.5 µDMA Channel Control Structure ..... 358
8.6 µDMA Register Descriptions ..... 365
**9 General-Purpose Input/Outputs (GPIOs) ..... 395**
9.1 Signal Description ..... 395
9.2 Functional Description ..... 399
9.2.1 Data Control ..... 401
9.2.2 Interrupt Control ..... 402
9.2.3 Mode Control ..... 403
9.2.4 Commit Control ..... 403
9.2.5 Pad Control ..... 404
9.2.6 Identification ..... 404
9.3 Initialization and Configuration ..... 404
9.4 Register Map ..... 405
9.5 Register Descriptions ..... 407
**10 General-Purpose Timers ..... 449**
10.1 Block Diagram ..... 449
10.2 Signal Description ..... 450
10.3 Functional Description ..... 453
10.3.1 GPTM Reset Conditions ..... 454
10.3.2 Timer Modes ..... 454
10.3.3 DMA Operation ..... 459
10.3.4 Accessing Concatenated Register Values ..... 460
10.4 Initialization and Configuration ..... 460
10.4.1 One-Shot/Periodic Timer Mode ..... 460
10.4.2 Real-Time Clock (RTC) Mode ..... 461
10.4.3 Input Edge-Count Mode ..... 461
10.4.4 Input Edge Timing Mode ..... 462
10.4.5 PWM Mode ..... 462
10.5 Register Map ..... 463
10.6 Register Descriptions ..... 464
**11 Watchdog Timers ..... 495**
11.1 Block Diagram ..... 496
11.2 Functional Description ..... 496
11.2.1 Register Access Timing ..... 497
11.3 Initialization and Configuration ..... 497
11.4 Register Map ..... 497
11.5 Register Descriptions ..... 498
**12 Analog-to-Digital Converter (ADC) ..... 520**
12.1 Block Diagram ..... 521
12.2 Signal Description ..... 522
12.3 Functional Description ..... 523
12.3.1 Sample Sequencers ..... 523
12.3.2 Module Control ..... 524
12.3.3 Hardware Sample Averaging Circuit ..... 527

12.3.4 Analog-to-Digital Converter ..... 528
12.3.5 Differential Sampling ..... 531
12.3.6 Internal Temperature Sensor ..... 533
12.3.7 Digital Comparator Unit ..... 534
12.4 Initialization and Configuration ..... 538
12.4.1 Module Initialization ..... 538
12.4.2 Sample Sequencer Configuration ..... 539
12.5 Register Map ..... 539
12.6 Register Descriptions ..... 541
**13 Universal Asynchronous Receivers/Transmitters (UARTs) ..... 598**
13.1 Block Diagram ..... 599
13.2 Signal Description ..... 599
13.3 Functional Description ..... 601
13.3.1 Transmit/Receive Logic ..... 601
13.3.2 Baud-Rate Generation ..... 601
13.3.3 Data Transmission ..... 602
13.3.4 Serial IR (SIR) ..... 603
13.3.5 ISO 7816 Support ..... 604
13.3.6 Modem Handshake Support ..... 604
13.3.7 LIN Support ..... 605
13.3.8 FIFO Operation ..... 607
13.3.9 Interrupts ..... 607
13.3.10 Loopback Operation ..... 608
13.3.11 DMA Operation ..... 608
13.4 Initialization and Configuration ..... 608
13.5 Register Map ..... 610
13.6 Register Descriptions ..... 611
**14 Synchronous Serial Interface (SSI) ..... 659**
14.1 Block Diagram ..... 660
14.2 Signal Description ..... 660
14.3 Functional Description ..... 661
14.3.1 Bit Rate Generation ..... 662
14.3.2 FIFO Operation ..... 662
14.3.3 Interrupts ..... 662
14.3.4 Frame Formats ..... 663
14.3.5 DMA Operation ..... 670
14.4 Initialization and Configuration ..... 671
14.5 Register Map ..... 672
14.6 Register Descriptions ..... 673
**15 Inter-Integrated Circuit ($I^2C$) Interface ..... 701**
15.1 Block Diagram ..... 702
15.2 Signal Description ..... 702
15.3 Functional Description ..... 703
15.3.1 $I^2C$ Bus Functional Overview ..... 703
15.3.2 Available Speed Modes ..... 705
15.3.3 Interrupts ..... 706
15.3.4 Loopback Operation ..... 707
15.3.5 Command Sequence Flow Charts ..... 708

15.4 Initialization and Configuration ..... 715
15.5 Register Map ..... 716
15.6 Register Descriptions ($I^2C$ Master) ..... 717
15.7 Register Descriptions ($I^2C$ Slave) ..... 729
**16 Pin Diagram ..... 738**
**17 Signal Tables ..... 740**
17.1 100-Pin LQFP Package Pin Tables ..... 741
17.2 108-Ball BGA Package Pin Tables ..... 761
17.3 Connections for Unused Signals ..... 782
**18 Operating Characteristics ..... 784**
**19 Electrical Characteristics ..... 785**
19.1 Maximum Ratings ..... 785
19.2 Recommended Operating Conditions ..... 785
19.3 Load Conditions ..... 786
19.4 JTAG and Boundary Scan ..... 786
19.5 Power and Brown-out ..... 788
19.6 Reset ..... 789
19.7 On-Chip Low Drop-Out (LDO) Regulator ..... 790
19.8 Clocks ..... 790
19.8.1 PLL Specifications ..... 790
19.8.2 PIOSC Specifications ..... 791
19.8.3 Internal 30-kHz Oscillator Specifications ..... 791
19.8.4 Hibernation Clock Source Specifications ..... 791
19.8.5 Main Oscillator Specifications ..... 792
19.8.6 System Clock Specification with ADC Operation ..... 793
19.9 Sleep Modes ..... 793
19.10 Hibernation Module ..... 793
19.11 Flash Memory ..... 795
19.12 GPIO Module ..... 795
19.13 Analog-to-Digital Converter (ADC) ..... 796
19.14 Synchronous Serial Interface (SSI) ..... 797
19.15 Inter-Integrated Circuit ($I^2C$) Interface ..... 799
19.16 Ethernet Controller ..... 800
19.17 Current Consumption ..... 801
19.17.1 Nominal Power Consumption ..... 801
19.17.2 Maximum Current Consumption ..... 802
**A Register Quick Reference ..... 804**
**B Ordering and Contact Information ..... 829**
B.1 Ordering Information ..... 829
B.2 Part Markings ..... 829
B.3 Kits ..... 830
B.4 Support Information ..... 830
**C Package Information ..... 831**
C.1 100-Pin LQFP Package ..... 831
C.1.1 Package Dimensions ..... 831
C.1.2 Tray Dimensions ..... 833

C.1.3 Tape and Reel Dimensions ..... 833
C.2 108-Ball BGA Package ..... 835
C.2.1 Package Dimensions ..... 835
C.2.2 Tray Dimensions ..... 837
C.2.3 Tape and Reel Dimensions ..... 838

# List of Figures


Figure 1-1. Stellaris LM3S1G58 Microcontroller High-Level Block Diagram ..... 46
Figure 2-1. CPU Block Diagram ..... 50
Figure 2-2. TPIU Block Diagram ..... 51
Figure 2-3. Cortex-M3 Register Set ..... 53
Figure 2-4. Bit-Band Mapping ..... 73
Figure 2-5. Data Storage ..... 74
Figure 2-6. Vector Table ..... 80
Figure 2-7. Exception Stack Frame ..... 82
Figure 3-1. SRD Use Example ..... 96
Figure 4-1. JTAG Module Block Diagram ..... 157
Figure 4-2. Test Access Port State Machine ..... 160
Figure 4-3. IDCODE Register Format ..... 166
Figure 4-4. BYPASS Register Format ..... 166
Figure 4-5. Boundary Scan Register Format ..... 167
Figure 5-1. Basic $\overline{\text{RST}}$ Configuration ..... 171
Figure 5-2. External Circuitry to Extend Power-On Reset ..... 172
Figure 5-3. Reset Circuit Controlled by Switch ..... 172
Figure 5-4. Power Architecture ..... 175
Figure 5-5. Main Clock Tree ..... 178
Figure 6-1. Hibernation Module Block Diagram ..... 264
Figure 6-2. Using a Crystal as the Hibernation Clock Source ..... 267
Figure 6-3. Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode ..... 267
Figure 7-1. Internal Memory Block Diagram ..... 290
Figure 8-1. µDMA Block Diagram ..... 336
Figure 8-2. Example of Ping-Pong µDMA Transaction ..... 342
Figure 8-3. Memory Scatter-Gather, Setup and Configuration ..... 344
Figure 8-4. Memory Scatter-Gather, µDMA Copy Sequence ..... 345
Figure 8-5. Peripheral Scatter-Gather, Setup and Configuration ..... 347
Figure 8-6. Peripheral Scatter-Gather, µDMA Copy Sequence ..... 348
Figure 9-1. Digital I/O Pads ..... 400
Figure 9-2. Analog/Digital I/O Pads ..... 401
Figure 9-3. GPIODATA Write Example ..... 402
Figure 9-4. GPIODATA Read Example ..... 402
Figure 10-1. GPTM Module Block Diagram ..... 450
Figure 10-2. Timer Daisy Chain ..... 456
Figure 10-3. Input Edge-Count Mode Example ..... 457
Figure 10-4. 16-Bit Input Edge-Time Mode Example ..... 458
Figure 10-5. 16-Bit PWM Mode Example ..... 459
Figure 11-1. WDT Module Block Diagram ..... 496
Figure 12-1. Implementation of Two ADC Blocks ..... 521
Figure 12-2. ADC Module Block Diagram ..... 521
Figure 12-3. ADC Sample Phases ..... 526
Figure 12-4. Doubling the ADC Sample Rate ..... 526
Figure 12-5. Skewed Sampling ..... 527
Figure 12-6. Sample Averaging Example ..... 528

Figure 12-7. Internal Voltage Conversion Result ..... 529
Figure 12-8. External Voltage Conversion Result with 3.0-V Setting ..... 530
Figure 12-9. External Voltage Conversion Result with 1.0-V Setting ..... 530
Figure 12-10. Differential Sampling Range, $V_{IN_ODD}$ = 1.5 V ..... 532
Figure 12-11. Differential Sampling Range, $V_{IN_ODD}$ = 0.75 V ..... 532
Figure 12-12. Differential Sampling Range, $V_{IN_ODD}$ = 2.25 V ..... 533
Figure 12-13. Internal Temperature Sensor Characteristic ..... 534
Figure 12-14. Low-Band Operation (CIC=0x0) ..... 536
Figure 12-15. Mid-Band Operation (CIC=0x1) ..... 537
Figure 12-16. High-Band Operation (CIC=0x3) ..... 538
Figure 13-1. UART Module Block Diagram ..... 599
Figure 13-2. UART Character Frame ..... 601
Figure 13-3. IrDA Data Modulation ..... 603
Figure 13-4. LIN Message ..... 606
Figure 13-5. LIN Synchronization Field ..... 607
Figure 14-1. SSI Module Block Diagram ..... 660
Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer) ..... 664
Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer) ..... 664
Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 ..... 665
Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 ..... 665
Figure 14-6. Freescale SPI Frame Format with SPO=0 and SPH=1 ..... 666
Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 ..... 667
Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 ..... 667
Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1 ..... 668
Figure 14-10. MICROWIRE Frame Format (Single Frame) ..... 669
Figure 14-11. MICROWIRE Frame Format (Continuous Transfer) ..... 670
Figure 14-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements ..... 670
Figure 15-1. $I^2C$ Block Diagram ..... 702
Figure 15-2. $I^2C$ Bus Configuration ..... 703
Figure 15-3. START and STOP Conditions ..... 704
Figure 15-4. Complete Data Transfer with a 7-Bit Address ..... 704
Figure 15-5. R/S Bit in First Byte ..... 705
Figure 15-6. Data Validity During Bit Transfer on the $I^2C$ Bus ..... 705
Figure 15-7. Master Single TRANSMIT ..... 709
Figure 15-8. Master Single RECEIVE ..... 710
Figure 15-9. Master TRANSMIT with Repeated START ..... 711
Figure 15-10. Master RECEIVE with Repeated START ..... 712
Figure 15-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START ..... 713
Figure 15-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START ..... 714
Figure 15-13. Slave Command Sequence ..... 715
Figure 16-1. 100-Pin LQFP Package Pin Diagram ..... 738
Figure 16-2. 108-Ball BGA Package Pin Diagram (Top View) ..... 739
Figure 19-1. Load Conditions ..... 786
Figure 19-2. JTAG Test Clock Input Timing ..... 787
Figure 19-3. JTAG Test Access Port (TAP) Timing ..... 787
Figure 19-4. Power-On Reset Timing ..... 788

Figure 19-5. Brown-Out Reset Timing ..... 788
Figure 19-6. Power-On Reset and Voltage Parameters ..... 789
Figure 19-7. External Reset Timing ($\overline{RST}$) ..... 789
Figure 19-8. Software Reset Timing ..... 789
Figure 19-9. Watchdog Reset Timing ..... 790
Figure 19-10. MOSC Failure Reset Timing ..... 790
Figure 19-11. Hibernation Module Timing with Internal Oscillator Running in Hibernation ..... 794
Figure 19-12. Hibernation Module Timing with Internal Oscillator Stopped in Hibernation ..... 794
Figure 19-13. ADC Input Equivalency Diagram ..... 797
Figure 19-14. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement ..... 798
Figure 19-15. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer ..... 798
Figure 19-16. SSI Timing for SPI Frame Format (FRF=00), with SPH=1 ..... 799
Figure 19-17. $I^2C$ Timing ..... 800
Figure C-1. 100-Pin LQFP Package Dimensions ..... 831
Figure C-2. 100-Pin LQFP Tray Dimensions ..... 833
Figure C-3. 100-Pin LQFP Tape and Reel Dimensions ..... 834
Figure C-4. 108-Ball BGA Package Dimensions ..... 835
Figure C-5. 108-Ball BGA Tray Dimensions ..... 837
Figure C-6. 108-Ball BGA Tape and Reel Dimensions ..... 838

# List of Tables


Table 1. Revision History ..... 26
Table 2. Documentation Conventions ..... 28
Table 2-1. Summary of Processor Mode, Privilege Level, and Stack Use ..... 53
Table 2-2. Processor Register Map ..... 54
Table 2-3. PSR Register Combinations ..... 59
Table 2-4. Memory Map ..... 67
Table 2-5. Memory Access Behavior ..... 70
Table 2-6. SRAM Memory Bit-Banding Regions ..... 72
Table 2-7. Peripheral Memory Bit-Banding Regions ..... 72
Table 2-8. Exception Types ..... 77
Table 2-9. Interrupts ..... 78
Table 2-10. Exception Return Behavior ..... 83
Table 2-11. Faults ..... 83
Table 2-12. Fault Status and Fault Address Registers ..... 85
Table 2-13. Cortex-M3 Instruction Summary ..... 87
Table 3-1. Core Peripheral Register Regions ..... 90
Table 3-2. Memory Attributes Summary ..... 93
Table 3-3. TEX, S, C, and B Bit Field Encoding ..... 96
Table 3-4. Cache Policy for Memory Attribute Encoding ..... 97
Table 3-5. AP Bit Field Encoding ..... 97
Table 3-6. Memory Region Attributes for Stellaris Microcontrollers ..... 97
Table 3-7. Peripherals Register Map ..... 98
Table 3-8. Interrupt Priority Levels ..... 125
Table 3-9. Example SIZE Field Values ..... 153
Table 4-1. Signals for JTAG_SWD_SWO (100LQFP) ..... 157
Table 4-2. Signals for JTAG_SWD_SWO (108BGA) ..... 158
Table 4-3. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion ..... 159
Table 4-4. JTAG Instruction Register Commands ..... 164
Table 5-1. Signals for System Control & Clocks (100LQFP) ..... 168
Table 5-2. Signals for System Control & Clocks (108BGA) ..... 168
Table 5-3. Reset Sources ..... 169
Table 5-4. Clock Source Options ..... 176
Table 5-5. Possible System Clock Frequencies Using the SYSDIV Field ..... 179
Table 5-6. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field ..... 179
Table 5-7. Examples of Possible System Clock Frequencies with DIV400=1 ..... 180
Table 5-8. System Control Register Map ..... 185
Table 5-9. RCC2 Fields that Override RCC Fields ..... 205
Table 6-1. Signals for Hibernate (100LQFP) ..... 264
Table 6-2. Signals for Hibernate (108BGA) ..... 265
Table 6-3. Hibernation Module Clock Operation ..... 270
Table 6-4. Hibernation Module Register Map ..... 273
Table 7-1. Flash Memory Protection Policy Combinations ..... 294
Table 7-2. User-Programmable Flash Memory Resident Registers ..... 297
Table 7-3. Flash Register Map ..... 298
Table 8-1. µDMA Channel Assignments ..... 337
Table 8-2. Request Type Support ..... 339

Table 8-3. Control Structure Memory Map ..... 340
Table 8-4. Channel Control Structure ..... 340
Table 8-5. µDMA Read Example: 8-Bit Peripheral ..... 349
Table 8-6. µDMA Interrupt Assignments ..... 350
Table 8-7. Channel Control Structure Offsets for Channel 30 ..... 351
Table 8-8. Channel Control Word Configuration for Memory Transfer Example ..... 351
Table 8-9. Channel Control Structure Offsets for Channel 7 ..... 352
Table 8-10. Channel Control Word Configuration for Peripheral Transmit Example ..... 353
Table 8-11. Primary and Alternate Channel Control Structure Offsets for Channel 8 ..... 354
Table 8-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example ..... 355
Table 8-13. µDMA Register Map ..... 357
Table 9-1. GPIO Pins With Non-Zero Reset Values ..... 396
Table 9-2. GPIO Pins and Alternate Functions (100LQFP) ..... 396
Table 9-3. GPIO Pins and Alternate Functions (108BGA) ..... 398
Table 9-4. GPIO Pad Configuration Examples ..... 404
Table 9-5. GPIO Interrupt Configuration Example ..... 405
Table 9-6. GPIO Pins With Non-Zero Reset Values ..... 406
Table 9-7. GPIO Register Map ..... 406
Table 9-8. GPIO Pins With Non-Zero Reset Values ..... 418
Table 9-9. GPIO Pins With Non-Zero Reset Values ..... 424
Table 9-10. GPIO Pins With Non-Zero Reset Values ..... 426
Table 9-11. GPIO Pins With Non-Zero Reset Values ..... 429
Table 9-12. GPIO Pins With Non-Zero Reset Values ..... 435
Table 10-1. Available CCP Pins ..... 450
Table 10-2. Signals for General-Purpose Timers (100LQFP) ..... 451
Table 10-3. Signals for General-Purpose Timers (108BGA) ..... 452
Table 10-4. General-Purpose Timer Capabilities ..... 453
Table 10-5. 16-Bit Timer With Prescaler Configurations ..... 455
Table 10-6. Timers Register Map ..... 463
Table 11-1. Watchdog Timers Register Map ..... 498
Table 12-1. Signals for ADC (100LQFP) ..... 522
Table 12-2. Signals for ADC (108BGA) ..... 522
Table 12-3. Samples and FIFO Depth of Sequencers ..... 523
Table 12-4. Differential Sampling Pairs ..... 531
Table 12-5. ADC Register Map ..... 539
Table 13-1. Signals for UART (100LQFP) ..... 600
Table 13-2. Signals for UART (108BGA) ..... 600
Table 13-3. Flow Control Mode ..... 605
Table 13-4. UART Register Map ..... 610
Table 14-1. Signals for SSI (100LQFP) ..... 661
Table 14-2. Signals for SSI (108BGA) ..... 661
Table 14-3. SSI Register Map ..... 672
Table 15-1. Signals for I2C (100LQFP) ..... 702
Table 15-2. Signals for I2C (108BGA) ..... 702
Table 15-3. Examples of $I^2C$ Master Timer Period versus Speed Mode ..... 706
Table 15-4. Inter-Integrated Circuit ($I^2C$) Interface Register Map ..... 716
Table 15-5. Write Field Decoding for I2CMCS[3:0] Field ..... 721

Table 17-1. GPIO Pins With Default Alternate Functions ..... 740
Table 17-2. Signals by Pin Number ..... 741
Table 17-3. Signals by Signal Name ..... 747
Table 17-4. Signals by Function, Except for GPIO ..... 753
Table 17-5. GPIO Pins and Alternate Functions ..... 757
Table 17-6. Possible Pin Assignments for Alternate Functions ..... 760
Table 17-7. Signals by Pin Number ..... 761
Table 17-8. Signals by Signal Name ..... 768
Table 17-9. Signals by Function, Except for GPIO ..... 774
Table 17-10. GPIO Pins and Alternate Functions ..... 778
Table 17-11. Possible Pin Assignments for Alternate Functions ..... 781
Table 17-12. Connections for Unused Signals (100-Pin LQFP) ..... 782
Table 17-13. Connections for Unused Signals (108-Ball BGA) ..... 783
Table 18-1. Temperature Characteristics ..... 784
Table 18-2. Thermal Characteristics ..... 784
Table 18-3. ESD Absolute Maximum Ratings ..... 784
Table 19-1. Maximum Ratings ..... 785
Table 19-2. Recommended DC Operating Conditions ..... 785
Table 19-3. JTAG Characteristics ..... 786
Table 19-4. Power Characteristics ..... 788
Table 19-5. Reset Characteristics ..... 789
Table 19-6. LDO Regulator Characteristics ..... 790
Table 19-7. Phase Locked Loop (PLL) Characteristics ..... 790
Table 19-8. Actual PLL Frequency ..... 791
Table 19-9. PIOSC Clock Characteristics ..... 791
Table 19-10. 30-kHz Clock Characteristics ..... 791
Table 19-11. Hibernation Clock Characteristics ..... 791
Table 19-12. HIB Oscillator Input Characteristics ..... 792
Table 19-13. Main Oscillator Clock Characteristics ..... 792
Table 19-14. Supported MOSC Crystal Frequencies ..... 792
Table 19-15. System Clock Characteristics with ADC Operation ..... 793
Table 19-16. Sleep Modes AC Characteristics ..... 793
Table 19-17. Hibernation Module Battery Characteristics ..... 793
Table 19-18. Hibernation Module AC Characteristics ..... 794
Table 19-19. Flash Memory Characteristics ..... 795
Table 19-20. GPIO Module Characteristics ..... 795
Table 19-21. ADC Characteristics ..... 796
Table 19-22. ADC Module External Reference Characteristics ..... 797
Table 19-23. ADC Module Internal Reference Characteristics ..... 797
Table 19-24. SSI Characteristics ..... 797
Table 19-25. $I^2C$ Characteristics ..... 799
Table 19-26. 100BASE-TX Transmitter Characteristics ..... 800
Table 19-27. 100BASE-TX Transmitter Characteristics (informative) ..... 800
Table 19-28. 100BASE-TX Receiver Characteristics ..... 800
Table 19-29. 10BASE-T Transmitter Characteristics ..... 800
Table 19-30. 10BASE-T Transmitter Characteristics (informative) ..... 801
Table 19-31. 10BASE-T Receiver Characteristics ..... 801
Table 19-32. Isolation Transformers ..... 801

Table 19-33. Nominal Power Consumption ..... 802
Table 19-34. Detailed Current Specifications ..... 802
Table 19-35. Hibernation Detailed Current Specifications ..... 803
Table 19-36. External $V_{DDC}$ Source Current Specifications ..... 803
Table B-1. Part Ordering Information ..... 829

# List of Registers

**The Cortex-M3 Processor ..... 48**
Register 1: Cortex General-Purpose Register 0 (R0) ..... 55
Register 2: Cortex General-Purpose Register 1 (R1) ..... 55
Register 3: Cortex General-Purpose Register 2 (R2) ..... 55
Register 4: Cortex General-Purpose Register 3 (R3) ..... 55
Register 5: Cortex General-Purpose Register 4 (R4) ..... 55
Register 6: Cortex General-Purpose Register 5 (R5) ..... 55
Register 7: Cortex General-Purpose Register 6 (R6) ..... 55
Register 8: Cortex General-Purpose Register 7 (R7) ..... 55
Register 9: Cortex General-Purpose Register 8 (R8) ..... 55
Register 10: Cortex General-Purpose Register 9 (R9) ..... 55
Register 11: Cortex General-Purpose Register 10 (R10) ..... 55
Register 12: Cortex General-Purpose Register 11 (R11) ..... 55
Register 13: Cortex General-Purpose Register 12 (R12) ..... 55
Register 14: Stack Pointer (SP) ..... 56
Register 15: Link Register (LR) ..... 57
Register 16: Program Counter (PC) ..... 58
Register 17: Program Status Register (PSR) ..... 59
Register 18: Priority Mask Register (PRIMASK) ..... 63
Register 19: Fault Mask Register (FAULTMASK) ..... 64
Register 20: Base Priority Mask Register (BASEPRI) ..... 65
Register 21: Control Register (CONTROL) ..... 66
**Cortex-M3 Peripherals ..... 90**
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010 ..... 101
Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014 ..... 103
Register 3: SysTick Current Value Register (STCURRENT), offset 0x018 ..... 104
Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100 ..... 105
Register 5: Interrupt 32-54 Set Enable (EN1), offset 0x104 ..... 106
Register 6: Interrupt 0-31 Clear Enable (DIS0), offset 0x180 ..... 107
Register 7: Interrupt 32-54 Clear Enable (DIS1), offset 0x184 ..... 108
Register 8: Interrupt 0-31 Set Pending (PEND0), offset 0x200 ..... 109
Register 9: Interrupt 32-54 Set Pending (PEND1), offset 0x204 ..... 110
Register 10: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280 ..... 111
Register 11: Interrupt 32-54 Clear Pending (UNPEND1), offset 0x284 ..... 112
Register 12: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300 ..... 113
Register 13: Interrupt 32-54 Active Bit (ACTIVE1), offset 0x304 ..... 114
Register 14: Interrupt 0-3 Priority (PRI0), offset 0x400 ..... 115
Register 15: Interrupt 4-7 Priority (PRI1), offset 0x404 ..... 115
Register 16: Interrupt 8-11 Priority (PRI2), offset 0x408 ..... 115
Register 17: Interrupt 12-15 Priority (PRI3), offset 0x40C ..... 115
Register 18: Interrupt 16-19 Priority (PRI4), offset 0x410 ..... 115
Register 19: Interrupt 20-23 Priority (PRI5), offset 0x414 ..... 115
Register 20: Interrupt 24-27 Priority (PRI6), offset 0x418 ..... 115
Register 21: Interrupt 28-31 Priority (PRI7), offset 0x41C ..... 115
Register 22: Interrupt 32-35 Priority (PRI8), offset 0x420 ..... 115

Register 23: Interrupt 36-39 Priority (PRI9), offset 0x424 ..... 115
Register 24: Interrupt 40-43 Priority (PRI10), offset 0x428 ..... 115
Register 25: Interrupt 44-47 Priority (PRI11), offset 0x42C ..... 115
Register 26: Interrupt 48-51 Priority (PRI12), offset 0x430 ..... 115
Register 27: Interrupt 52-54 Priority (PRI13), offset 0x434 ..... 115
Register 28: Software Trigger Interrupt (SWTRIG), offset 0xF00 ..... 117
Register 29: Auxiliary Control (ACTLR), offset 0x008 ..... 118
Register 30: CPU ID Base (CPUID), offset 0xD00 ..... 120
Register 31: Interrupt Control and State (INTCTRL), offset 0xD04 ..... 121
Register 32: Vector Table Offset (VTABLE), offset 0xD08 ..... 124
Register 33: Application Interrupt and Reset Control (APINT), offset 0xD0C ..... 125
Register 34: System Control (SYSCTRL), offset 0xD10 ..... 127
Register 35: Configuration and Control (CFGCTRL), offset 0xD14 ..... 129
Register 36: System Handler Priority 1 (SYSPRI1), offset 0xD18 ..... 131
Register 37: System Handler Priority 2 (SYSPRI2), offset 0xD1C ..... 132
Register 38: System Handler Priority 3 (SYSPRI3), offset 0xD20 ..... 133
Register 39: System Handler Control and State (SYSHNDCTRL), offset 0xD24 ..... 134
Register 40: Configurable Fault Status (FAULTSTAT), offset 0xD28 ..... 138
Register 41: Hard Fault Status (HFAULTSTAT), offset 0xD2C ..... 144
Register 42: Memory Management Fault Address (MMADDR), offset 0xD34 ..... 145
Register 43: Bus Fault Address (FAULTADDR), offset 0xD38 ..... 146
Register 44: MPU Type (MPUTYPE), offset 0xD90 ..... 147
Register 45: MPU Control (MPUCTRL), offset 0xD94 ..... 148
Register 46: MPU Region Number (MPUNUMBER), offset 0xD98 ..... 150
Register 47: MPU Region Base Address (MPUBASE), offset 0xD9C ..... 151
Register 48: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4 ..... 151
Register 49: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC ..... 151
Register 50: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4 ..... 151
Register 51: MPU Region Attribute and Size (MPUATTR), offset 0xDA0 ..... 153
Register 52: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8 ..... 153
Register 53: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0 ..... 153
Register 54: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8 ..... 153

**System Control ..... 168**

Register 1: Device Identification 0 (DID0), offset 0x000 ..... 187
Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030 ..... 189
Register 3: Raw Interrupt Status (RIS), offset 0x050 ..... 190
Register 4: Interrupt Mask Control (IMC), offset 0x054 ..... 192
Register 5: Masked Interrupt Status and Clear (MISC), offset 0x058 ..... 194
Register 6: Reset Cause (RESC), offset 0x05C ..... 196
Register 7: Run-Mode Clock Configuration (RCC), offset 0x060 ..... 198
Register 8: XTAL to PLL Translation (PLLCFG), offset 0x064 ..... 202
Register 9: GPIO High-Performance Bus Control (GPIOHBCTL), offset 0x06C ..... 203
Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070 ..... 205
Register 11: Main Oscillator Control (MOSCCTL), offset 0x07C ..... 208
Register 12: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144 ..... 209
Register 13: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150 ..... 211
Register 14: Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154 ..... 213
Register 15: Device Identification 1 (DID1), offset 0x004 ..... 214

Register 16: Device Capabilities 0 (DC0), offset 0x008 ..... 216
Register 17: Device Capabilities 1 (DC1), offset 0x010 ..... 217
Register 18: Device Capabilities 2 (DC2), offset 0x014 ..... 219
Register 19: Device Capabilities 3 (DC3), offset 0x018 ..... 221
Register 20: Device Capabilities 4 (DC4), offset 0x01C ..... 223
Register 21: Device Capabilities 5 (DC5), offset 0x020 ..... 225
Register 22: Device Capabilities 6 (DC6), offset 0x024 ..... 226
Register 23: Device Capabilities 7 (DC7), offset 0x028 ..... 227
Register 24: Device Capabilities 8 ADC Channels (DC8), offset 0x02C ..... 231
Register 25: Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190 ..... 234
Register 26: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0 ..... 236
Register 27: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 ..... 237
Register 28: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 ..... 239
Register 29: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 ..... 241
Register 30: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 ..... 243
Register 31: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 ..... 245
Register 32: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 ..... 248
Register 33: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 ..... 251
Register 34: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 ..... 253
Register 35: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 ..... 255
Register 36: Software Reset Control 0 (SRCR0), offset 0x040 ..... 257
Register 37: Software Reset Control 1 (SRCR1), offset 0x044 ..... 259
Register 38: Software Reset Control 2 (SRCR2), offset 0x048 ..... 261

**Hibernation Module ..... 263**

Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000 ..... 274
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004 ..... 275
Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008 ..... 276
Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C ..... 277
Register 5: Hibernation Control (HIBCTL), offset 0x010 ..... 278
Register 6: Hibernation Interrupt Mask (HIBIM), offset 0x014 ..... 281
Register 7: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018 ..... 283
Register 8: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C ..... 285
Register 9: Hibernation Interrupt Clear (HIBIC), offset 0x020 ..... 287
Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024 ..... 288
Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C ..... 289

**Internal Memory ..... 290**

Register 1: Flash Memory Address (FMA), offset 0x000 ..... 300
Register 2: Flash Memory Data (FMD), offset 0x004 ..... 301
Register 3: Flash Memory Control (FMC), offset 0x008 ..... 302
Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C ..... 305
Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010 ..... 306
Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 ..... 307
Register 7: Flash Memory Control 2 (FMC2), offset 0x020 ..... 308
Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030 ..... 309
Register 9: Flash Control (FCTL), offset 0x0F8 ..... 310
Register 10: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C ..... 311
Register 11: ROM Control (RMCTL), offset 0x0F0 ..... 312
Register 12: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200 ..... 313

Register 13: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400 ..... 314
Register 14: Boot Configuration (BOOTCFG), offset 0x1D0 ..... 315
Register 15: User Register 0 (USER_REG0), offset 0x1E0 ..... 317
Register 16: User Register 1 (USER_REG1), offset 0x1E4 ..... 318
Register 17: User Register 2 (USER_REG2), offset 0x1E8 ..... 319
Register 18: User Register 3 (USER_REG3), offset 0x1EC ..... 320
Register 19: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204 ..... 321
Register 20: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208 ..... 322
Register 21: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C ..... 323
Register 22: Flash Memory Protection Read Enable 4 (FMPRE4), offset 0x210 ..... 324
Register 23: Flash Memory Protection Read Enable 5 (FMPRE5), offset 0x214 ..... 325
Register 24: Flash Memory Protection Read Enable 6 (FMPRE6), offset 0x218 ..... 326
Register 25: Flash Memory Protection Read Enable 7 (FMPRE7), offset 0x21C ..... 327
Register 26: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404 ..... 328
Register 27: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408 ..... 329
Register 28: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C ..... 330
Register 29: Flash Memory Protection Program Enable 4 (FMPPE4), offset 0x410 ..... 331
Register 30: Flash Memory Protection Program Enable 5 (FMPPE5), offset 0x414 ..... 332
Register 31: Flash Memory Protection Program Enable 6 (FMPPE6), offset 0x418 ..... 333
Register 32: Flash Memory Protection Program Enable 7 (FMPPE7), offset 0x41C ..... 334

**Micro Direct Memory Access (µDMA) ..... 335**

Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000 ..... 359
Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004 ..... 360
Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008 ..... 361
Register 4: DMA Status (DMASTAT), offset 0x000 ..... 366
Register 5: DMA Configuration (DMACFG), offset 0x004 ..... 368
Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008 ..... 369
Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C ..... 370
Register 8: DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010 ..... 371
Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014 ..... 372
Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018 ..... 373
Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C ..... 374
Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020 ..... 375
Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024 ..... 376
Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028 ..... 377
Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C ..... 378
Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030 ..... 379
Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034 ..... 380
Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038 ..... 381
Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C ..... 382
Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C ..... 383
Register 21: DMA Channel Assignment (DMACHASGN), offset 0x500 ..... 384
Register 22: DMA Channel Interrupt Status (DMACHIS), offset 0x504 ..... 385
Register 23: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0 ..... 386
Register 24: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4 ..... 387
Register 25: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8 ..... 388
Register 26: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC ..... 389
Register 27: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0 ..... 390

Register 28: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0 ..... 391
Register 29: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4 ..... 392
Register 30: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8 ..... 393
Register 31: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC ..... 394

**General-Purpose Input/Outputs (GPIOs) ..... 395**
Register 1: GPIO Data (GPIODATA), offset 0x000 ..... 408
Register 2: GPIO Direction (GPIODIR), offset 0x400 ..... 409
Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404 ..... 410
Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 ..... 411
Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C ..... 412
Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410 ..... 413
Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414 ..... 414
Register 8: GPIO Masked Interrupt Status (GPIOMIS), offset 0x418 ..... 415
Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C ..... 417
Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 ..... 418
Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 ..... 420
Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 ..... 421
Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 ..... 422
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C ..... 423
Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510 ..... 424
Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514 ..... 426
Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 ..... 428
Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C ..... 429
Register 19: GPIO Lock (GPIOLOCK), offset 0x520 ..... 431
Register 20: GPIO Commit (GPIOCR), offset 0x524 ..... 432
Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528 ..... 434
Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C ..... 435
Register 23: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 ..... 437
Register 24: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 ..... 438
Register 25: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 ..... 439
Register 26: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC ..... 440
Register 27: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 ..... 441
Register 28: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 ..... 442
Register 29: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 ..... 443
Register 30: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC ..... 444
Register 31: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 ..... 445
Register 32: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 ..... 446
Register 33: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 ..... 447
Register 34: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC ..... 448

**General-Purpose Timers ..... 449**
Register 1: GPTM Configuration (GPTMCFG), offset 0x000 ..... 465
Register 2: GPTM Timer A Mode (GPTMTAMR), offset 0x004 ..... 466
Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008 ..... 468
Register 4: GPTM Control (GPTMCTL), offset 0x00C ..... 470
Register 5: GPTM Interrupt Mask (GPTMIMR), offset 0x018 ..... 473
Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C ..... 475
Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 ..... 478
Register 8: GPTM Interrupt Clear (GPTMICR), offset 0x024 ..... 481

Register 9: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028 ..... 483
Register 10: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C ..... 484
Register 11: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030 ..... 485
Register 12: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034 ..... 486
Register 13: GPTM Timer A Prescale (GPTMTAPR), offset 0x038 ..... 487
Register 14: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C ..... 488
Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040 ..... 489
Register 16: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044 ..... 490
Register 17: GPTM Timer A (GPTMTAR), offset 0x048 ..... 491
Register 18: GPTM Timer B (GPTMTBR), offset 0x04C ..... 492
Register 19: GPTM Timer A Value (GPTMTAV), offset 0x050 ..... 493
Register 20: GPTM Timer B Value (GPTMTBV), offset 0x054 ..... 494

**Watchdog Timers ..... 495**

Register 1: Watchdog Load (WDTLOAD), offset 0x000 ..... 499
Register 2: Watchdog Value (WDTVALUE), offset 0x004 ..... 500
Register 3: Watchdog Control (WDTCTL), offset 0x008 ..... 501
Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C ..... 503
Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 ..... 504
Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 ..... 505
Register 7: Watchdog Test (WDTTEST), offset 0x418 ..... 506
Register 8: Watchdog Lock (WDTLOCK), offset 0xC00 ..... 507
Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 ..... 508
Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 ..... 509
Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 ..... 510
Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC ..... 511
Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 ..... 512
Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 ..... 513
Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 ..... 514
Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC ..... 515
Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 ..... 516
Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 ..... 517
Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 ..... 518
Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3 ), offset 0xFFC ..... 519

**Analog-to-Digital Converter (ADC) ..... 520**

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000 ..... 542
Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004 ..... 543
Register 3: ADC Interrupt Mask (ADCIM), offset 0x008 ..... 545
Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C ..... 547
Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010 ..... 550
Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014 ..... 552
Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018 ..... 556
Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 ..... 557
Register 9: ADC Sample Phase Control (ADCSPC), offset 0x024 ..... 559
Register 10: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 ..... 561
Register 11: ADC Sample Averaging Control (ADCSAC), offset 0x030 ..... 563
Register 12: ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034 ..... 564
Register 13: ADC Control (ADCCTL), offset 0x038 ..... 566
Register 14: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 ..... 567

Register 15: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 ..... 569
Register 16: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 ..... 572
Register 17: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 ..... 572
Register 18: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 ..... 572
Register 19: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 ..... 572
Register 20: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C ..... 573
Register 21: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C ..... 573
Register 22: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C ..... 573
Register 23: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC ..... 573
Register 24: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050 ..... 575
Register 25: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054 ..... 577
Register 26: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 ..... 579
Register 27: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 ..... 579
Register 28: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 ..... 580
Register 29: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 ..... 580
Register 30: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070 ..... 582
Register 31: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090 ..... 582
Register 32: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074 ..... 583
Register 33: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094 ..... 583
Register 34: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 ..... 585
Register 35: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 ..... 586
Register 36: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0 ..... 587
Register 37: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4 ..... 588
Register 38: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00 ..... 589
Register 39: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00 ..... 594
Register 40: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04 ..... 594
Register 41: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08 ..... 594
Register 42: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C ..... 594
Register 43: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10 ..... 594
Register 44: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14 ..... 594
Register 45: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18 ..... 594
Register 46: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C ..... 594
Register 47: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40 ..... 596
Register 48: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44 ..... 596
Register 49: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48 ..... 596
Register 50: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C ..... 596
Register 51: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50 ..... 596
Register 52: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54 ..... 596
Register 53: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58 ..... 596
Register 54: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C ..... 596

**Universal Asynchronous Receivers/Transmitters (UARTs) ..... 598**

Register 1: UART Data (UARTDR), offset 0x000 ..... 612
Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 ..... 614
Register 3: UART Flag (UARTFR), offset 0x018 ..... 617
Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020 ..... 620
Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 ..... 621
Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 ..... 622
Register 7: UART Line Control (UARTLCRH), offset 0x02C ..... 623

Register 8: UART Control (UARTCTL), offset 0x030 ..... 625
Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 ..... 629
Register 10: UART Interrupt Mask (UARTIM), offset 0x038 ..... 631
Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C ..... 635
Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040 ..... 638
Register 13: UART Interrupt Clear (UARTICR), offset 0x044 ..... 641
Register 14: UART DMA Control (UARTDMACTL), offset 0x048 ..... 643
Register 15: UART LIN Control (UARTLCTL), offset 0x090 ..... 644
Register 16: UART LIN Snap Shot (UARTLSS), offset 0x094 ..... 645
Register 17: UART LIN Timer (UARTLTIM), offset 0x098 ..... 646
Register 18: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 ..... 647
Register 19: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 ..... 648
Register 20: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 ..... 649
Register 21: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC ..... 650
Register 22: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 ..... 651
Register 23: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 ..... 652
Register 24: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 ..... 653
Register 25: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC ..... 654
Register 26: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 ..... 655
Register 27: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 ..... 656
Register 28: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 ..... 657
Register 29: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC ..... 658

**Synchronous Serial Interface (SSI) ..... 659**
Register 1: SSI Control 0 (SSICR0), offset 0x000 ..... 674
Register 2: SSI Control 1 (SSICR1), offset 0x004 ..... 676
Register 3: SSI Data (SSIDR), offset 0x008 ..... 678
Register 4: SSI Status (SSISR), offset 0x00C ..... 679
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010 ..... 681
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014 ..... 682
Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018 ..... 683
Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C ..... 685
Register 9: SSI Interrupt Clear (SSIICR), offset 0x020 ..... 687
Register 10: SSI DMA Control (SSIDMACTL), offset 0x024 ..... 688
Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 ..... 689
Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 ..... 690
Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 ..... 691
Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC ..... 692
Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 ..... 693
Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 ..... 694
Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 ..... 695
Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC ..... 696
Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 ..... 697
Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 ..... 698
Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 ..... 699
Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC ..... 700

**Inter-Integrated Circuit ($I^2C$) Interface ..... 701**
Register 1: $I^2C$ Master Slave Address (I2CMSA), offset 0x000 ..... 718
Register 2: $I^2C$ Master Control/Status (I2CMCS), offset 0x004 ..... 719

Register 3: I2C Master Data (I2CMDR), offset 0x008 ..... 723
Register 4: I2C Master Timer Period (I2CMTPR), offset 0x00C ..... 724
Register 5: I2C Master Interrupt Mask (I2CMIMR), offset 0x010 ..... 725
Register 6: I2C Master Raw Interrupt Status (I2CMRIS), offset 0x014 ..... 726
Register 7: I2C Master Masked Interrupt Status (I2CMMIS), offset 0x018 ..... 727
Register 8: I2C Master Interrupt Clear (I2CMICR), offset 0x01C ..... 728
Register 9: I2C Master Configuration (I2CMCR), offset 0x020 ..... 729
Register 10: I2C Slave Own Address (I2CSOAR), offset 0x800 ..... 730
Register 11: I2C Slave Control/Status (I2CSCSR), offset 0x804 ..... 731
Register 12: I2C Slave Data (I2CSDR), offset 0x808 ..... 733
Register 13: I2C Slave Interrupt Mask (I2CSIMR), offset 0x80C ..... 734
Register 14: I2C Slave Raw Interrupt Status (I2CSRIS), offset 0x810 ..... 735
Register 15: I2C Slave Masked Interrupt Status (I2CSMIS), offset 0x814 ..... 736
Register 16: I2C Slave Interrupt Clear (I2CSICR), offset 0x818 ..... 737

# Revision History

The revision history table notes changes made between the indicated revisions of the LM3S1G58 data sheet.

**Table 1. Revision History**

| Date | Revision | Description |
|---|---|---|
| July 2011 | 9970 | ■ Corrected "Reset Sources" table.<br>■ Added Important Note that **RCC** register must be written before **RCC2** register.<br>■ Added missing Start Calibration (CAL) bit to the **Precision Internal Oscillator Calibration (PIOSCCAL)** register.<br>■ Added missing **Precision Internal Oscillator Statistics (PIOSCSTAT)** register.<br>■ In Hibernation Module chapter, deleted section "Special Considerations When Using a 4.194304-MHz Crystal" as this content was added to the errata document.<br>■ Added a note that all GPIO signals are 5-V tolerant when configured as inputs except for PB0 and PB1, which are limited to 3.6 V.<br>■ Corrected LIN Mode bit names in **UART Interrupt Clear (UARTICR)** register.<br>■ Corrected pin number for $\overline{\text{RST}}$ in table "Connections for Unused Signals" (other pin tables were correct).<br>■ In the "Operating Characteristics" chapter:<br>– In the "Thermal Characteristics" table, the Thermal resistance value was changed.<br>– In the "ESD Absolute Maximum Ratings" table, the $V_{ESDCDM}$ parameter was changed and the $V_{ESDMM}$ parameter was deleted.<br>■ The "Electrical Characteristics" chapter was reorganized by module. In addition, some of the Recommended DC Operating Conditions, LDO Regulator, Clock, GPIO, Hibernation Module, ADC, and SSI characteristics were finalized.<br>■ Additional minor data sheet clarifications and corrections. |
| March 2011 | 9538 | Started tracking revision history. |

# About This Document

This data sheet provides reference information for the LM3S1G58 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M3 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 Stellaris® web site at www.ti.com/stellaris:

- *Stellaris® Errata*
- *ARM® Cortex™-M3 Errata*
- *Cortex™-M3 Instruction Set Technical User's Manual*
- *Stellaris® Boot Loader User's Guide*
- *Stellaris® Graphics Library User's Guide*
- *Stellaris® Peripheral Driver Library User's Guide*
- *Stellaris® ROM User's Guide*

The following related documents are also referenced:

- *ARM® Debug Interface V5 Architecture Specification*
- *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 28.

**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 67. |
| 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. |
| R/W | Software can read or write this field. |
| R/WC | Software can read or write this field. Writing to it with any value clears the register. |
| R/W1C | 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. |
| R/W1S | Software can read or write a 1 to this field. A write of a 0 to a R/W1S 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 Instruments is the industry leader in bringing 32-bit capabilities and the full benefits of ARM® Cortex™-M3-based microcontrollers to the broadest reach of the microcontroller market. For current users of 8- and 16-bit MCUs, Stellaris® with Cortex-M3 offers a direct path to the strongest ecosystem of development tools, software and knowledge in the industry. Designers who migrate to Stellaris benefit from great tools, small code footprint and outstanding performance. Even more important, designers can enter the ARM ecosystem with full confidence in a compatible roadmap from $1 to 1 GHz. For users of current 32-bit MCUs, the Stellaris family offers the industry's first implementation of Cortex-M3 and the Thumb-2 instruction set. With blazingly-fast responsiveness, Thumb-2 technology combines both 16-bit and 32-bit instructions to deliver the best balance of code density and performance. Thumb-2 uses 26 percent less memory than pure 32-bit code to reduce system cost while delivering 25 percent better performance. The Texas Instruments Stellaris family of microcontrollers—the first ARM Cortex-M3 based controllers—brings high-performance 32-bit computing to cost-sensitive embedded microcontroller applications. These pioneering parts deliver customers 32-bit performance at a cost equivalent to legacy 8- and 16-bit devices, all in a package with a small footprint.

The LM3S1G58 microcontroller has the following features:

- ARM Cortex-M3 Processor Core
  - 80-MHz operation; 100 DMIPS performance
  - ARM Cortex SysTick Timer
  - Nested Vectored Interrupt Controller (NVIC)
- On-Chip Memory
  - 384 KB single-cycle Flash memory up to 50 MHz; a prefetch buffer improves performance above 50 MHz
  - 64 KB single-cycle SRAM
  - Internal ROM loaded with StellarisWare® software:
    - Stellaris Peripheral Driver Library
    - Stellaris Boot Loader
    - Advanced Encryption Standard (AES) cryptography tables
    - Cyclic Redundancy Check (CRC) error detection functionality
- Advanced Serial Integration
  - Three UARTs with IrDA and ISO 7816 support (one UART with modem flow control and status)
  - Two $I^2C$ modules
  - Two Synchronous Serial Interface modules (SSI)
- System Integration
  - Direct Memory Access Controller (DMA)

  - System control and clocks including on-chip precision 16-MHz oscillator
  - Four 32-bit timers (up to eight 16-bit)
  - Eight Capture Compare PWM pins (CCP)
  - 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 60 GPIOs, depending on configuration
    - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
    - Independently configurable to 2, 4 or 8 mA drive capability
    - Up to 4 GPIOs can have 18 mA drive capability

- Analog
  - Two 12-bit Analog-to-Digital Converters (ADC) with 16 analog input channels and a sample rate of one million samples/second
  - 16 digital comparators
  - On-chip voltage regulator
- JTAG and ARM Serial Wire Debug (SWD)
- 100-pin LQFP package
- 108-ball BGA package
- Industrial (-40°C to 85°C) Temperature Range

The LM3S1G58 microcontroller is targeted for industrial applications, including remote monitoring, electronic point-of-sale machines, test and measurement equipment, network appliances and switches, factory automation, HVAC and building control, gaming equipment, and fire and security.

For applications requiring extreme conservation of power, the LM3S1G58 microcontroller features a battery-backed Hibernation module to efficiently power down the LM3S1G58 to a low-power state during extended periods of inactivity. With a power-up/power-down sequencer, a continuous time counter (RTC), a pair of match registers, an APB interface to the system bus, and dedicated non-volatile memory, the Hibernation module positions the LM3S1G58 microcontroller perfectly for battery applications.

In addition, the LM3S1G58 microcontroller offers the advantages of ARM's widely available development tools, System-on-Chip (SoC) infrastructure IP applications, and a large user community. Additionally, the microcontroller uses ARM's Thumb®-compatible Thumb-2 instruction set to reduce memory requirements and, thereby, cost. Finally, the LM3S1G58 microcontroller is code-compatible to all members of the extensive Stellaris family; providing flexibility to fit our customers' 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. See “Ordering and Contact Information” on page 829 for ordering information for Stellaris family devices.

# 1.1 Functional Overview

The following sections provide an overview of the features of the LM3S1G58 microcontroller. The page number in parentheses indicates where that feature is discussed in detail. Ordering and support information can be found in “Ordering and Contact Information” on page 829.

## 1.1.1 ARM Cortex-M3

The following sections provide an overview of the ARM Cortex-M3 processor core and instruction set, the integrated System Timer (SysTick) and the Nested Vectored Interrupt Controller.

### 1.1.1.1 Processor Core (see page 48)

All members of the Stellaris product family, including the LM3S1G58 microcontroller, are designed around an ARM Cortex-M3 processor core. The ARM Cortex-M3 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.

- 32-bit ARM Cortex-M3 architecture optimized for small-footprint embedded applications
- 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
- 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 multiplier
- 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
- Ultra-low power consumption with integrated sleep modes
- 80-MHz operation
- 1.25 DMIPS/MHz

#### 1.1.1.2 Memory Map (see page 67)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S1G58 controller can be found in “Memory Model” on page 67. Register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map.

#### 1.1.1.3 System Timer (SysTick) (see page 90)

ARM Cortex-M3 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.

#### 1.1.1.4 Nested Vectored Interrupt Controller (NVIC) (see page 91)

The LM3S1G58 controller includes the ARM Nested Vectored Interrupt Controller (NVIC). The NVIC and Cortex-M3 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 37 interrupts.

- Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining
- 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

#### 1.1.1.5 System Control Block (SCB) (see page 93)

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

#### 1.1.1.6 Memory Protection Unit (MPU) (see page 93)

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.

### 1.1.2 On-Chip Memory

The following sections describe the on-chip memory modules.

#### 1.1.2.1 SRAM (see page 291)

The LM3S1G58 microcontroller provides 64 KB of single-cycle on-chip SRAM. The internal SRAM of the Stellaris devices is located at offset 0x2000.0000 of the device memory map.

Because read-modify-write (RMW) operations are very time consuming, ARM has introduced *bit-banding* technology in the Cortex-M3 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 the SRAM using the Micro Direct Memory Access Controller (µDMA).

#### 1.1.2.2 Flash Memory (see page 293)

The LM3S1G58 microcontroller provides 384 KB of single-cycle on-chip Flash memory (above 50 MHz, the Flash memory can be accessed in a single cycle as long as the code is linear; branches incur a one-cycle stall). The Flash memory is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The 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.

#### 1.1.2.3 ROM (see page 291)

The LM3S1G58 ROM is preprogrammed with the following software and programs:

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

The Stellaris 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-M3 core. No special pragmas or custom assembly code prologue/epilogue functions are required. For applications that require in-field programmability, the royalty-free Stellaris 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 (e.g. XOR all bits) because it catches changes more readily.

### 1.1.3 Serial Communications Peripherals

The LM3S1G58 controller supports both asynchronous and synchronous serial communications with:

- Three UARTs with IrDA and ISO 7816 support (one UART with modem flow control and status)
- Two $I^2C$ modules
- Two Synchronous Serial Interface modules (SSI)

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

#### 1.1.3.1 UART (see page 598)

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 LM3S1G58 microcontroller includes three 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 three UARTs have the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 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
- Full modem handshake support (on UART1)
- LIN protocol 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

#### 1.1.3.2 $I^2C$ (see page 701)

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. Each $I^2C$ module supports both sending and receiving data as either a master or a slave and can operate simultaneously as both a master and a slave. Both the $I^2C$ master and slave can generate interrupts.

The LM3S1G58 microcontroller includes two $I^2C$ 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 transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
- Master and slave interrupt generation
  - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
  - Slave generates interrupts when data has been transferred or requested by a master or when a START or STOP condition is detected
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

#### 1.1.3.3 SSI (see page 659)

Synchronous Serial Interface (SSI) is a four-wire bi-directional communications interface that converts data between parallel and serial. The SSI 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 SSI module can be configured as either a master or slave device. As a slave device, the SSI 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 SSI module also includes a programmable bit rate clock divider and prescaler to generate the output serial clock derived from the SSI 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 LM3S1G58 microcontroller includes two SSI modules with the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- 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 FIFO contains 4 entries

### 1.1.4 System Integration

The LM3S1G58 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
- Four 32-bit timers (up to eight 16-bit)
- Eight Capture Compare PWM pins (CCP)
- 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 60 GPIOs, depending on configuration
  – Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
  – Independently configurable to 2, 4 or 8 mA drive capability
  – Up to 4 GPIOs can have 18 mA drive capability

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

#### 1.1.4.1 Direct Memory Access (see page 335)

The LM3S1G58 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-M3 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 arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  – Independently configured and operated channels

  - Dedicated channels for supported on-chip modules
  - Primary and secondary channel assignments
  - 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

#### 1.1.4.2 System Control and Clocks (see page 168)

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

  - Precision Oscillator (PIOSC): On-chip resource providing a 16 MHz ±1% frequency at room temperature
    - 16 MHz ±3% across temperature
    - Can be recalibrated with 7-bit trim resolution
    - Software power down control for low power modes
  - 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.
    - External crystal used with or without on-chip PLL: select supported frequencies from 1 MHz to 16.384 MHz.
    - External oscillator: from DC to maximum device speed
  - Internal 30-kHz Oscillator: on chip resource providing a 30 kHz ± 50% frequency, used during power-saving modes
  - 32.768-kHz external oscillator for the Hibernation Module: eliminates need for additional crystal for main clock source

- 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
  - MOSC failure

#### 1.1.4.3 Programmable Timers (see page 449)

Programmable timers can be used to count or time external events that drive the Timer input pins. Each 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.

The General-Purpose Timer Module (GPTM) contains four 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
  - 16-bit PWM mode with software-programmable output inversion of the PWM signal

- Count up or down
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- 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

#### 1.1.4.4 CCP Pins (see page 456)

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 LM3S1G58 microcontroller includes eight Capture Compare PWM pins (CCP) 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.

#### 1.1.4.5 Hibernation Module (see page 263)

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 counter (RTC)
  - Two 32-bit RTC match registers for timed wake-up and interrupt generation
  - RTC predivider trim for making fine adjustments to the clock rate
- Two mechanisms for power control
  - System power control using discrete external regulator
  - On-chip power control using internal switches under register control
- Dedicated pin for waking using an external signal

- RTC operational and hibernation memory valid as long as $V_{BAT}$ is valid
- Low-battery detection, signaling, and interrupt generation
- Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal; 32.768-kHz external oscillator can be used for main controller clock
- 64 32-bit words of non-volatile memory to save state during hibernation
- Programmable interrupts for RTC match, external wake, and low battery events

#### 1.1.4.6 Watchdog Timers (see page 495)

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 Stellaris Watchdog Timer can generate an 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 time-out. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

The LM3S1G58 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 Stellaris 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
- 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

#### 1.1.4.7 Programmable GPIOs (see page 395)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections. The Stellaris GPIO module is comprised of eight 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 0-60 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 740 for the signals available to each GPIO pin).

- Up to 60 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Fast toggle capable of a change every two clock cycles

- Two means of port access: either Advanced High-Performance Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- 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, and 8-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

### 1.1.5 Analog

The LM3S1G58 microcontroller provides analog functions integrated into the device, including:

- Two 12-bit Analog-to-Digital Converters (ADC) with 16 analog input channels and a sample rate of one million samples/second
- 16 digital comparators
- On-chip voltage regulator

The following provides more detail on these analog functions.

#### 1.1.5.1 ADC (see page 520)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. The Stellaris ADC module features 12-bit conversion resolution and supports 16 input channels plus an internal temperature sensor. Four buffered sample sequencers allow rapid sampling of up to 16 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 LM3S1G58 microcontroller provides two ADC modules with the following features:

- 16 shared analog input channels

- 12-bit precision ADC with an accurate 10-bit data compatibility mode
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5º to 337.5º
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - GPIO
- Hardware averaging of up to 64 samples
- Digital comparison unit providing eight digital comparators
- Converter uses an internal 3-V reference or an external 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

### 1.1.6 JTAG and ARM Serial Wire Debug (see page 156)

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, EXTEST and INTEST
- 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
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

### 1.1.7 Packaging and Temperature

- Industrial-range 100-pin RoHS-compliant LQFP package
- Industrial-range 108-ball RoHS-compliant BGA package

## 1.2 Target Applications

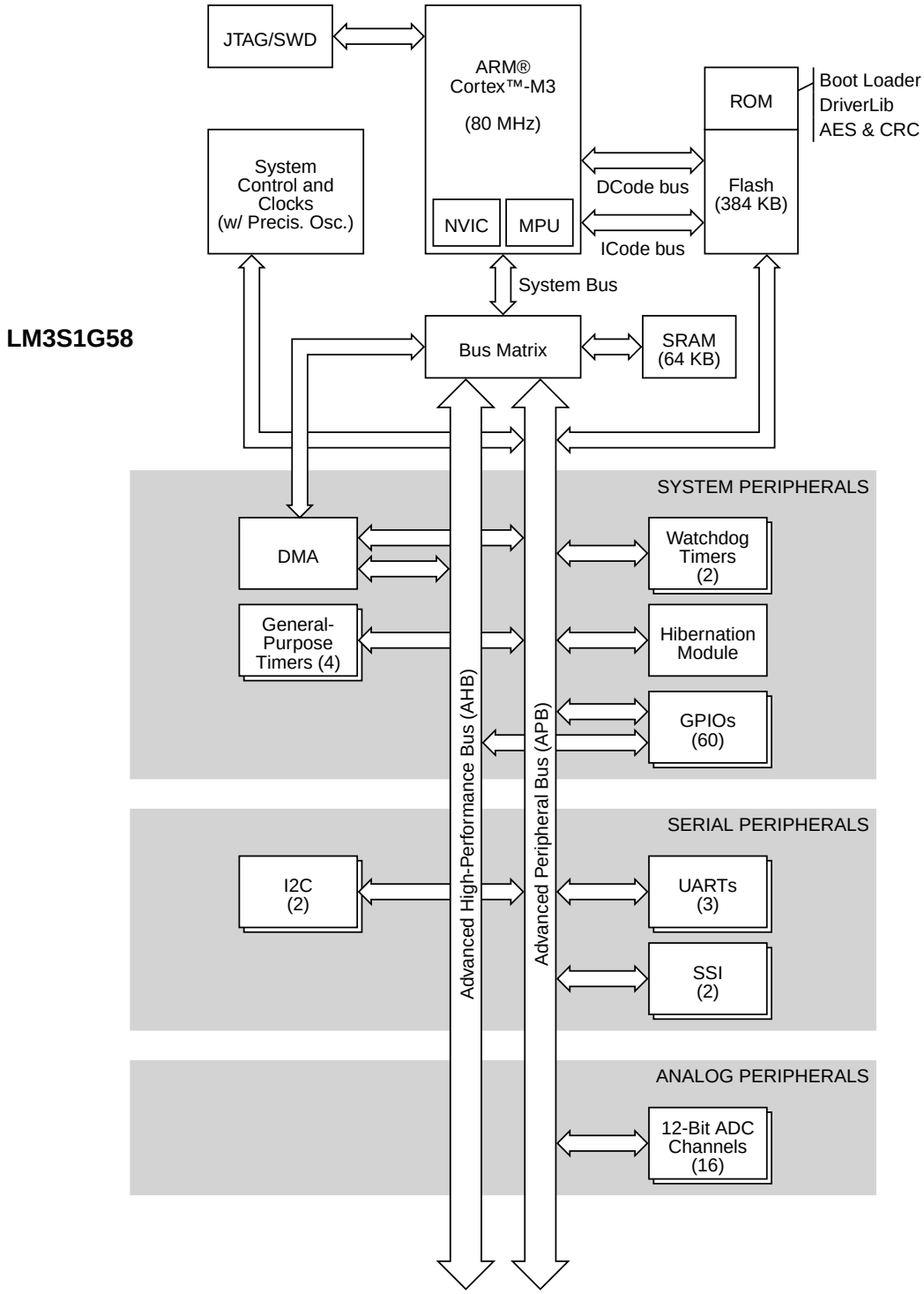
The Stellaris family is positioned for cost-conscious applications requiring significant control processing and connectivity capabilities such as:

- Test and measurement equipment
- Factory automation
- HVAC and building control
- Gaming equipment
- Motion control
- Medical instrumentation
- Fire and security
- Power and energy
- Transportation

## 1.3 High-Level Block Diagram

Figure 1-1 on page 46 depicts the features on the Stellaris LM3S1G58 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. Stellaris LM3S1G58 Microcontroller High-Level Block Diagram**



## 1.4 Hardware Details

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


- "Pin Diagram" on page 738
- "Signal Tables" on page 740
- "Operating Characteristics" on page 784
- "Electrical Characteristics" on page 785
- "Package Information" on page 831

# 2 The Cortex-M3 Processor

The ARM® Cortex™-M3 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™-M3 architecture optimized for small-footprint embedded applications
- 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
- 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 multiplier
- 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
- Ultra-low power consumption with integrated sleep modes
- 80-MHz operation
- 1.25 DMIPS/MHz

The Stellaris® family of microcontrollers builds on this core to bring high-performance 32-bit computing to cost-sensitive embedded microcontroller applications, such as factory automation and control, industrial control power devices, building and home automation, and stepper motor control.

This chapter provides information on the Stellaris implementation of the Cortex-M3 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™-M3 Instruction Set Technical User's Manual*.

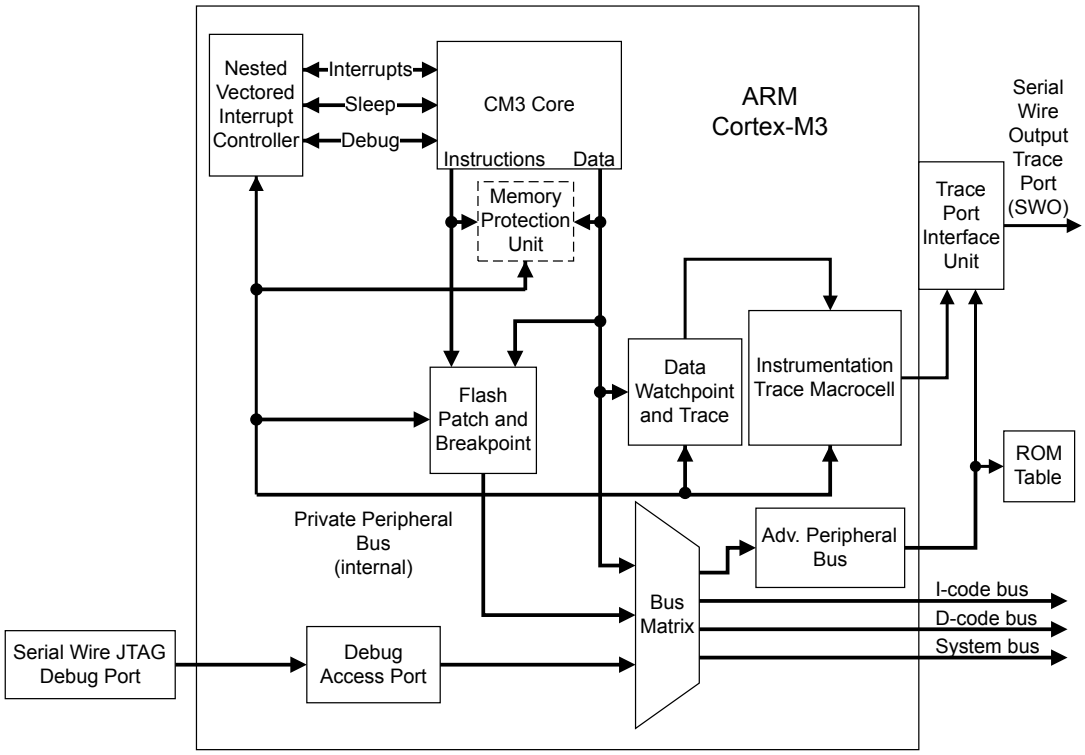
## 2.1 Block Diagram

The Cortex-M3 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 single-cycle 32x32 multiplication and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M3 processor implements tightly coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M3 processor implements a version of the Thumb® instruction set, ensuring high code density and reduced program memory requirements. The Cortex-M3 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-M3 processor closely integrates a nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The Stellaris 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-M3 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-M3 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-M3 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 Stellaris 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.

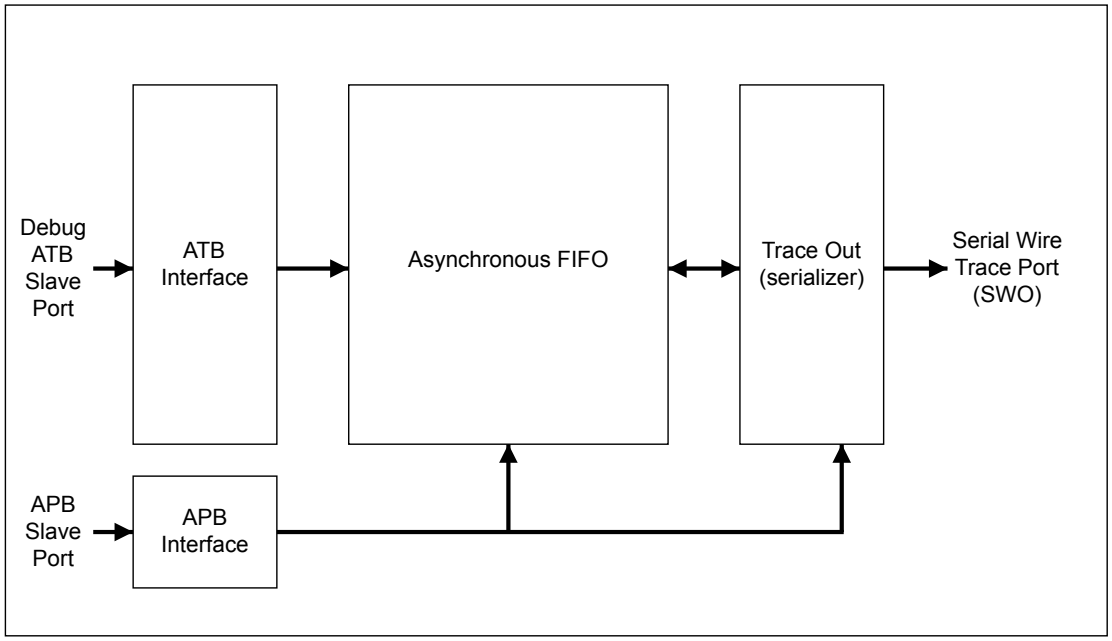
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words in the program code in the CODE memory region. This enables applications stored in a read-only area of Flash 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-M3 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-M3 trace data from the ITM, and an off-chip Trace Port Analyzer, as shown in Figure 2-2 on page 51.

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




### 2.2.4 Cortex-M3 System Component Details

The Cortex-M3 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 90).

- Nested Vectored Interrupt Controller (NVIC)

  An embedded interrupt controller that supports low latency interrupt processing (see “Nested Vectored Interrupt Controller (NVIC)” on page 91).

- 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 93).

- 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 93).

## 2.3 Programming Model

This section describes the Cortex-M3 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-M3 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-M3 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 66) 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 stack 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 independent copies of the stack pointer (see the **SP** register on page 56).

In Thread mode, the **CONTROL** register (see page 66) 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 53.

**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 66).

## 2.3.3 Register Map

Figure 2-3 on page 53 shows the Cortex-M3 register set. Table 2-2 on page 54 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-M3 Register Set**

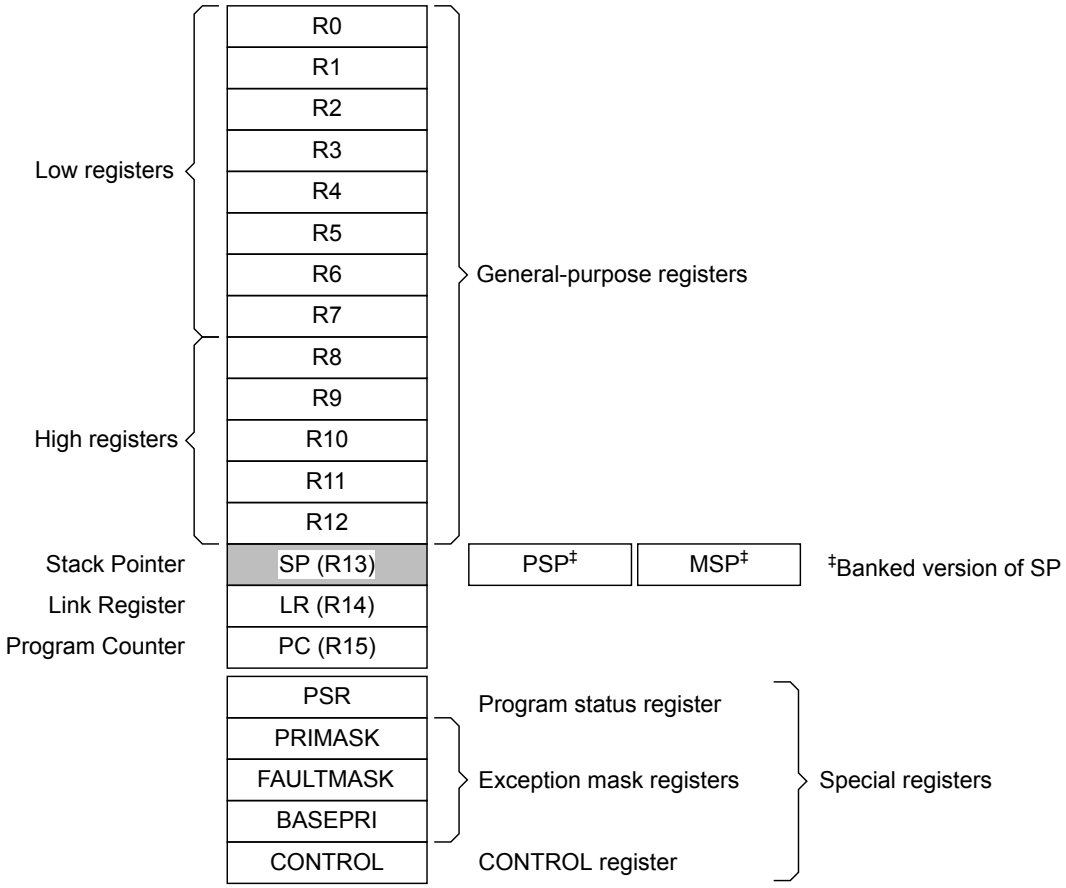


**Table 2-2. Processor Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| - | R0 | R/W | - | Cortex General-Purpose Register 0 | 55 |
| - | R1 | R/W | - | Cortex General-Purpose Register 1 | 55 |
| - | R2 | R/W | - | Cortex General-Purpose Register 2 | 55 |
| - | R3 | R/W | - | Cortex General-Purpose Register 3 | 55 |
| - | R4 | R/W | - | Cortex General-Purpose Register 4 | 55 |
| - | R5 | R/W | - | Cortex General-Purpose Register 5 | 55 |
| - | R6 | R/W | - | Cortex General-Purpose Register 6 | 55 |
| - | R7 | R/W | - | Cortex General-Purpose Register 7 | 55 |
| - | R8 | R/W | - | Cortex General-Purpose Register 8 | 55 |
| - | R9 | R/W | - | Cortex General-Purpose Register 9 | 55 |
| - | R10 | R/W | - | Cortex General-Purpose Register 10 | 55 |
| - | R11 | R/W | - | Cortex General-Purpose Register 11 | 55 |
| - | R12 | R/W | - | Cortex General-Purpose Register 12 | 55 |
| - | SP | R/W | - | Stack Pointer | 56 |
| - | LR | R/W | 0xFFFF.FFFF | Link Register | 57 |
| - | PC | R/W | - | Program Counter | 58 |
| - | PSR | R/W | 0x0100.0000 | Program Status Register | 59 |
| - | PRIMASK | R/W | 0x0000.0000 | Priority Mask Register | 63 |
| - | FAULTMASK | R/W | 0x0000.0000 | Fault Mask Register | 64 |
| - | BASEPRI | R/W | 0x0000.0000 | Base Priority Mask Register | 65 |
| - | CONTROL | R/W | 0x0000.0000 | Control Register | 66 |

### 2.3.4 Register Descriptions

This section lists and describes the Cortex-M3 registers, in the order shown in Figure 2-3 on page 53. 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)

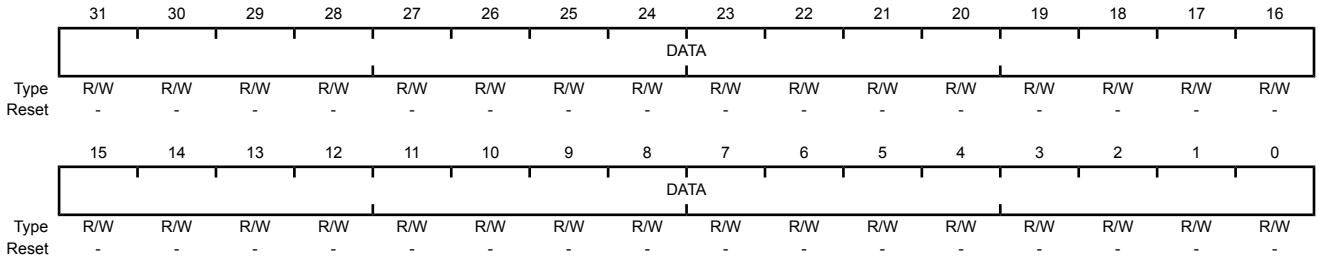
### Register 12: Cortex General-Purpose Register 11 (R11)

### 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 R/W, reset -



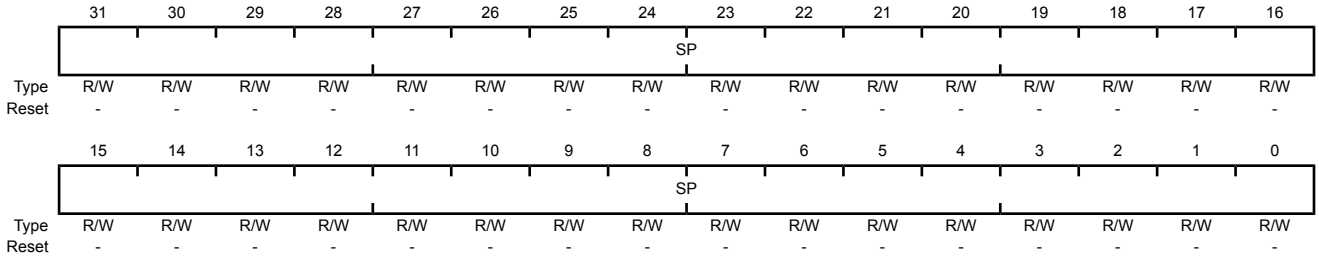

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | R/W | - | Register data. |

### 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 R/W, reset -




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

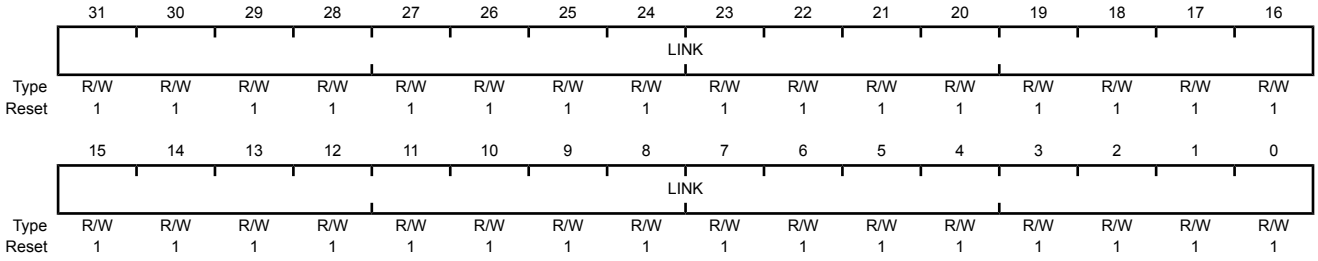
### Register 15: Link Register (LR)

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

EXC_RETURN is loaded into **LR** on exception entry. See Table 2-10 on page 83 for the values and description.

Link Register (LR)

Type R/W, reset 0xFFFF.FFFF



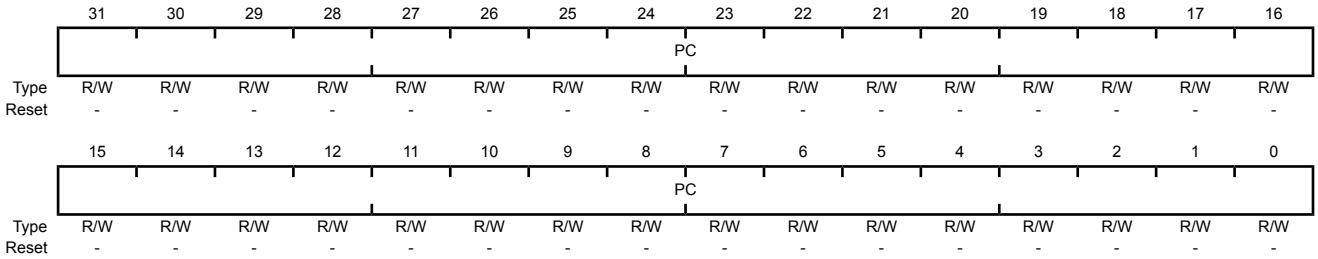

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LINK | R/W | 0xFFFF.FFFF | This field is the return address. |

### 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 R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PC | R/W | - | 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,
- **Execution Program Status Register (EPSR)**, bits 26:24, 15:10
- **Interrupt Program Status Register (IPSR)**, bits 6:0

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 81).

**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 59 shows the possible register combinations for the **PSR**. See the `MRS` and `MSR` instruction descriptions in the *Cortex™-M3 Instruction Set Technical User's Manual* for more information about how to access the program status registers.

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

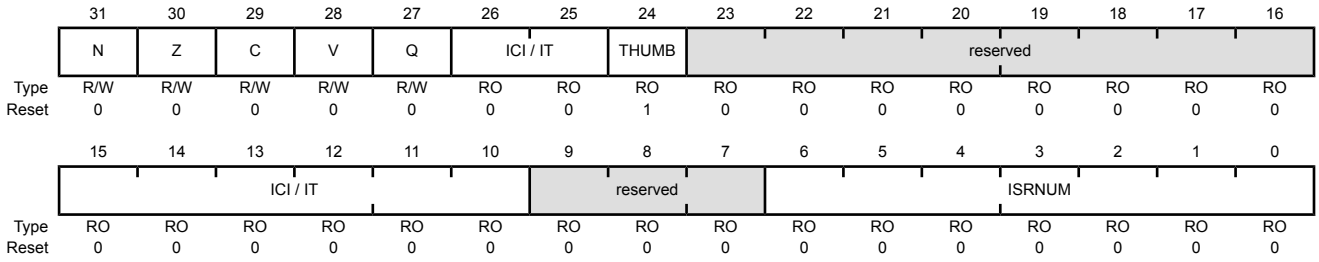
| Register | Type | Combination |
|---|---|---|
| **PSR** | R/W[a, b] | **APSR**, **EPSR**, and **IPSR** |
| **IEPSR** | RO | **EPSR** and **IPSR** |
| **IAPSR** | R/W[a] | **APSR** and **IPSR** |
| **EAPSR** | R/W[b] | **APSR** and **EPSR** |

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 R/W, reset 0x0100.0000



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | N | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 0 | **APSR** DSP Overflow and Saturation Flag<br><br>Value Description<br>1 DSP Overflow or saturation has occurred.<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 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™-M3 Instruction Set Technical User's Manual* for more information.<br>The value of this field is only meaningful when accessing **PSR** or **EPSR**. |
| 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<br>Attempting to execute instructions when this bit is clear results in a fault or lockup. See "Lockup" on page 85 for more information.<br>The value of this bit is only meaningful when accessing **PSR** or **EPSR**. |
| 23:16 | 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. |
| 15:10 | 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 or POP 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™-M3 Instruction Set Technical User's Manual* for more information.<br>The value of this field is only meaningful when accessing **PSR** or **EPSR**. |
| 9:7 | 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 |
|---|---|---|---|---|
| 6:0 | ISRNUM | RO | 0x00 | **IPSR** ISR Number<br>This field contains the exception type number of the current Interrupt Service Routine (ISR).<br><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>0x46 — Interrupt Vector 54<br>0x47-0x7F — Reserved<br><br>See “Exception Types” on page 76 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™-M3 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 76.

Priority Mask Register (PRIMASK)

Type R/W, 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 | | | | | | | | | | | | | | | PRIMASK |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| 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 | 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 | R/W | 0 | Priority Mask |

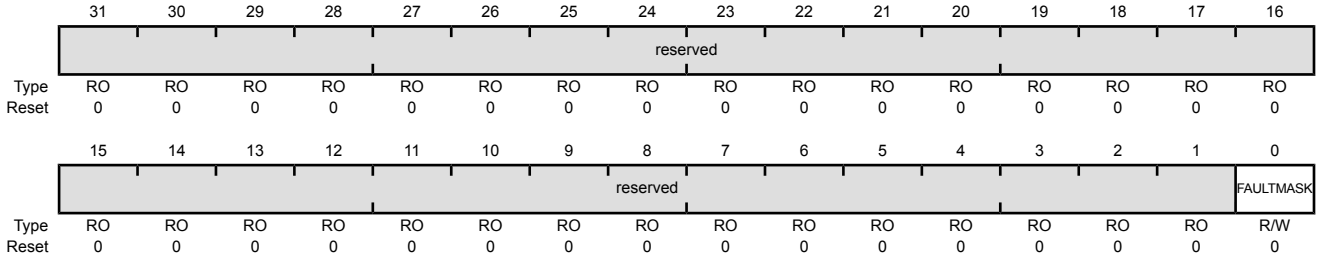
| Value | Description |
|---|---|
| 1 | Prevents the activation of all exceptions with configurable priority. |
| 0 | No effect. |

### 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™-M3 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 76.

Fault Mask Register (FAULTMASK)

Type R/W, reset 0x0000.0000



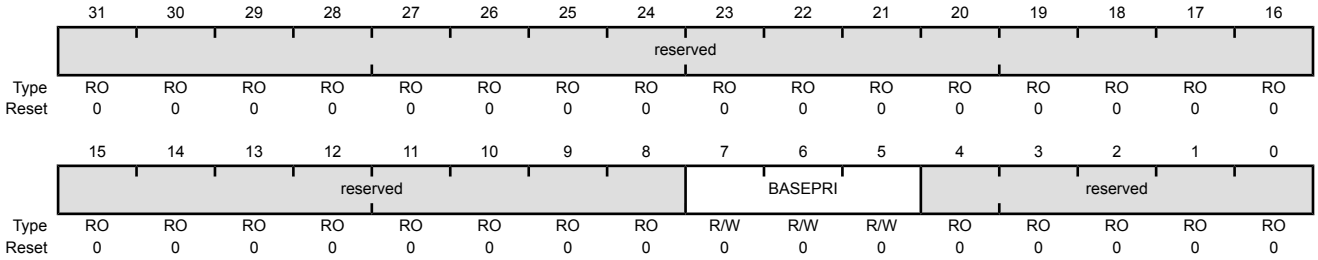

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | FAULTMASK | R/W | 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 76.

Base Priority Mask Register (BASEPRI)

Type R/W, 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 | R/W | 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)

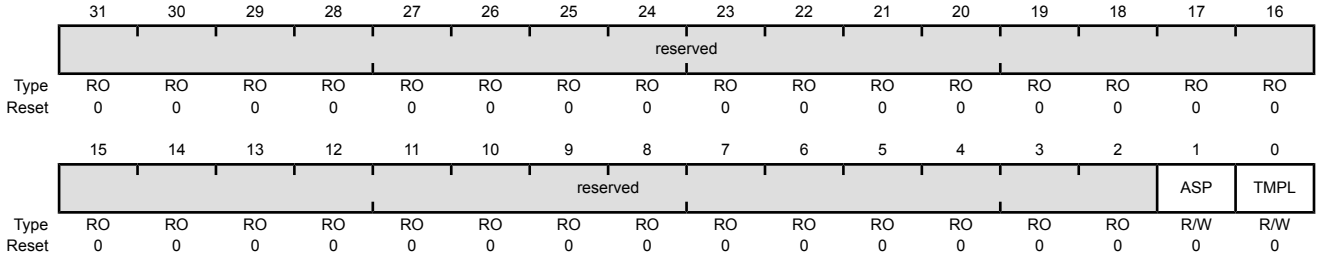
The **CONTROL** register controls the stack used and the privilege level for software execution when the processor is in Thread mode. This register is only accessible in privileged mode.

Handler mode always uses **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 83). 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 **MSP**. To switch the stack pointer used in Thread mode to **PSP**, either use the MSR instruction to set the ASP bit, as detailed in the *Cortex™-M3 Instruction Set Technical User's Manual*, or perform an exception return to Thread mode with the appropriate EXC_RETURN value, as shown in Table 2-10 on page 83.

**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™-M3 Instruction Set Technical User's Manual*.

Control Register (CONTROL)

Type R/W, 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 | ASP | R/W | 0 | Active Stack Pointer<br><br>Value Description<br>1 **PSP** is the current stack pointer.<br>0 **MSP** is the current stack pointer<br><br>In Handler mode, this bit reads as zero and ignores writes. The Cortex-M3 updates this bit automatically on exception return. |
| 0 | TMPL | R/W | 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. |

### 2.3.5 Exceptions and Interrupts

The Cortex-M3 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 81 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” on page 91 for more information.

### 2.3.6 Data Types

The Cortex-M3 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 69 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 LM3S1G58 controller is provided in Table 2-4 on page 67. 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 71).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see “Cortex-M3 Peripherals” on page 90).

**Note:** Within the memory map, all reserved space returns a bus fault when read or written.

**Table 2-4. Memory Map**

| Start | End | Description | For details, see page ... |
|---|---|---|---|
| **Memory** | | | |
| 0x0000.0000 | 0x0005.FFFF | On-chip Flash | 293 |
| 0x0006.0000 | 0x00FF.FFFF | Reserved | - |
| 0x0100.0000 | 0x1FFF.FFFF | Reserved for ROM | 291 |
| 0x2000.0000 | 0x2000.FFFF | Bit-banded on-chip SRAM | 291 |
| 0x2001.0000 | 0x21FF.FFFF | Reserved | - |
| 0x2200.0000 | 0x221F.FFFF | Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000 | 291 |
| 0x2220.0000 | 0x3FFF.FFFF | Reserved | - |
| **FiRM Peripherals** | | | |
| 0x4000.0000 | 0x4000.0FFF | Watchdog timer 0 | 498 |
| 0x4000.1000 | 0x4000.1FFF | Watchdog timer 1 | 498 |
| 0x4000.2000 | 0x4000.3FFF | Reserved | - |
| 0x4000.4000 | 0x4000.4FFF | GPIO Port A | 407 |
| 0x4000.5000 | 0x4000.5FFF | GPIO Port B | 407 |
| 0x4000.6000 | 0x4000.6FFF | GPIO Port C | 407 |

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

| Start | End | Description | For details, see page ... |
|---|---|---|---|
| 0x4000.7000 | 0x4000.7FFF | GPIO Port D | 407 |
| 0x4000.8000 | 0x4000.8FFF | SSI0 | 673 |
| 0x4000.9000 | 0x4000.9FFF | SSI1 | 673 |
| 0x4000.A000 | 0x4000.BFFF | Reserved | - |
| 0x4000.C000 | 0x4000.CFFF | UART0 | 611 |
| 0x4000.D000 | 0x4000.DFFF | UART1 | 611 |
| 0x4000.E000 | 0x4000.EFFF | UART2 | 611 |
| 0x4000.F000 | 0x4001.FFFF | Reserved | - |
| **Peripherals** | | | |
| 0x4002.0000 | 0x4002.0FFF | $I^2C$ 0 | 717 |
| 0x4002.1000 | 0x4002.1FFF | $I^2C$ 1 | 717 |
| 0x4002.2000 | 0x4002.3FFF | Reserved | - |
| 0x4002.4000 | 0x4002.4FFF | GPIO Port E | 407 |
| 0x4002.5000 | 0x4002.5FFF | GPIO Port F | 407 |
| 0x4002.6000 | 0x4002.6FFF | GPIO Port G | 407 |
| 0x4002.7000 | 0x4002.7FFF | GPIO Port H | 407 |
| 0x4002.8000 | 0x4002.FFFF | Reserved | - |
| 0x4003.0000 | 0x4003.0FFF | Timer 0 | 464 |
| 0x4003.1000 | 0x4003.1FFF | Timer 1 | 464 |
| 0x4003.2000 | 0x4003.2FFF | Timer 2 | 464 |
| 0x4003.3000 | 0x4003.3FFF | Timer 3 | 464 |
| 0x4003.4000 | 0x4003.7FFF | Reserved | - |
| 0x4003.8000 | 0x4003.8FFF | ADC0 | 541 |
| 0x4003.9000 | 0x4003.9FFF | ADC1 | 541 |
| 0x4003.A000 | 0x4005.7FFF | Reserved | - |
| 0x4005.8000 | 0x4005.8FFF | GPIO Port A (AHB aperture) | 407 |
| 0x4005.9000 | 0x4005.9FFF | GPIO Port B (AHB aperture) | 407 |
| 0x4005.A000 | 0x4005.AFFF | GPIO Port C (AHB aperture) | 407 |
| 0x4005.B000 | 0x4005.BFFF | GPIO Port D (AHB aperture) | 407 |
| 0x4005.C000 | 0x4005.CFFF | GPIO Port E (AHB aperture) | 407 |
| 0x4005.D000 | 0x4005.DFFF | GPIO Port F (AHB aperture) | 407 |
| 0x4005.E000 | 0x4005.EFFF | GPIO Port G (AHB aperture) | 407 |
| 0x4005.F000 | 0x4005.FFFF | GPIO Port H (AHB aperture) | 407 |
| 0x4006.0000 | 0x400F.BFFF | Reserved | - |
| 0x400F.C000 | 0x400F.CFFF | Hibernation Module | 273 |
| 0x400F.D000 | 0x400F.DFFF | Flash memory control | 299 |
| 0x400F.E000 | 0x400F.EFFF | System control | 186 |
| 0x400F.F000 | 0x400F.FFFF | µDMA | 356 |
| 0x4010.0000 | 0x41FF.FFFF | Reserved | - |
| 0x4200.0000 | 0x43FF.FFFF | Bit-banded alias of 0x4000.0000 through 0x400F.FFFF | - |
| 0x4400.0000 | 0xDFFF.FFFF | Reserved | - |

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

| Start | End | Description | For details, see page ... |
|---|---|---|---|
| **Private Peripheral Bus** | | | |
| 0xE000.0000 | 0xE000.0FFF | Instrumentation Trace Macrocell (ITM) | 50 |
| 0xE000.1000 | 0xE000.1FFF | Data Watchpoint and Trace (DWT) | 50 |
| 0xE000.2000 | 0xE000.2FFF | Flash Patch and Breakpoint (FPB) | 50 |
| 0xE000.3000 | 0xE000.DFFF | Reserved | - |
| 0xE000.E000 | 0xE000.EFFF | Cortex-M3 Peripherals (SysTick, NVIC, SCB, and MPU) | 75 |
| 0xE000.F000 | 0xE003.FFFF | Reserved | - |
| 0xE004.0000 | 0xE004.0FFF | Trace Port Interface Unit (TPIU) | 51 |
| 0xE004.1000 | 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 70).

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 70 shows the behavior of accesses to each region in the memory map. See “Memory Regions, Types and Attributes” on page 69 for more information on memory types and

the XN attribute. Stellaris devices may have reserved memory areas within the address ranges shown below (refer to Table 2-4 on page 67 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 72). |
| 0x4000.0000 - 0x5FFF.FFFF | Peripheral | Device | XN | This region includes bit band and bit band alias areas (see Table 2-7 on page 72). |
| 0x6000.0000 - 0x9FFF.FFFF | External RAM | Normal | - | This executable region is for data. |
| 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-M3 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 93.

The Cortex-M3 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 69 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-M3 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™-M3 Instruction Set Technical User's Manual*.

### 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 72. 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 72. For the specific address range of the bit-band regions, see Table 2-4 on page 67.

**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 |
|---|---|---|
| 0x2000.0000 - 0x200F.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. |
| 0x2200.0000 - 0x23FF.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 |
|---|---|---|
| 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 73 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)`

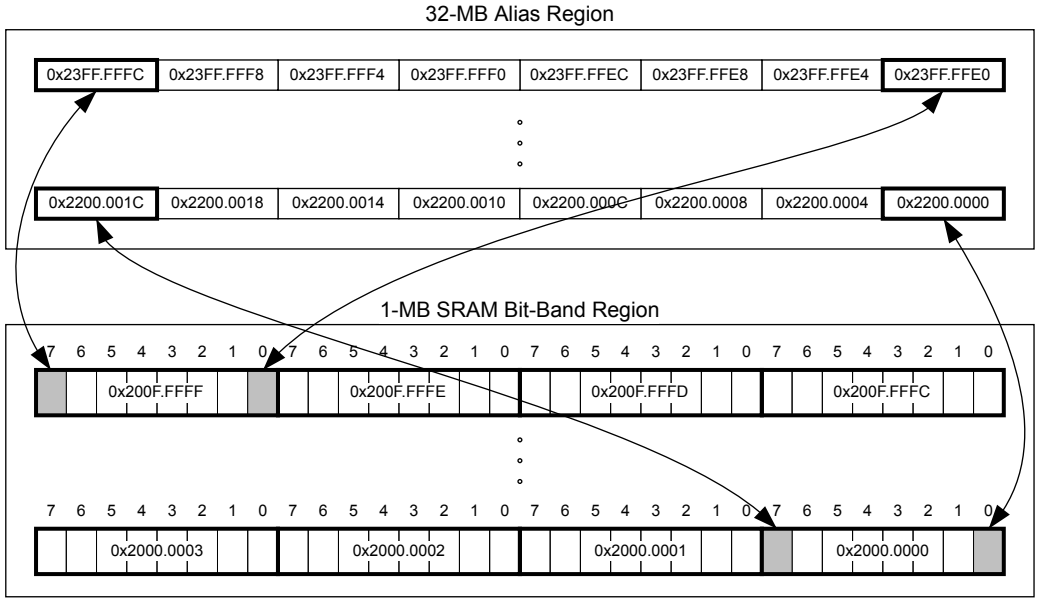
- The alias word at 0x2200.0000 maps to bit 0 of the bit-band byte at 0x2000.0000:

  `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.

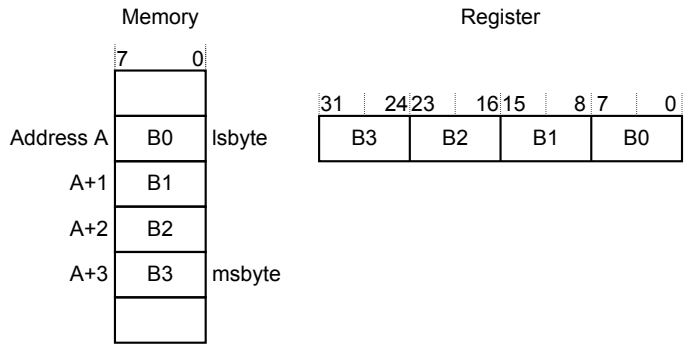
### 2.4.5.2 Directly Accessing a Bit-Band Region

“Behavior of Memory Accesses” on page 69 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 74 illustrates how data is stored.

**Figure 2-5. Data Storage**




## 2.4.7 Synchronization Primitives

The Cortex-M3 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.

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 is 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 a guaranteed read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Update the value, as required.

3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location, and 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 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-M3 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™-M3 Instruction Set Technical User's Manual*.

## 2.5 Exception Model

The ARM Cortex-M3 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 77 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 37 interrupts (listed in Table 2-9 on page 78).

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 91.

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 de-assert. 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 91 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 78 lists the interrupts on the LM3S1G58 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 77 shows as having configurable priority (see the **SYSHNDCTRL** register on page 134 and the **DIS0** register on page 107).

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” on page 83.

**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 79.

c. See **SYSPRI1** on page 131.

d. See **PRIn** registers on page 115.

**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 | $I^2C0$ |
| 25-29 | 9-13 | - | Reserved |
| 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 | Timer 0A |
| 36 | 20 | 0x0000.0090 | Timer 0B |
| 37 | 21 | 0x0000.0094 | Timer 1A |
| 38 | 22 | 0x0000.0098 | Timer 1B |
| 39 | 23 | 0x0000.009C | Timer 2A |
| 40 | 24 | 0x0000.00A0 | Timer 2B |
| 41-43 | 25-27 | - | Reserved |

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

| Vector Number | Interrupt Number (Bit in Interrupt Registers) | Vector Address or Offset | Description |
|---|---|---|---|
| 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 | Timer 3A |
| 52 | 36 | 0x0000.00D0 | Timer 3B |
| 53 | 37 | 0x0000.00D4 | $I^2C1$ |
| 54-58 | 38-42 | - | Reserved |
| 59 | 43 | 0x0000.00EC | Hibernation Module |
| 60-61 | 44-45 | - | Reserved |
| 62 | 46 | 0x0000.00F8 | µDMA Software |
| 63 | 47 | 0x0000.00FC | µDMA Error |
| 64 | 48 | 0x0000.0100 | ADC1 Sequence 0 |
| 65 | 49 | 0x0000.0104 | ADC1 Sequence 1 |
| 66 | 50 | 0x0000.0108 | ADC1 Sequence 2 |
| 67 | 51 | 0x0000.010C | ADC1 Sequence 3 |
| 68-70 | 52-54 | - | Reserved |

### 2.5.3 Exception Handlers

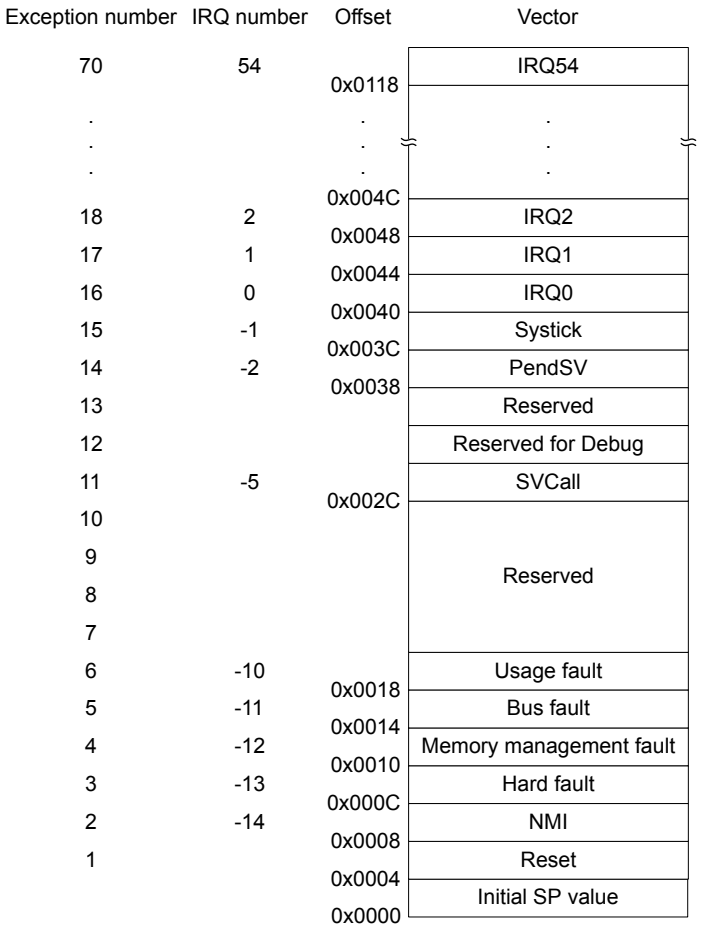
The processor handles exceptions using:

- **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 77. Figure 2-6 on page 80 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 0x0000.0200 to 0x3FFF.FE00 (see "Vector Table" on page 79). Note that when configuring the **VTABLE** register, the offset must be aligned on a 512-byte boundary.

## 2.5.5 Exception Priorities

As Table 2-8 on page 77 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 131 and page 115.

**Note:** Configurable priority values for the Stellaris 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 125.

### 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 81 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” on page 82 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 82 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

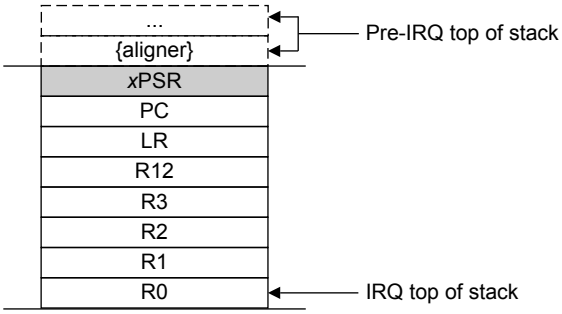
Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

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 63, **FAULTMASK** on page 64, and **BASEPRI** on page 65). 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*.

**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 to 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 four bits of this value provide information on the return stack and processor mode. Table 2-10 on page 83 shows the EXC_RETURN values with a description of the exception return behavior.

EXC_RETURN bits 31:4 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.FFF0 | Reserved |
| 0xFFFF.FFF1 | Return to Handler mode.<br>Exception return uses state from **MSP**.<br>Execution uses **MSP** after return. |
| 0xFFFF.FFF2 - 0xFFFF.FFF8 | Reserved |
| 0xFFFF.FFF9 | Return to Thread mode.<br>Exception return uses state from **MSP**.<br>Execution uses **MSP** after return. |
| 0xFFFF.FFFA - 0xFFFF.FFFC | Reserved |
| 0xFFFF.FFFD | Return to Thread mode.<br>Exception return uses 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 75). 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 83 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 138 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 |

**Table 2-11. Faults *(continued)***

| Fault | Handler | Fault Status Register | Bit Name |
|---|---|---|---|
| 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 |
| 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 |
| 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-multiple 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 131). Software can disable execution of the handlers for these faults (see **SYSHNDCTRL** on page 134).

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 75.

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 85.

**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 144 |
| Memory management fault | **Memory Management Fault Status (MFAULTSTAT)** | **Memory Management Fault Address (MMADDR)** | page 138<br>page 145 |
| Bus fault | **Bus Fault Status (BFAULTSTAT)** | **Bus Fault Address (FAULTADDR)** | page 138<br>page 146 |
| Usage fault | **Usage Fault Status (UFAULTSTAT)** | - | page 138 |

### 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 or an NMI occurs.

**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-M3 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 127). For more information about the behavior of the sleep modes, see “System Control” on page 182.

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 86). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the *Cortex™-M3 Instruction Set Technical User's Manual* 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™-M3 Instruction Set Technical User's Manual* 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 an exception handler, 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 cause it to enter sleep mode.

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

Normally, the processor wakes up only when it 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 63 and page 64.

#### 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 127.

## 2.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 2-13 on page 87 lists the supported instructions.

**Note:** In Table 2-13 on page 87:

- 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 *Cortex™-M3 Instruction Set Technical User's Manual*.

**Table 2-13. Cortex-M3 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 | N,Z,C,V |
| 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 | - |
| 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 | - |

**Table 2-13. Cortex-M3 Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| 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 |
| ORR, ORRS | {Rd,} Rn, Op2 | Logical OR | N,Z,C |
| POP | reglist | Pop registers from stack | - |
| PUSH | reglist | Push registers onto stack | - |
| 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 |
| SBC, SBCS | {Rd,} Rn, Op2 | Subtract with carry | N,Z,C,V |
| SBFX | Rd, Rn, #lsb, #width | Signed bit field extract | - |

**Table 2-13. Cortex-M3 Instruction Summary *(continued)***

| Mnemonic | Operands | Brief Description | Flags |
|---|---|---|---|
| SDIV | {Rd,} Rn, Rm | Signed divide | - |
| SEV | - | Send event | - |
| SMLAL | RdLo, RdHi, Rn, Rm | Signed multiply with accumulate (32x32+64), 64-bit result | - |
| SMULL | RdLo, RdHi, Rn, Rm | Signed multiply (32x32), 64-bit result | - |
| SSAT | Rd, #n, Rm {,shift #s} | Signed saturate | Q |
| STM | Rn{!}, reglist | Store multiple registers, increment after | - |
| 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 | - |
| 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 |
| UBFX | Rd, Rn, #lsb, #width | Unsigned bit field extract | - |
| UDIV | {Rd,} Rn, Rm | Unsigned divide | - |
| 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 | - |
| USAT | Rd, #n, Rm {,shift #s} | Unsigned Saturate | Q |
| UXTB | {Rd,} Rm, {,ROR #n} | Zero extend a Byte | - |
| UXTH | {Rd,} Rm, {,ROR #n} | Zero extend a Halfword | - |
| USAT | Rd, #n, Rm {,shift #s} | Unsigned saturate | Q |
| UXTB | {Rd,} Rm {,ROR #n} | Zero extend a byte | - |
| UXTH | {Rd,} Rm {,ROR #n} | Zero extend a halfword | - |
| WFE | - | Wait for event | - |
| WFI | - | Wait for interrupt | - |

# 3 Cortex-M3 Peripherals

This chapter provides information on the Stellaris® implementation of the Cortex-M3 processor peripherals, including:

- SysTick (see page 90)

  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 91)
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers

- System Control Block (SCB) (see page 93)

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

- Memory Protection Unit (MPU) (see page 93)

  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.

Table 3-1 on page 90 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 | 90 |
| 0xE000.E100-0xE000.E4EF<br>0xE000.EF00-0xE000.EF03 | Nested Vectored Interrupt Controller | 91 |
| 0xE000.E008-0xE000.E00F<br>0xE000.ED00-0xE000.ED3F | System Control Block | 93 |
| 0xE000.ED90-0xE000.EDB8 | Memory Protection Unit | 93 |

## 3.1 Functional Description

This chapter provides information on the Stellaris implementation of the Cortex-M3 processor peripherals: SysTick, NVIC, SCB and MPU.

### 3.1.1 System Timer (SysTick)

Cortex-M3 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.

**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:

- 37 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 92 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-M3 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 109 or **SWTRIG** on page 117.

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-M3 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-M3 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 69 for more information).

Table 3-2 on page 93 shows the possible MPU region attributes. See the section called “MPU Configuration for a Stellaris Microcontroller” on page 97 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 pre-packed information, meaning that the **MPU Region Base Address (MPUBASE)** register (see page 151) 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
```

An STM instruction can be used to optimize this:

```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0,=MPUBASE     ; 0xE000ED9C, MPU Region Base register
STM R0, {R1-R2}     ; Region base address, region number and VALID bit,
                    ; and 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 153) 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

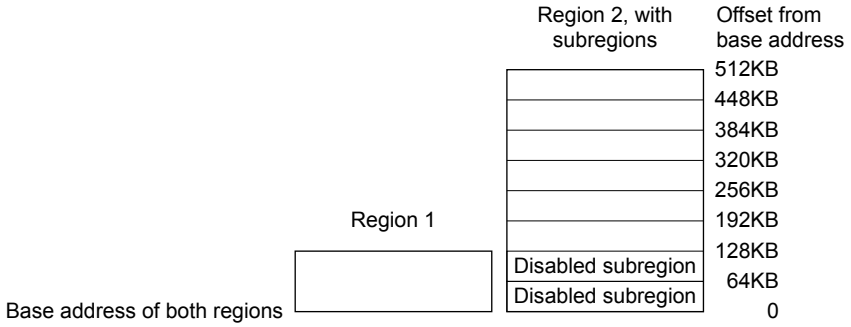
overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

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 96 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 96 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-M3 does not support the concept of cacheability or shareability. Refer to the section called “MPU Configuration for a Stellaris Microcontroller” on page 97 for information on programming the MPU for Stellaris 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 | - |
| 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 noncacheable. |
| 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 | |

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

| TEX | S | C | B | Memory Type | Shareability | Other Attributes |
|---|---|---|---|---|---|---|
| 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 97 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 97 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 | R/W | No access | Access from privileged software only. |
| 010 | R/W | RO | Writes by unprivileged software generate a permission fault. |
| 011 | R/W | R/W | Full access. |
| 100 | Unpredictable | Unpredictable | Reserved. |
| 101 | RO | No access | Reads by privileged software only. |
| 110 | RO | RO | Read-only, by privileged or unprivileged software. |
| 111 | RO | RO | Read-only, by privileged or unprivileged software. |

### *MPU Configuration for a Stellaris Microcontroller*

Stellaris 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 97.

**Table 3-6. Memory Region Attributes for Stellaris 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 |

**Table 3-6. Memory Region Attributes for Stellaris Microcontrollers *(continued)***

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

In current Stellaris 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 67 for more information). The **MFAULTSTAT** register indicates the cause of the fault. See page 138 for more information.

## 3.2 Register Map

Table 3-7 on page 98 lists the Cortex-M3 Peripheral SysTick, NVIC, SCB, and MPU 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 3-7. Peripherals Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **System Timer (SysTick) Registers** | | | | | |
| 0x010 | STCTRL | R/W | 0x0000.0004 | SysTick Control and Status Register | 101 |
| 0x014 | STRELOAD | R/W | 0x0000.0000 | SysTick Reload Value Register | 103 |
| 0x018 | STCURRENT | R/WC | 0x0000.0000 | SysTick Current Value Register | 104 |
| **Nested Vectored Interrupt Controller (NVIC) Registers** | | | | | |
| 0x100 | EN0 | R/W | 0x0000.0000 | Interrupt 0-31 Set Enable | 105 |
| 0x104 | EN1 | R/W | 0x0000.0000 | Interrupt 32-54 Set Enable | 106 |
| 0x180 | DIS0 | R/W | 0x0000.0000 | Interrupt 0-31 Clear Enable | 107 |
| 0x184 | DIS1 | R/W | 0x0000.0000 | Interrupt 32-54 Clear Enable | 108 |
| 0x200 | PEND0 | R/W | 0x0000.0000 | Interrupt 0-31 Set Pending | 109 |
| 0x204 | PEND1 | R/W | 0x0000.0000 | Interrupt 32-54 Set Pending | 110 |
| 0x280 | UNPEND0 | R/W | 0x0000.0000 | Interrupt 0-31 Clear Pending | 111 |
| 0x284 | UNPEND1 | R/W | 0x0000.0000 | Interrupt 32-54 Clear Pending | 112 |
| 0x300 | ACTIVE0 | RO | 0x0000.0000 | Interrupt 0-31 Active Bit | 113 |
| 0x304 | ACTIVE1 | RO | 0x0000.0000 | Interrupt 32-54 Active Bit | 114 |
| 0x400 | PRI0 | R/W | 0x0000.0000 | Interrupt 0-3 Priority | 115 |

**Table 3-7. Peripherals Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x404 | PRI1 | R/W | 0x0000.0000 | Interrupt 4-7 Priority | 115 |
| 0x408 | PRI2 | R/W | 0x0000.0000 | Interrupt 8-11 Priority | 115 |
| 0x40C | PRI3 | R/W | 0x0000.0000 | Interrupt 12-15 Priority | 115 |
| 0x410 | PRI4 | R/W | 0x0000.0000 | Interrupt 16-19 Priority | 115 |
| 0x414 | PRI5 | R/W | 0x0000.0000 | Interrupt 20-23 Priority | 115 |
| 0x418 | PRI6 | R/W | 0x0000.0000 | Interrupt 24-27 Priority | 115 |
| 0x41C | PRI7 | R/W | 0x0000.0000 | Interrupt 28-31 Priority | 115 |
| 0x420 | PRI8 | R/W | 0x0000.0000 | Interrupt 32-35 Priority | 115 |
| 0x424 | PRI9 | R/W | 0x0000.0000 | Interrupt 36-39 Priority | 115 |
| 0x428 | PRI10 | R/W | 0x0000.0000 | Interrupt 40-43 Priority | 115 |
| 0x42C | PRI11 | R/W | 0x0000.0000 | Interrupt 44-47 Priority | 115 |
| 0x430 | PRI12 | R/W | 0x0000.0000 | Interrupt 48-51 Priority | 115 |
| 0x434 | PRI13 | R/W | 0x0000.0000 | Interrupt 52-54 Priority | 115 |
| 0xF00 | SWTRIG | WO | 0x0000.0000 | Software Trigger Interrupt | 117 |
| **System Control Block (SCB) Registers** | | | | | |
| 0x008 | ACTLR | R/W | 0x0000.0000 | Auxiliary Control | 118 |
| 0xD00 | CPUID | RO | 0x412F.C230 | CPU ID Base | 120 |
| 0xD04 | INTCTRL | R/W | 0x0000.0000 | Interrupt Control and State | 121 |
| 0xD08 | VTABLE | R/W | 0x0000.0000 | Vector Table Offset | 124 |
| 0xD0C | APINT | R/W | 0xFA05.0000 | Application Interrupt and Reset Control | 125 |
| 0xD10 | SYSCTRL | R/W | 0x0000.0000 | System Control | 127 |
| 0xD14 | CFGCTRL | R/W | 0x0000.0200 | Configuration and Control | 129 |
| 0xD18 | SYSPRI1 | R/W | 0x0000.0000 | System Handler Priority 1 | 131 |
| 0xD1C | SYSPRI2 | R/W | 0x0000.0000 | System Handler Priority 2 | 132 |
| 0xD20 | SYSPRI3 | R/W | 0x0000.0000 | System Handler Priority 3 | 133 |
| 0xD24 | SYSHNDCTRL | R/W | 0x0000.0000 | System Handler Control and State | 134 |
| 0xD28 | FAULTSTAT | R/W1C | 0x0000.0000 | Configurable Fault Status | 138 |
| 0xD2C | HFAULTSTAT | R/W1C | 0x0000.0000 | Hard Fault Status | 144 |
| 0xD34 | MMADDR | R/W | - | Memory Management Fault Address | 145 |
| 0xD38 | FAULTADDR | R/W | - | Bus Fault Address | 146 |
| **Memory Protection Unit (MPU) Registers** | | | | | |
| 0xD90 | MPUTYPE | RO | 0x0000.0800 | MPU Type | 147 |

**Table 3-7. Peripherals Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xD94 | MPUCTRL | R/W | 0x0000.0000 | MPU Control | 148 |
| 0xD98 | MPUNUMBER | R/W | 0x0000.0000 | MPU Region Number | 150 |
| 0xD9C | MPUBASE | R/W | 0x0000.0000 | MPU Region Base Address | 151 |
| 0xDA0 | MPUATTR | R/W | 0x0000.0000 | MPU Region Attribute and Size | 153 |
| 0xDA4 | MPUBASE1 | R/W | 0x0000.0000 | MPU Region Base Address Alias 1 | 151 |
| 0xDA8 | MPUATTR1 | R/W | 0x0000.0000 | MPU Region Attribute and Size Alias 1 | 153 |
| 0xDAC | MPUBASE2 | R/W | 0x0000.0000 | MPU Region Base Address Alias 2 | 151 |
| 0xDB0 | MPUATTR2 | R/W | 0x0000.0000 | MPU Region Attribute and Size Alias 2 | 153 |
| 0xDB4 | MPUBASE3 | R/W | 0x0000.0000 | MPU Region Base Address Alias 3 | 151 |
| 0xDB8 | MPUATTR3 | R/W | 0x0000.0000 | MPU Region Attribute and Size Alias 3 | 153 |

## 3.3 System Timer (SysTick) Register Descriptions

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

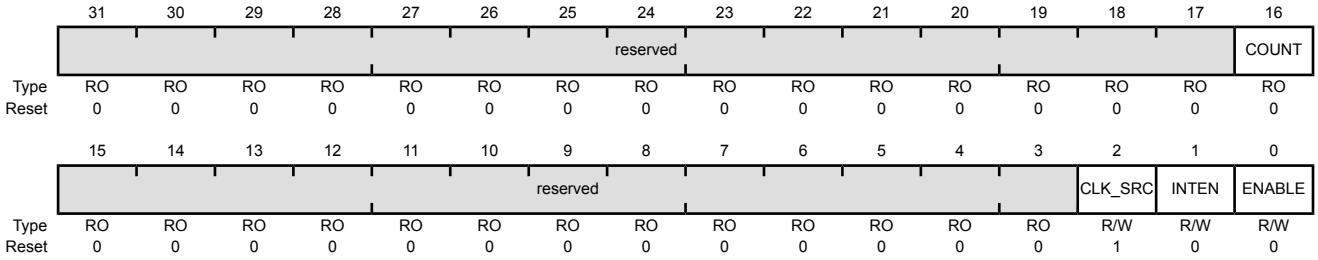
## Register 1: SysTick Control and Status Register (STCTRL), offset 0x010

**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 R/W, reset 0x0000.0004




| 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 | R/W | 1 | Clock Source<br><br>Value Description<br>0 External reference clock. (Not implemented for Stellaris microcontrollers.)<br>1 System clock<br><br>Because an external reference clock is not implemented, this bit must be set in order for SysTick to operate. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INTEN | R/W | 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 | R/W | 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

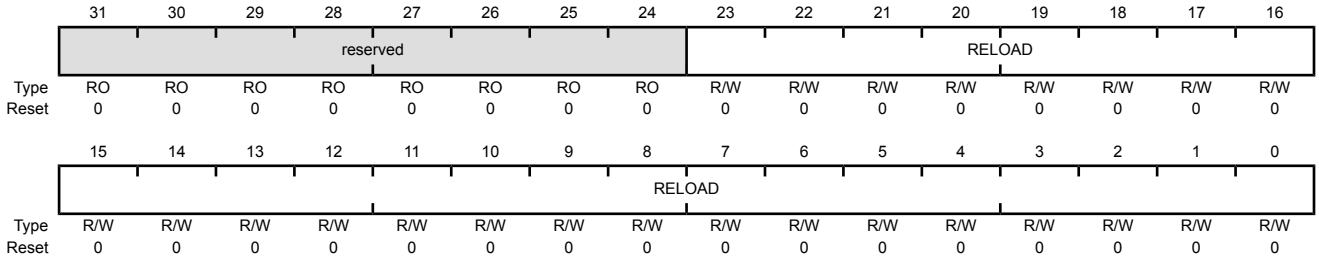
**Note:** This register can only be accessed from privileged mode.

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.

SysTick Reload Value Register (STRELOAD)

Base 0xE000.E000
Offset 0x014
Type R/W, reset 0x0000.0000



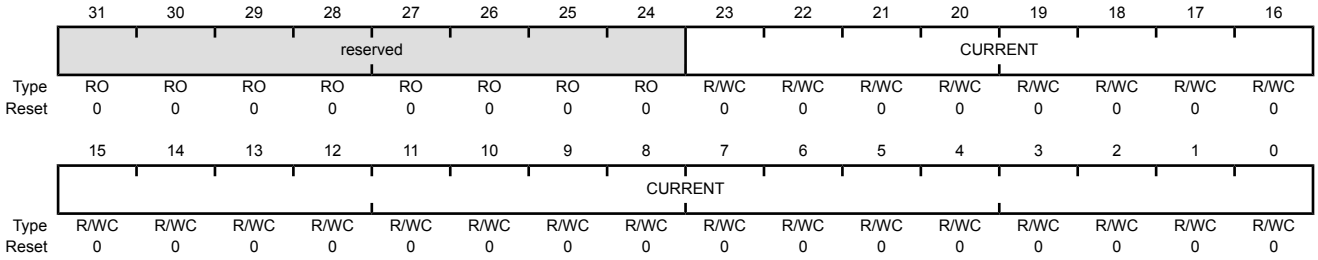

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 23:0 | RELOAD | R/W | 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 R/WC, 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. |
| 23:0 | CURRENT | R/WC | 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 124.

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

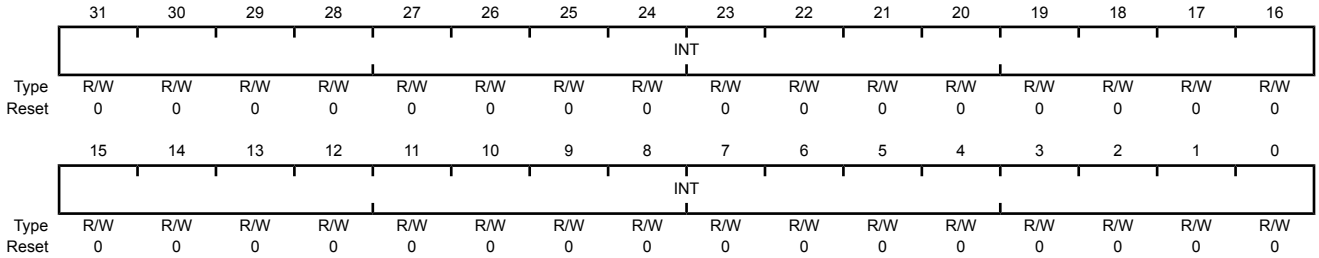
**Note:** This register can only be accessed from privileged mode.

See Table 2-9 on page 78 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | R/W | 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 5: Interrupt 32-54 Set Enable (EN1), offset 0x104

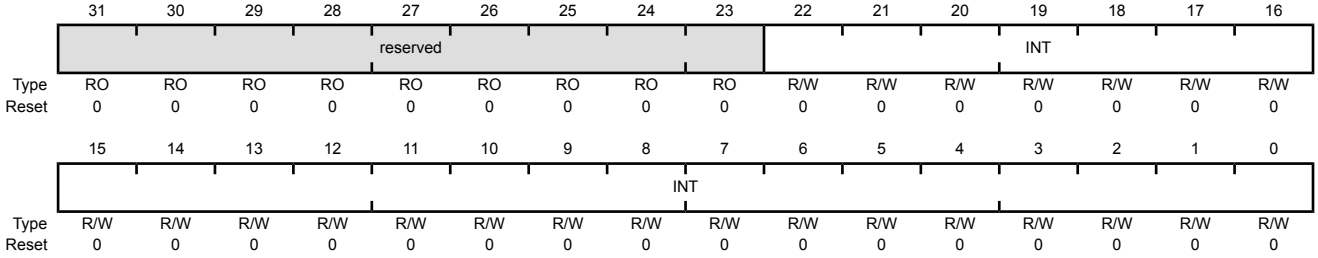
**Note:** This register can only be accessed from privileged mode.

The **EN1** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 78 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 32-54 Set Enable (EN1)

Base 0xE000.E000
Offset 0x104
Type R/W, reset 0x0000.0000




| 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. |
| 22:0 | INT | R/W | 0x00.0000 | 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 **DIS1** register.

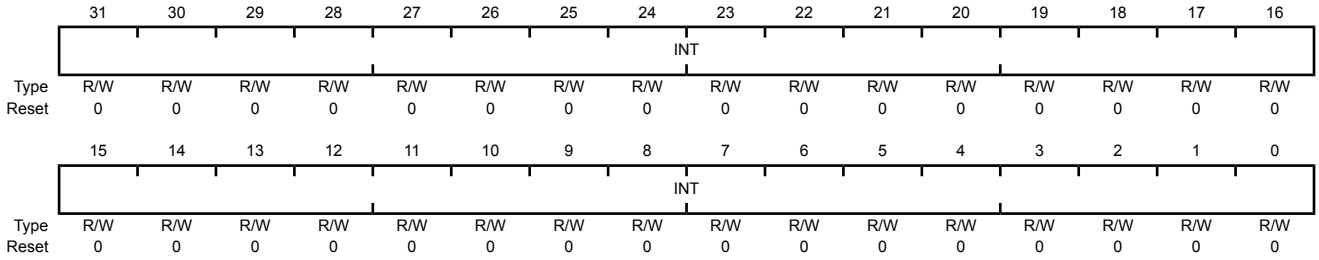
## Register 6: Interrupt 0-31 Clear Enable (DIS0), offset 0x180

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

See Table 2-9 on page 78 for interrupt assignments.

Interrupt 0-31 Clear Enable (DIS0)

Base 0xE000.E000
Offset 0x180
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | R/W | 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]. |

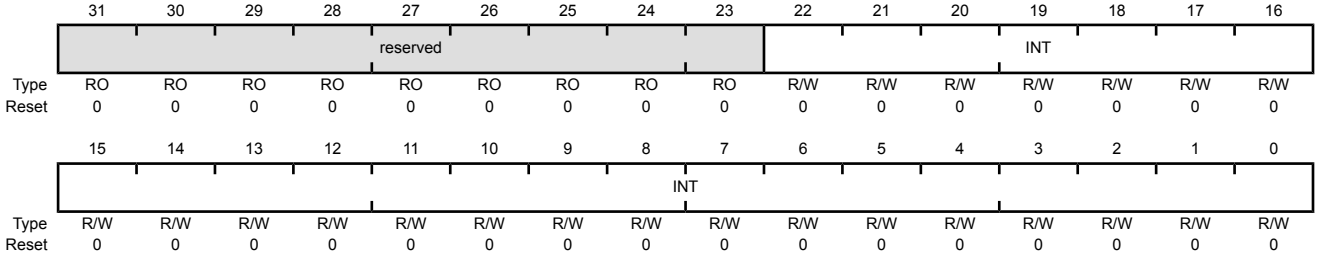
## Register 7: Interrupt 32-54 Clear Enable (DIS1), offset 0x184

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

The **DIS1** register disables interrupts. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 78 for interrupt assignments.

Interrupt 32-54 Clear Enable (DIS1)

Base 0xE000.E000
Offset 0x184
Type R/W, reset 0x0000.0000




| 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. |
| 22:0 | INT | R/W | 0x00.0000 | 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 **EN1** register, disabling interrupt [n]. |

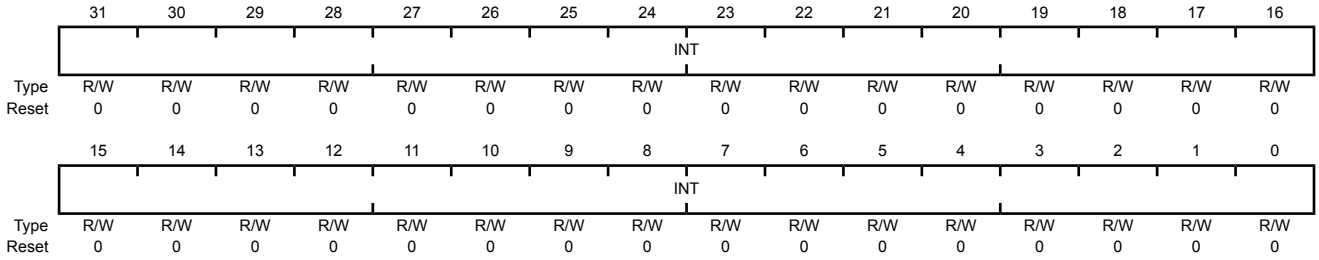
## Register 8: Interrupt 0-31 Set Pending (PEND0), offset 0x200

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

See Table 2-9 on page 78 for interrupt assignments.

Interrupt 0-31 Set Pending (PEND0)

Base 0xE000.E000
Offset 0x200
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | R/W | 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.

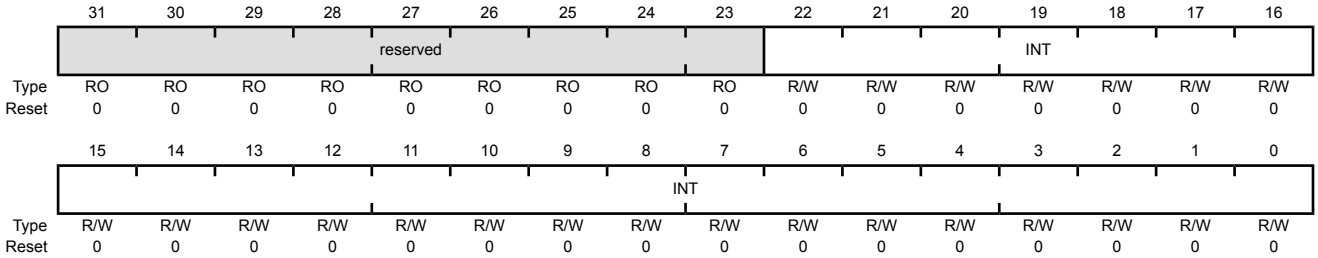
## Register 9: Interrupt 32-54 Set Pending (PEND1), offset 0x204

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

The **PEND1** register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 78 for interrupt assignments.

Interrupt 32-54 Set Pending (PEND1)

Base 0xE000.E000
Offset 0x204
Type R/W, reset 0x0000.0000




| 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. |
| 22:0 | INT | R/W | 0x00.0000 | 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 **UNPEND1** register.

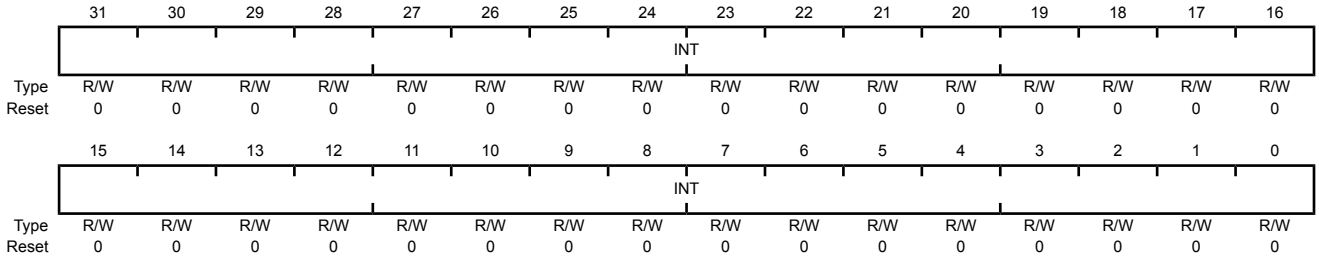
## Register 10: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280

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

See Table 2-9 on page 78 for interrupt assignments.

Interrupt 0-31 Clear Pending (UNPEND0)

Base 0xE000.E000
Offset 0x280
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | INT | R/W | 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. |

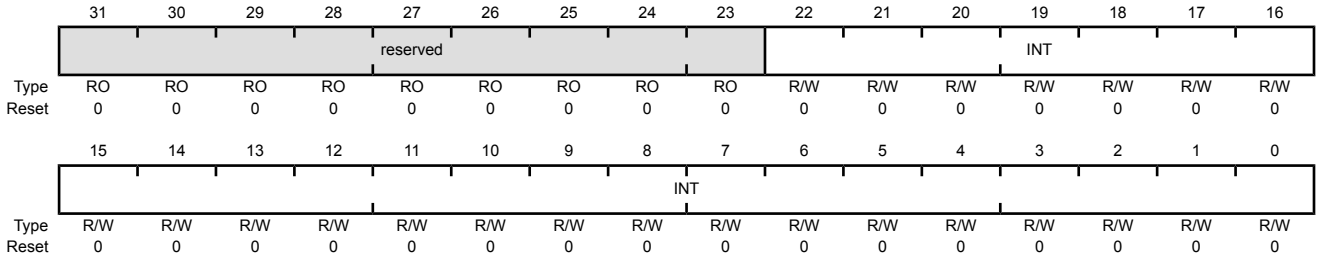
## Register 11: Interrupt 32-54 Clear Pending (UNPEND1), offset 0x284

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

The **UNPEND1** register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 78 for interrupt assignments.

Interrupt 32-54 Clear Pending (UNPEND1)

Base 0xE000.E000
Offset 0x284
Type R/W, reset 0x0000.0000




| 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. |
| 22:0 | INT | R/W | 0x00.0000 | 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 **PEND1** register, so that interrupt [n] is no longer pending.<br>Setting a bit does not affect the active state of the corresponding interrupt. |

## Register 12: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300

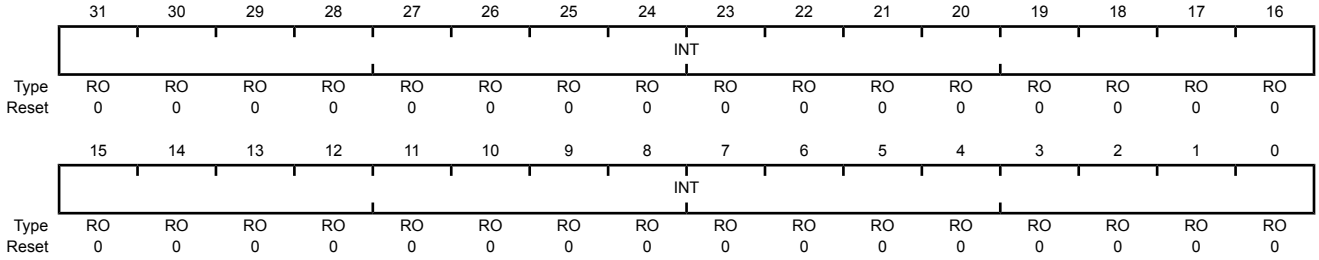
**Note:** This register can only be accessed from privileged mode.

See Table 2-9 on page 78 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 13: Interrupt 32-54 Active Bit (ACTIVE1), offset 0x304

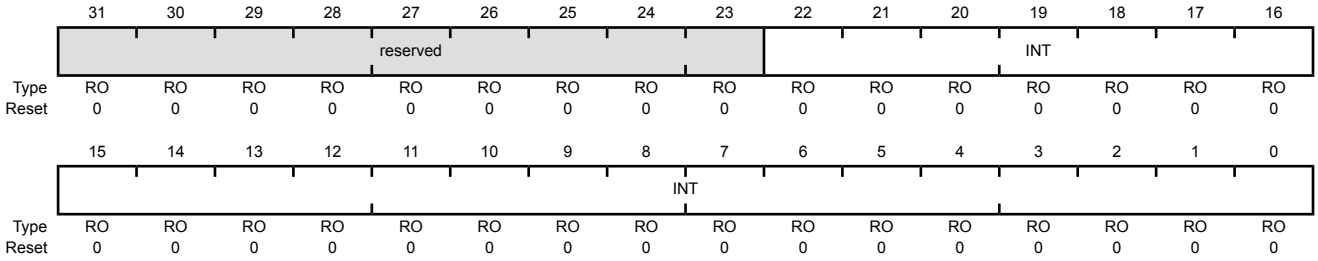
**Note:** This register can only be accessed from privileged mode.

The **ACTIVE1** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 78 for interrupt assignments.

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

Interrupt 32-54 Active Bit (ACTIVE1)

Base 0xE000.E000
Offset 0x304
Type RO, reset 0x0000.0000




| 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. |
| 22:0 | INT | RO | 0x00.0000 | Interrupt Active |

| Value | Description |
|---|---|
| 0 | The corresponding interrupt is not active. |
| 1 | The corresponding interrupt is active, or active and pending. |

### Register 14: Interrupt 0-3 Priority (PRI0), offset 0x400

### Register 15: Interrupt 4-7 Priority (PRI1), offset 0x404

### Register 16: Interrupt 8-11 Priority (PRI2), offset 0x408

### Register 17: Interrupt 12-15 Priority (PRI3), offset 0x40C

### Register 18: Interrupt 16-19 Priority (PRI4), offset 0x410

### Register 19: Interrupt 20-23 Priority (PRI5), offset 0x414

### Register 20: Interrupt 24-27 Priority (PRI6), offset 0x418

### Register 21: Interrupt 28-31 Priority (PRI7), offset 0x41C

### Register 22: Interrupt 32-35 Priority (PRI8), offset 0x420

### Register 23: Interrupt 36-39 Priority (PRI9), offset 0x424

### Register 24: Interrupt 40-43 Priority (PRI10), offset 0x428

### Register 25: Interrupt 44-47 Priority (PRI11), offset 0x42C

### Register 26: Interrupt 48-51 Priority (PRI12), offset 0x430

### Register 27: Interrupt 52-54 Priority (PRI13), offset 0x434

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

The **PRIn** registers 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] |

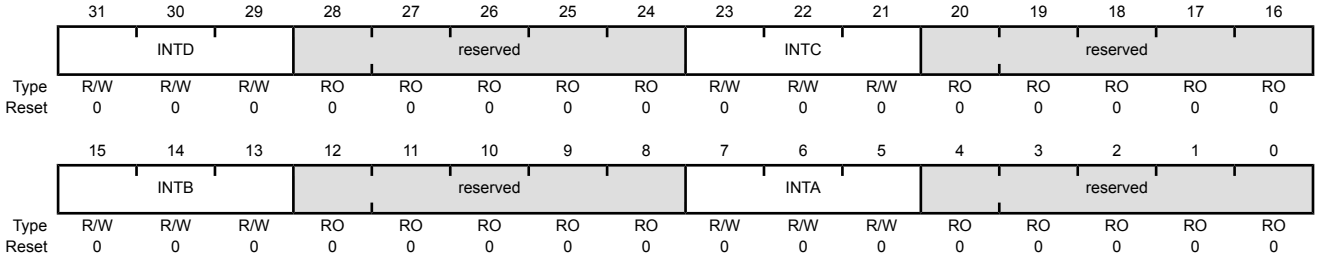
See Table 2-9 on page 78 for interrupt assignments.

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 125) 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | INTD | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 28: Software Trigger Interrupt (SWTRIG), offset 0xF00

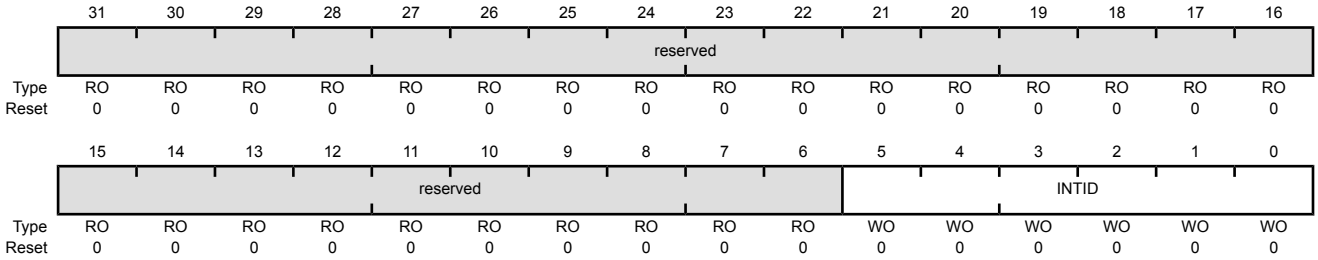
**Note:** Only privileged software can enable unprivileged access to the **SWTRIG** register.

Writing an interrupt number to the **SWTRIG** register generates a Software Generated Interrupt (SGI). See Table 2-9 on page 78 for interrupt assignments.

When the MAINPEND bit in the **Configuration and Control (CFGCTRL)** register (see page 129) 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 |
|---|---|---|---|---|
| 31:6 | 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. |
| 5:0 | 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.

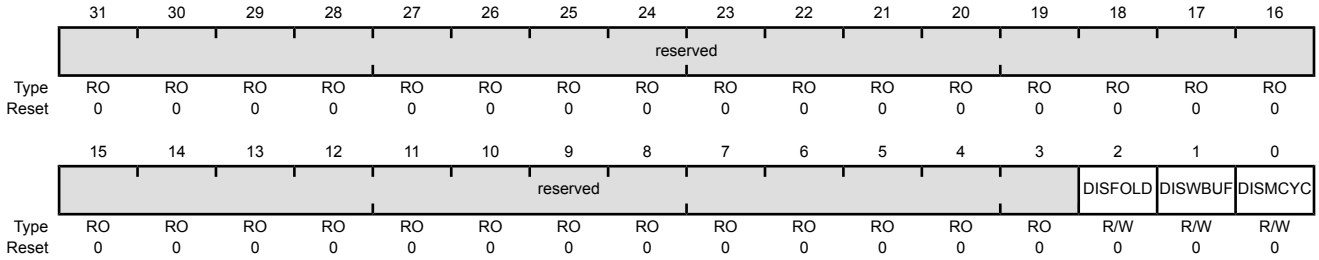
## Register 29: Auxiliary Control (ACTLR), offset 0x008

**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-M3 processor and does not normally require modification.

Auxiliary Control (ACTLR)

Base 0xE000.E000
Offset 0x008
Type R/W, 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 | DISFOLD | R/W | 0 | Disable IT Folding<br><br>Value Description<br>0 No effect.<br>1 Disables IT folding.<br><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. |
| 1 | DISWBUF | R/W | 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-M3 processor. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | DISMCYC | R/W | 0 | Disable Interrupts of Multiple Cycle Instructions |

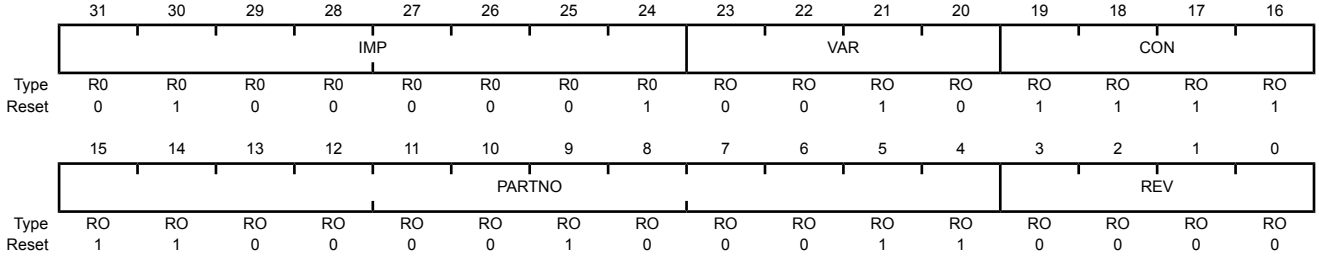
| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Disables interruption of load multiple and store multiple instructions. In this situation, the interrupt latency of the processor is increased because any LDM or STM must complete before the processor can stack the current state and enter the interrupt handler. |

## Register 30: CPU ID Base (CPUID), offset 0xD00

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

The **CPUID** register contains the ARM® Cortex™-M3 processor part number, version, and implementation information.

CPU ID Base (CPUID)
Base 0xE000.E000
Offset 0xD00
Type RO, reset 0x412F.C230




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | IMP | R0 | 0x41 | Implementer Code<br><br>Value Description<br>0x41 ARM |
| 23:20 | VAR | RO | 0x2 | Variant Number<br><br>Value Description<br>0x2 The rn value in the rnpn product revision identifier, for example, the 2 in r2p0. |
| 19:16 | CON | RO | 0xF | Constant<br><br>Value Description<br>0xF Always reads as 0xF. |
| 15:4 | PARTNO | RO | 0xC23 | Part Number<br><br>Value Description<br>0xC23 Cortex-M3 processor. |
| 3:0 | REV | RO | 0x0 | Revision Number<br><br>Value Description<br>0x0 The pn value in the rnpn product revision identifier, for example, the 0 in r2p0. |

## Register 31: Interrupt Control and State (INTCTRL), offset 0xD04

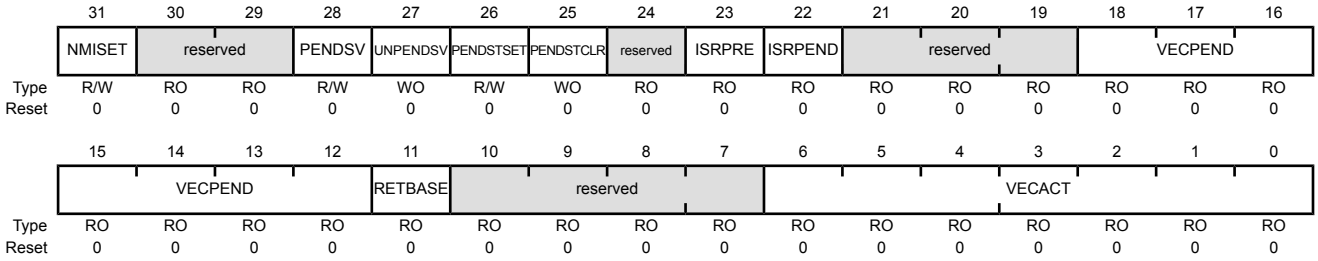
**Note:** This register can only be accessed from privileged mode.

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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NMISET | R/W | 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 | R/W | 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 | R/W | 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. |
| 21:19 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 18:12 | 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>0x46 Interrupt Vector 54<br>0x47-0x7F Reserved |
| 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). |
| 10:7 | 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. |
| 6:0 | 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 59). |

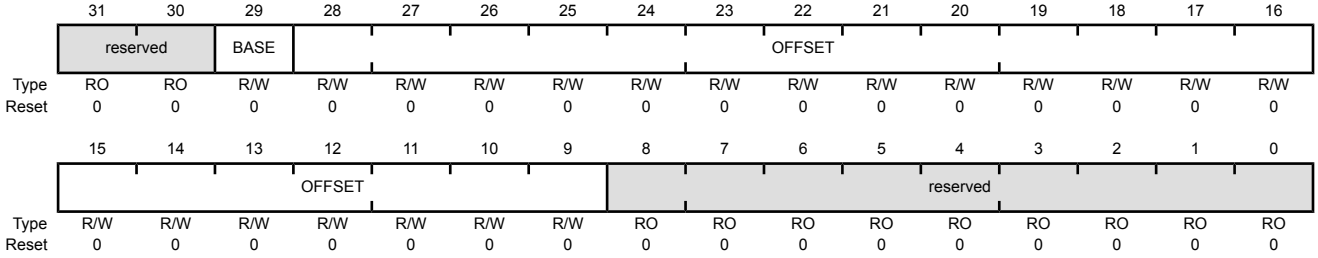
## Register 32: Vector Table Offset (VTABLE), offset 0xD08

**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 R/W, reset 0x0000.0000




| 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 | BASE | R/W | 0 | Vector Table Base<br>Value Description<br>0 The vector table is in the code memory region.<br>1 The vector table is in the SRAM memory region. |
| 28:9 | OFFSET | R/W | 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 54 interrupts, the minimum alignment is 128 words. |
| 8:0 | 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 33: 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 3-8 on page 125 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.

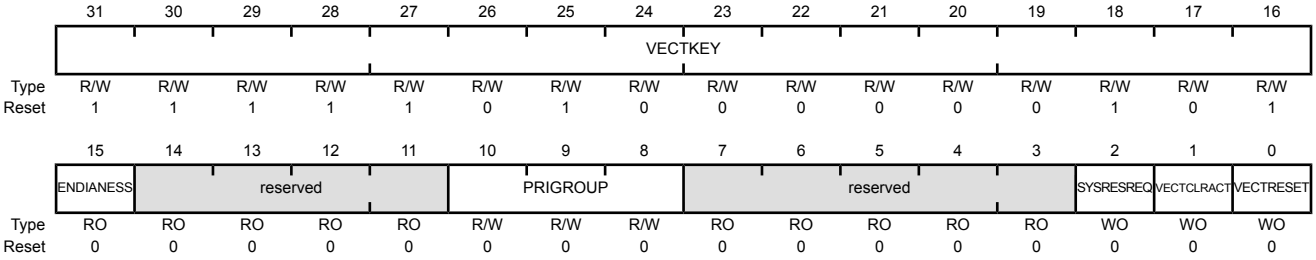
**Table 3-8. Interrupt Priority Levels**

| 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 R/W, reset 0xFA05.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | VECTKEY | R/W | 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 Stellaris 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. |

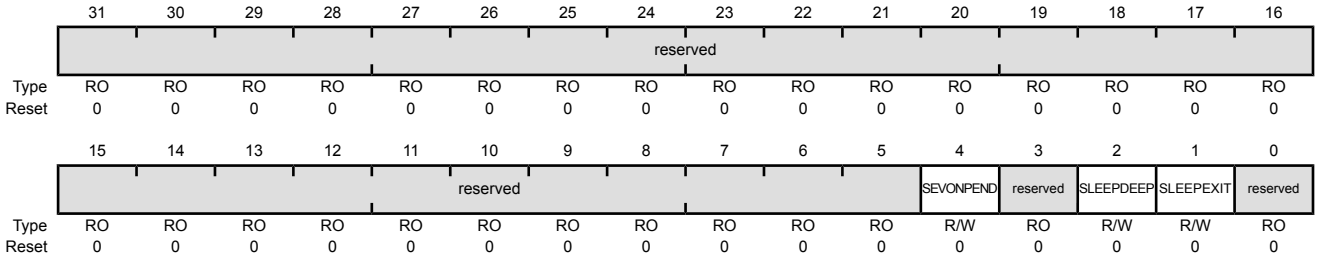
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10:8 | PRIGROUP | R/W | 0x0 | Interrupt Priority Grouping<br>This field determines the split of group priority from subpriority (see Table 3-8 on page 125 for more information). |
| 7:3 | reserved | RO | 0x0 | Software should not 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 34: 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 R/W, 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 | R/W | 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 | R/W | 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 | R/W | 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. |

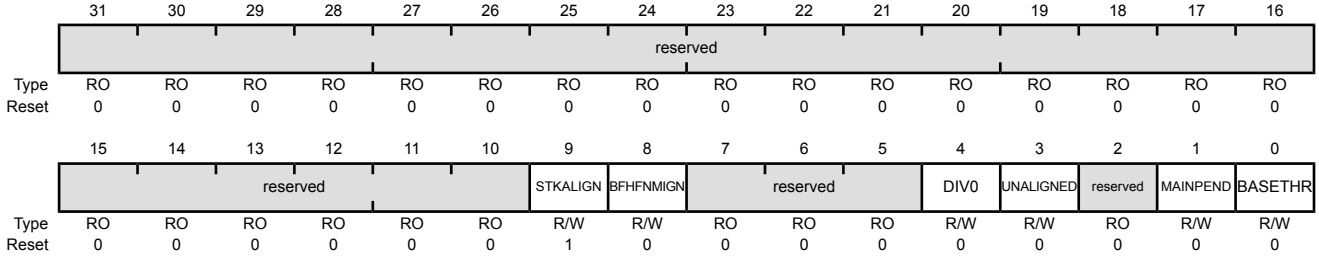
## Register 35: Configuration and Control (CFGCTRL), offset 0xD14

**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 117).

Configuration and Control (CFGCTRL)

Base 0xE000.E000
Offset 0xD14
Type R/W, 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 | R/W | 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 | R/W | 0 | Ignore Bus Fault in NMI and Fault<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 | R/W | 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 | R/W | 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 | R/W | 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 117). |
| 0 | BASETHR | R/W | 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 82 for more information). |

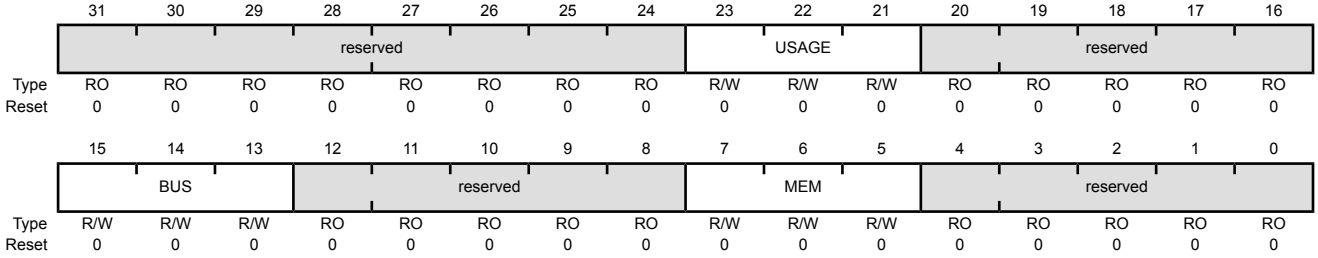
## Register 36: System Handler Priority 1 (SYSPRI1), offset 0xD18

**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 R/W, 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 | R/W | 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 | R/W | 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 | R/W | 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. |

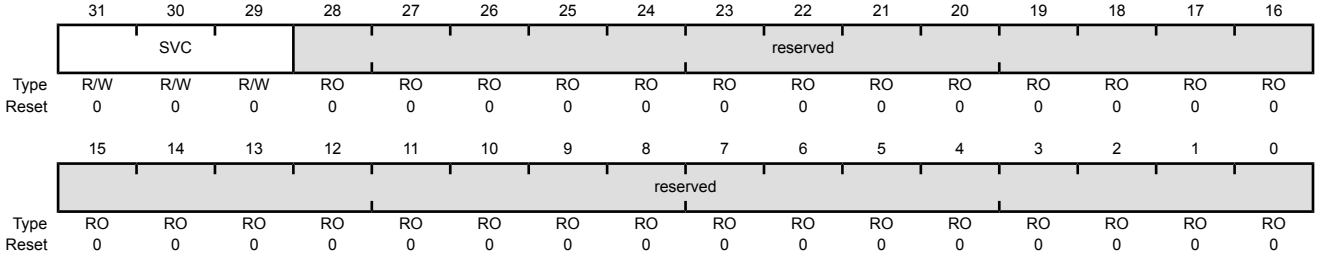
## Register 37: System Handler Priority 2 (SYSPRI2), offset 0xD1C

**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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | SVC | R/W | 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. |

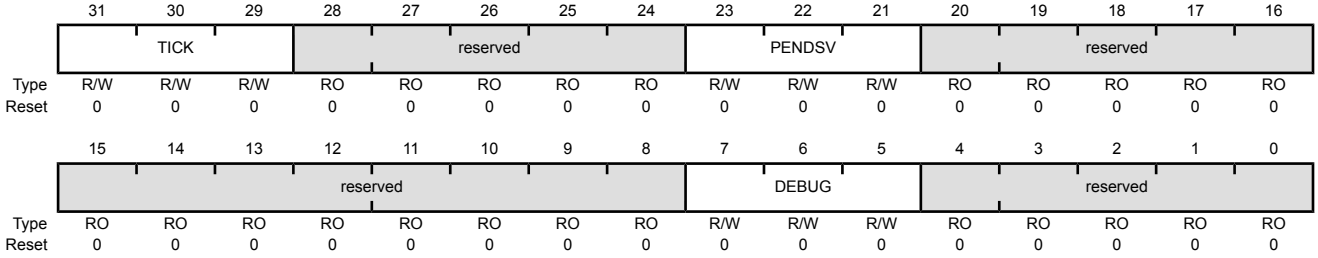
## Register 38: System Handler Priority 3 (SYSPRI3), offset 0xD20

**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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | TICK | R/W | 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 | R/W | 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 | R/W | 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 39: 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.

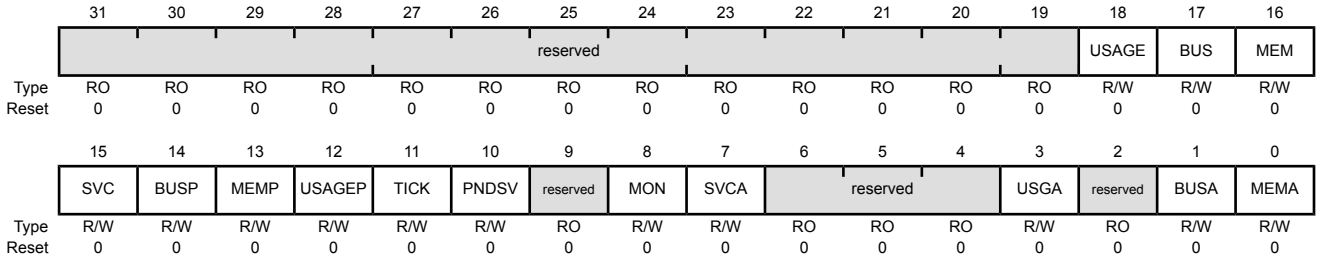
This register can be modified to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

**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 R/W, reset 0x0000.0000




| 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 | R/W | 0 | Usage Fault Enable<br>Value Description<br>0 Disables the usage fault exception.<br>1 Enables the usage fault exception. |
| 17 | BUS | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 40: 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 R/W1C, 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 |
| Type RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BFARV | reserved | | BSTKE | BUSTKE | IMPRE | PRECISE | IBUS | MMARV | reserved | | MSTKE | MUSTKE | reserved | DERR | IERR |
| Type R/W1C | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | RO | RO | R/W1C | R/W1C | RO | R/W1C | R/W1C |
| 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25 | DIV0 | R/W1C | 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 129).<br>This bit is cleared by writing a 1 to it. |
| 24 | UNALIGN | R/W1C | 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 129).<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 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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. |
| 14:13 | 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 | BSTKE | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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. |
| 6:5 | 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 | MSTKE | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 41: Hard Fault Status (HFAULTSTAT), offset 0xD2C

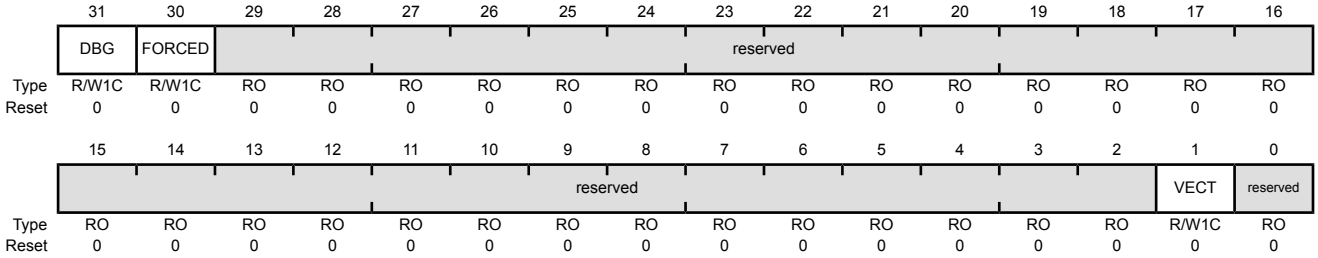
**Note:** This register can only be accessed from privileged mode.

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 R/W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | DBG | R/W1C | 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 | R/W1C | 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 | R/W1C | 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. |

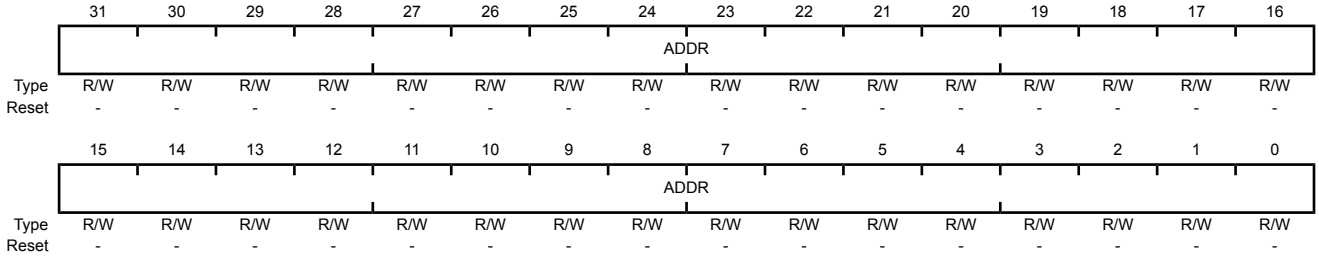
## Register 42: Memory Management Fault Address (MMADDR), offset 0xD34

**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 138).

Memory Management Fault Address (MMADDR)

Base 0xE000.E000
Offset 0xD34
Type R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | R/W | - | 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. |

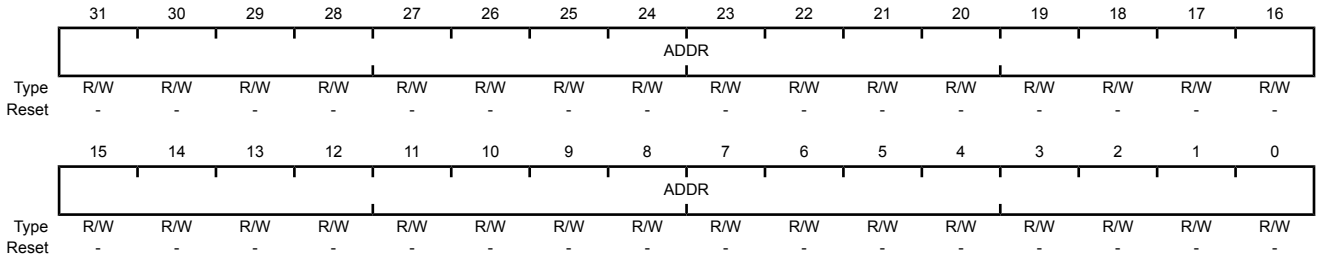
### Register 43: Bus Fault Address (FAULTADDR), offset 0xD38

**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 138).

Bus Fault Address (FAULTADDR)

Base 0xE000.E000
Offset 0xD38
Type R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | R/W | - | 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.

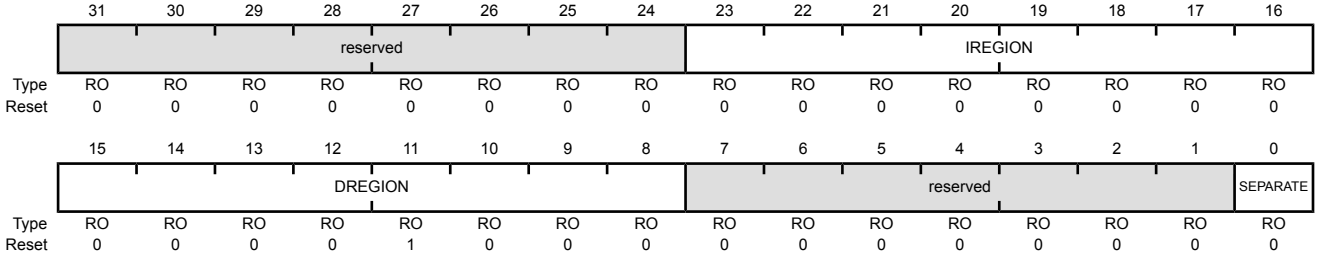
The MPU registers can only be accessed from privileged mode.

## Register 44: 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 45: 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 67. 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.

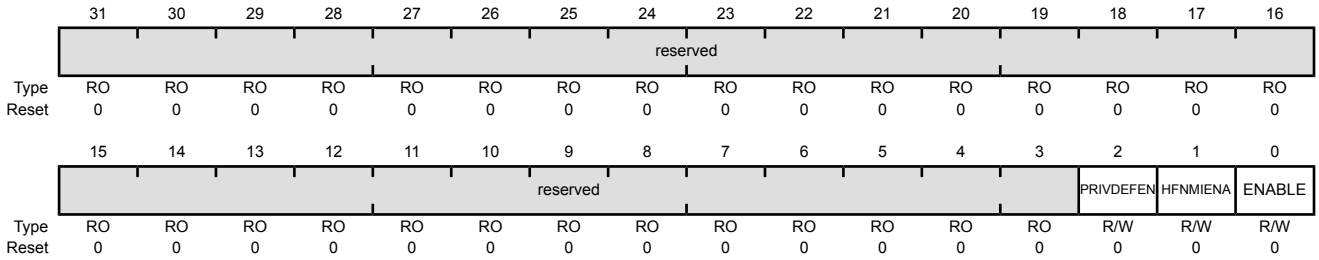
When the ENABLE bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see Table 2-5 on page 70 for more information). The default memory map applies to accesses from both privileged and unprivileged software.

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 R/W, 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | PRIVDEFEN | R/W | 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 | R/W | 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 | R/W | 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. |

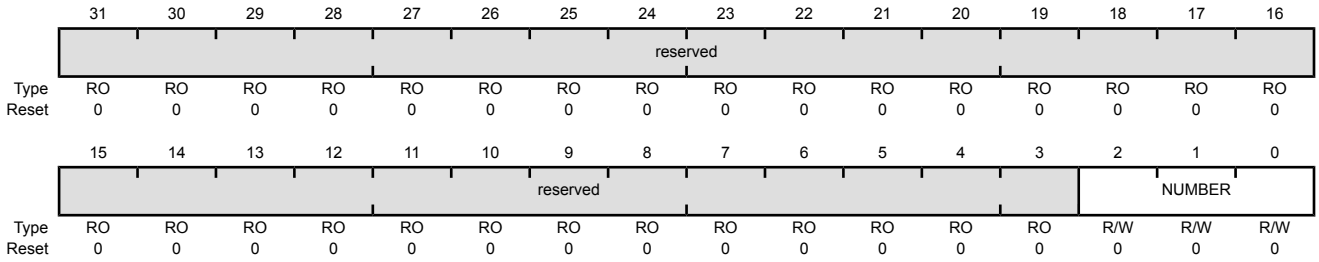
## Register 46: MPU Region Number (MPUNUMBER), offset 0xD98

**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 151). This write updates the value of the REGION field.

MPU Region Number (MPUNUMBER)

Base 0xE000.E000
Offset 0xD98
Type R/W, 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 | R/W | 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 47: MPU Region Base Address (MPUBASE), offset 0xD9C

## Register 48: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4

## Register 49: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC

## Register 50: 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:

$N = Log_2(Region\ size\ in\ bytes)$

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 R/W, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADDR | | | | | | | | | | | VALID | reserved | REGION | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | WO | RO | R/W | R/W | R/W |
| 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 | R/W | 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 | R/W | 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 51: MPU Region Attribute and Size (MPUATTR), offset 0xDA0

## Register 52: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8

## Register 53: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0

## Register 54: 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 3-9 on page 153 gives example SIZE values with the corresponding region size and value of N in the **MPU Region Base Address (MPUBASE)** register.

**Table 3-9. 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 151).

MPU Region Attribute and Size (MPUATTR)

Base 0xE000.E000
Offset 0xDA0
Type R/W, 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 | R/W | RO | R/W | R/W | R/W | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | 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 | R/W | 0 | Access Privilege<br>For information on using this bit field, see Table 3-5 on page 97. |
| 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 | R/W | 0x0 | Type Extension Mask<br>For information on using this bit field, see Table 3-3 on page 96. |
| 18 | S | R/W | 0 | Shareable<br>For information on using this bit, see Table 3-3 on page 96. |
| 17 | C | R/W | 0 | Cacheable<br>For information on using this bit, see Table 3-3 on page 96. |
| 16 | B | R/W | 0 | Bufferable<br>For information on using this bit, see Table 3-3 on page 96. |
| 15:8 | SRD | R/W | 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 95 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 | R/W | 0x0 | Region Size Mask<br>The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register. Refer to Table 3-9 on page 153 for more information. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ENABLE | R/W | 0 | Region Enable |

| Value | Description |
|---|---|
| 0 | The region is disabled. |
| 1 | The region is enabled. |

# 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 Stellaris® JTAG controller works with the ARM JTAG controller built into the Cortex-M3 core by multiplexing the TDO outputs from both JTAG controllers. ARM JTAG instructions select the ARM TDO output while Stellaris JTAG instructions select the Stellaris TDO output. The multiplexer is controlled by the Stellaris JTAG controller, which has comprehensive programming for the ARM, Stellaris, and unimplemented JTAG instructions.

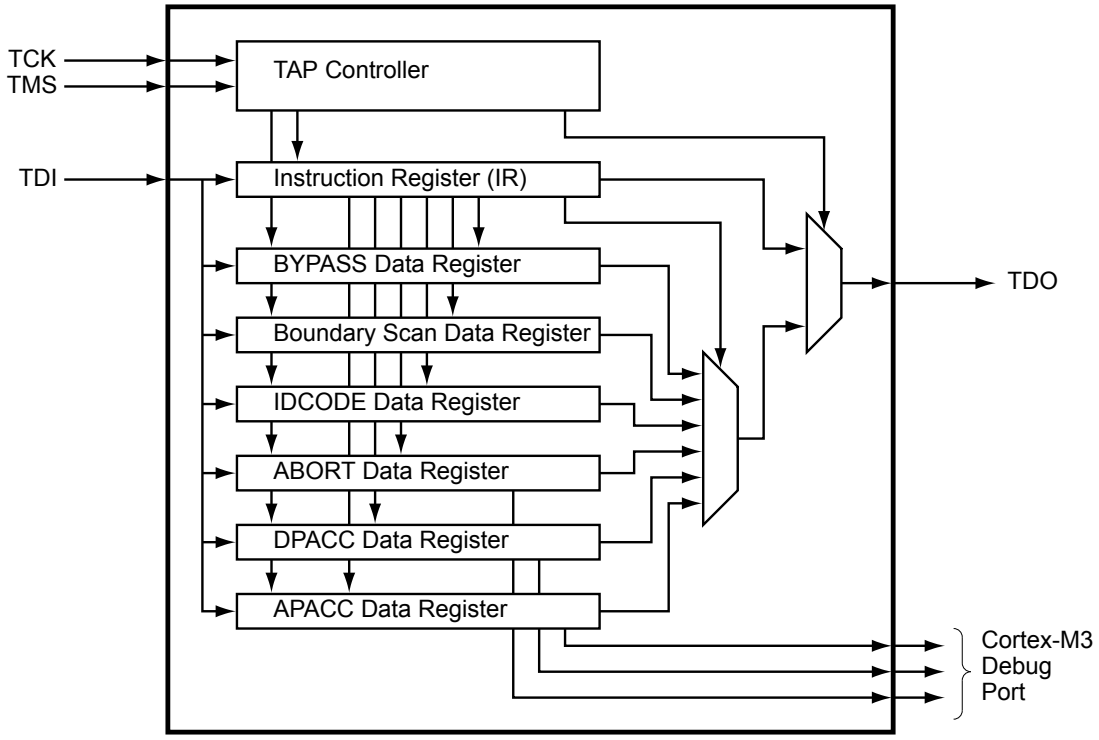
The Stellaris JTAG module has the following features:

- 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, EXTEST and INTEST
- 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
  - 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 403. 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 418) 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 435) 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 395.

**Table 4-1. Signals for JTAG_SWD_SWO (100LQFP)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| SWCLK | 80 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| SWDIO | 79 | PC1 (3) | I/O | TTL | JTAG TMS and SWDIO. |
| SWO | 77 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TCK | 80 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| TDI | 78 | PC2 (3) | I | TTL | JTAG TDI. |
| TDO | 77 | PC3 (3) | O | TTL | JTAG TDO and SWO. |

**Table 4-1. Signals for JTAG_SWD_SWO (100LQFP) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| TMS | 79 | PC1 (3) | I | TTL | JTAG TMS and SWDIO. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 4-2. Signals for JTAG_SWD_SWO (108BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| SWCLK | A9 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| SWDIO | B9 | PC1 (3) | I/O | TTL | JTAG TMS and SWDIO. |
| SWO | A10 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TCK | A9 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| TDI | B8 | PC2 (3) | I | TTL | JTAG TDI. |
| TDO | A10 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TMS | B9 | PC1 (3) | I | TTL | JTAG TMS and SWDIO. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 4.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 4-1 on page 157. 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 and INTEST, 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 4-4 on page 164 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 786 for JTAG timing diagrams.

**Note:** Of all the possible reset sources, only Power-On reset (POR) and the assertion of the $\overline{\text{RST}}$ input have any effect on the JTAG module. The pin configurations are reset by both the $\overline{\text{RST}}$ input and POR, whereas the internal JTAG logic is only reset with POR. See “Reset Sources” on page 169 for more information on 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 4-3. Detailed information on each pin follows. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 395 for information on how to reprogram the configuration of these pins.

**Table 4-3. 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 424 and page 426).

#### 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 160.

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 424).

#### 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 424).

#### 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

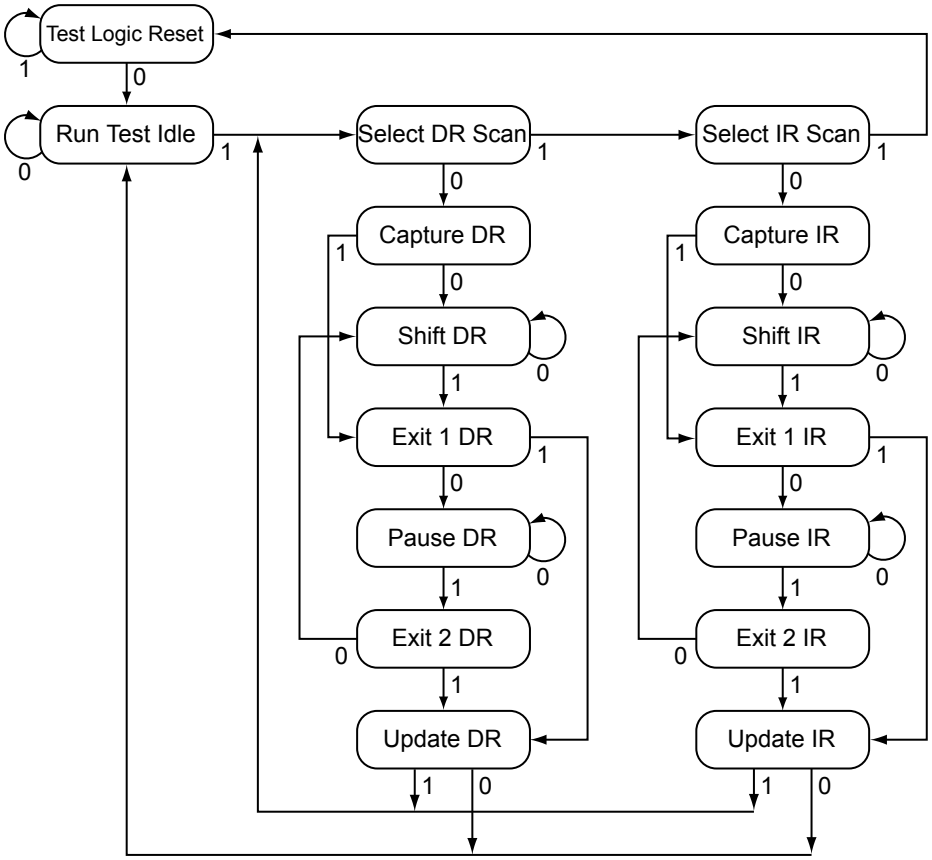
chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an inactive drive state when not actively shifting out data. Because TDO can be connected to the TDI of another controller in a daisy-chain configuration, the *IEEE Standard 1149.1* expects the 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 424 and page 426).

### 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 164.

### 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 418, page 424, page 426, and page 429.

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 Stellaris 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.**

---

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 418), **GPIO Pull Up Select (GPIOPUR)** register (see page 424), **GPIO Pull-Down Select (GPIOPDR)** register (see page 426), and **GPIO Digital Enable (GPIODEN)** register (see page 429) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 431) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 432) 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 nonvolatile registers discussed in “Nonvolatile Register Programming” on page 297 to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the nonvolatile registers being restored.

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. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence on the section called “JTAG-to-SWD Switching” on page 163.
3. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on the section called “SWD-to-JTAG Switching” on page 163.
4. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
5. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
6. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
7. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
8. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
9. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
10. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
11. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
12. Release the $\overline{\text{RST}}$ signal.
13. Wait 400 ms.
14. 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-M3 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 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/idle states.
2. Send the 16-bit JTAG-to-SWD switch command, 0xE79E, on TMS.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in SWD mode, the SWD goes into the line reset state before sending the switch sequence.

#### *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 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/idle states.
2. Send the 16-bit SWD-to-JTAG switch command, 0xE73C, on TMS.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in JTAG mode, the JTAG goes into the Test Logic Reset state before sending the switch sequence.

## 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 4-4. A detailed explanation of each instruction, along with its associated Data Register, follows.

**Table 4-4. 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. |
| 0x1 | INTEST | Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction into the controller. |
| 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 INTEST Instruction

The INTEST instruction is not associated with its own Data Register chain. Instead, the INTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the INTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the inputs are used to drive the signals going into the core rather than the signals coming from the GPIO pads. With tests that drive known values into the controller, this instruction can be used for testing. It is important to note that although the RST input pin is on the Boundary Scan Data Register chain, it is only observable.

While the INTEST 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.

#### 4.5.1.3 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 and INTEST instructions to drive data into or out of the controller. See "Boundary Scan Data Register" on page 166 for more information.

#### 4.5.1.4 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 167 for more information.

#### 4.5.1.5 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 167 for more information.

#### 4.5.1.6 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 167 for more information.

#### 4.5.1.7 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 166 for more information.

#### 4.5.1.8 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 166 for more information.

### 4.5.2 Data Registers

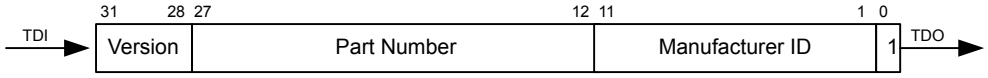
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

#### 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-M3 during debug.

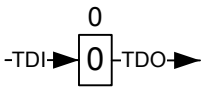
**Figure 4-3. IDCODE Register Format**




#### 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**



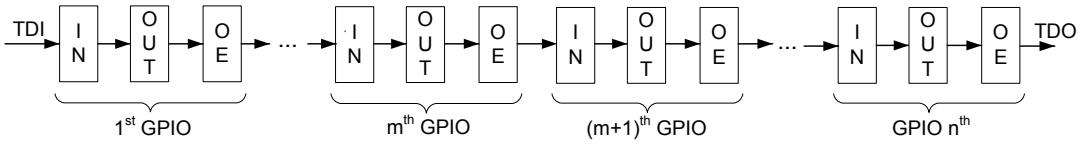

#### 4.5.2.3 Boundary Scan Data Register

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 and INTEST instructions. The EXTEST instruction forces data out of the controller, and the INTEST instruction forces data into 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 the GPIO PB7 signal and functions as a GPIO after reset. PB7 is under commit protection and requires a special process to be configured as any alternate function or to subsequently return to the GPIO function, see “Commit Control” on page 403. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the NMI signal. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 418) should be set to choose the NMI function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 435) to assign the NMI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

**Table 5-1. Signals for System Control & Clocks (100LQFP)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CMOD0 | 65 | fixed | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| CMOD1 | 76 | fixed | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |
| NMI | 89 | PB7 (4) | I | TTL | Non-maskable interrupt. |
| OSC0 | 48 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | 49 | fixed | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| $\overline{\text{RST}}$ | 64 | fixed | I | TTL | System reset input. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 5-2. Signals for System Control & Clocks (108BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CMOD0 | E11 | fixed | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| CMOD1 | B10 | fixed | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |
| NMI | A8 | PB7 (4) | I | TTL | Non-maskable interrupt. |
| OSC0 | L11 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | M11 | fixed | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| $\overline{\text{RST}}$ | H11 | fixed | I | TTL | System reset input. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 5.2 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 169
- Local control, such as reset (see “Reset Control” on page 169), power (see “Power Control” on page 175) and clock control (see “Clock Control” on page 175)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 182

### 5.2.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, Flash memory size, and other features. See the **DID0** (page 187), **DID1** (page 214), **DC0**-**DC9** (page 216) and **NVMSTAT** (page 236) registers.

### 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 CMOD0 and CMOD1 Test-Mode Control Pins

Two pins, CMOD0 and CMOD1, are defined for internal use for testing the microcontroller during manufacture. They have no end-user function and should not be used. The CMOD pins should be connected to ground.

#### 5.2.2.2 Reset Sources

The LM3S1G58 microcontroller has six sources of reset:

1. Power-on reset (POR) (see page 170).
2. External reset input pin ($\overline{\text{RST}}$) assertion (see page 171).
3. Internal brown-out (BOR) detector (see page 172).
4. Software-initiated reset (with the software reset registers) (see page 173).
5. A watchdog timer reset condition violation (see page 173).
6. MOSC failure (see page 174).

Table 5-3 provides a summary of results of the various reset operations.

**Table 5-3. Reset Sources**

| Reset Source | Core Reset? | JTAG Reset? | On-Chip Peripherals Reset?[a] |
|---|---|---|---|
| Power-On Reset | Yes | Yes | Yes |
| $\overline{\text{RST}}$ | Yes | Yes | Yes |
| Brown-Out Reset | Yes | Yes | Yes |
| Software System Request Reset using the SYSRESREQ bit in the **APINT** register. | Yes | Yes | Yes |

**Table 5-3. Reset Sources *(continued)***

| Reset Source | Core Reset? | JTAG Reset? | On-Chip Peripherals Reset?[a] |
|---|---|---|---|
| Software System Request Reset using the VECTRESET bit in the **APINT** register. | Yes | No | No |
| Software Peripheral Reset | No | Yes | Yes[b] |
| Watchdog Reset | Yes | Yes | Yes |
| MOSC Failure Reset | Yes | Yes | Yes |

a. Refer to "Register Reset" on page 272 for information on how reset affects the Hibernation module.

b. Programmable on a module-by-module basis using the Software Reset Control Registers.

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, except when an internal POR is the cause, in which case, all the bits in the **RESC** register are cleared except for the POR indicator. A bit in the **RESC** register can be cleared by writing a 0.

At any reset that resets the core, 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 as configured in the **Boot Configuration (BOOTCFG)** register.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The BA bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the EN bit is clear, 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 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 valid data at address 0x0000.0004, 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.

For example, if the **BOOTCFG** register is written and committed with the value of 0x0000.3C01, then PB7 is examined at reset to determine if the ROM Boot Loader should be executed. If PB7 is Low, the core unconditionally begins executing the ROM boot loader. If PB7 is High, then the application in Flash memory is executed if the reset vector at location 0x0000.0004 is not 0xFFFF.FFFF. Otherwise, the ROM boot loader is executed.

### 5.2.2.3 Power-On Reset (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 ($V_{TH}$). The microcontroller must be operating within the specified operating parameters when the on-chip power-on reset pulse is complete (see "Power and Brown-out" on page 788). For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the $\overline{\text{RST}}$ input may be used as discussed in "External $\overline{\text{RST}}$ Pin" on page 171.

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 loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

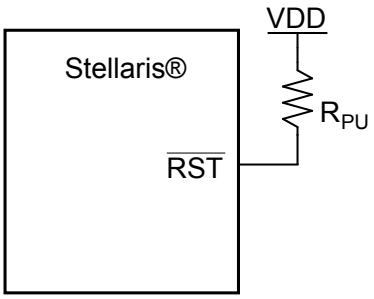
The internal POR is only active on the initial power-up of the microcontroller and when the microcontroller wakes from hibernation. The Power-On Reset timing is shown in Figure 19-4 on page 788.

#### 5.2.2.4 External $\overline{\text{RST}}$ Pin

**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 171.

**Figure 5-1. Basic $\overline{\text{RST}}$ Configuration**



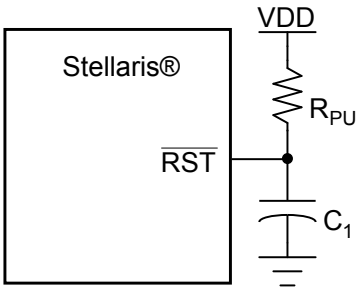

$R_{PU}$ = 0 to 100 kΩ

The external reset pin ($\overline{\text{RST}}$) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see “JTAG Interface” on page 156). 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 de-asserted (see “Reset” on page 789).
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.

To improve noise immunity and/or to delay reset at power up, the $\overline{\text{RST}}$ input may be connected to an RC network as shown in Figure 5-2 on page 172.

**Figure 5-2. External Circuitry to Extend Power-On Reset**

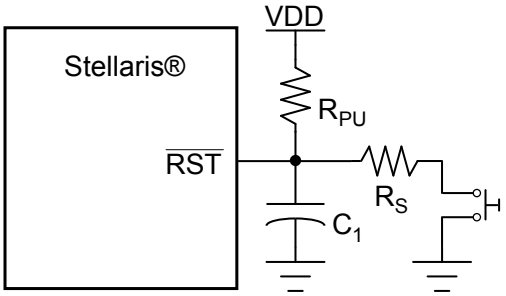



$R_{PU}$ = 1 kΩ to 100 kΩ

$C_1$ = 1 nF to 10 µF

If the application requires the use of an external reset switch, Figure 5-3 on page 172 shows the proper circuitry to use.

**Figure 5-3. Reset Circuit Controlled by Switch**




Typical $R_{PU}$ = 10 kΩ

Typical $R_S$ = 470 Ω

$C_1$ = 10 nF

The $R_{PU}$ and $C_1$ components define the power-on delay.

The external reset timing is shown in Figure 19-7 on page 789.

### 5.2.2.5 Brown-Out Reset (BOR)

The microcontroller provides a brown-out detection circuit that triggers if the power supply ($V_{DD}$) drops below a brown-out threshold voltage ($V_{BTH}$). If a brown-out condition is detected, the system may generate an interrupt or a system reset. The default condition is to reset the microcontroller. Brown-out resets are controlled with the **Power-On and Brown-Out Reset Control (PBORCTL)** register. The `BORIOR` bit in the **PBORCTL** register must be set for a brown-out condition to trigger a reset; if `BORIOR` is clear, an interrupt is generated. When a Brown-out condition occurs during a Flash PROGRAM or ERASE operation, a full system reset is always triggered without regard to the setting in the **PBORCTL** register.

The brown-out reset sequence is as follows:

1. When $V_{DD}$ drops below $V_{BTH}$, an internal BOR condition is set.

2. If the BOR condition exists, 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.

4. The internal BOR condition is reset after 500 µs to prevent another BOR condition from being set before software has a chance to investigate the original cause.

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 $V_{DD}$ 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 Figure 19-5 on page 788.

#### 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 three registers that control reset signals to each on-chip peripheral (see the **SRCRn** registers, page 257). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset. The encoding of the reset registers is consistent with the encoding of the clock gating control for peripherals and on-chip functions (see “System Control” on page 182).

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. 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 19-8 on page 789.

#### 5.2.2.7 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The LM3S1G58 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). Each module operates in the same manner except that because the PIOSC watchdog timer module is in a different

clock domain, register accesses must have a time delay between them. The watchdog timer can be configured to generate an interrupt to the microcontroller on its first time-out and to generate a 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 the reset signal has been enabled, the watchdog timer asserts its reset signal to the microcontroller. The watchdog timer 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 495.

The watchdog reset timing is shown in Figure 19-9 on page 790.

### 5.2.3 Non-Maskable Interrupt

The microcontroller has three 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™-M3 (see page 121).

Software must check the cause of the interrupt in order to distinguish among the sources.

#### 5.2.3.1 NMI Pin

The NMI signal is the alternate function for GPIO port pin PB7. 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 395. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function just like the GPIO port pins associated with JTAG/SWD functionality, see page 432. 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

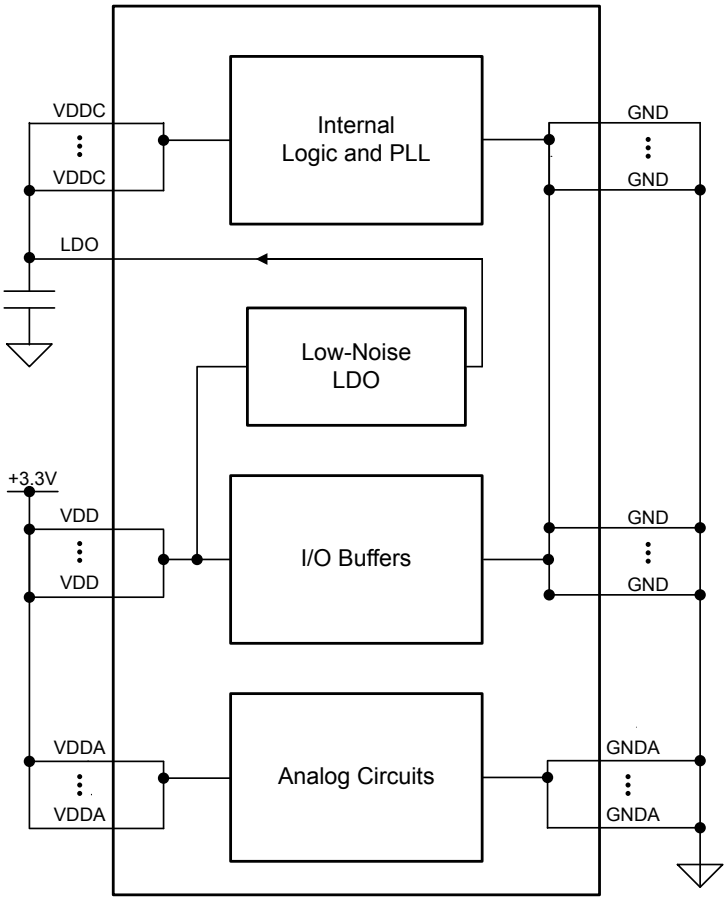
The LM3S1G58 microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or too slow. If the main oscillator verification circuit is enabled and a failure occurs, a power-on reset is generated and control is transferred to the NMI handler. 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 “Main Oscillator Verification Circuit” on page 182.

### 5.2.4 Power Control

The Stellaris® 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. An external regulator may be used instead of the on-chip LDO, but must meet the requirements in Table 19-4 on page 788. Regardless of the LDO implementation, the internal LDO requires decoupling capacitors as specified in “On-Chip Low Drop-Out (LDO) Regulator” on page 790.

**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**




### 5.2.5 Clock Control

System control 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:

- **Precision Internal Oscillator (PIOSC).** The precision internal oscillator is an on-chip clock source that is the clock source the microcontroller uses during and following POR. It does not require the use of any external components and provides a clock that is 16 MHz ±1% at room temperature and ±3% across temperature. The PIOSC allows for a reduced system cost in

applications that require an accurate 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.

- **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 must be one of the supported frequencies between 3.579545 MHz to 16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz to 16.384 MHz. The single-ended clock source range is from DC through the specified speed of the microcontroller. The supported crystals are listed in the XTAL bit field in the **RCC** register (see page 198).

- **Internal 30-kHz Oscillator.** The internal 30-kHz oscillator provides an operational frequency of 30 kHz ± 50%. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the MOSC to be powered down.

- **Hibernation Module Clock Source.** The Hibernation module can be clocked in one of two ways. The first way is a 4.194304-MHz crystal connected to the XOSC0 and XOSC1 pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. The second way is 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. The Hibernation module clock source 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.

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL and the precision internal oscillator divided by four (4 MHz ± 1%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 16.384 MHz (inclusive). Table 5-4 on page 176 shows how the various clock sources can be used in a system.

**Table 5-4. Clock Source Options**

| Clock Source | Drive PLL? | | Used as SysClk? | |
|---|---|---|---|---|
| Precision Internal Oscillator | Yes | BYPASS = 0, OSCSRC = 0x1 | Yes | BYPASS = 1, OSCSRC = 0x1 |
| Precision Internal Oscillator divide by 4 (4 MHz ± 1%) | No | - | Yes | BYPASS = 1, OSCSRC = 0x2 |
| Main Oscillator | Yes | BYPASS = 0, OSCSRC = 0x0 | Yes | BYPASS = 1, OSCSRC = 0x0 |
| Internal 30-kHz Oscillator | No | - | Yes | BYPASS = 1, OSCSRC = 0x3 |
| Hibernation Module 32.768-kHz Oscillator | No | - | Yes | BYPASS = 1, OSCSRC2 = 0x7 |
| Hibernation Module 4.194304-MHz Crystal | No | - | No | - |

### 5.2.5.2 Clock Configuration

The **Run-Mode Clock Configuration (RCC)** and **Run-Mode Clock Configuration 2 (RCC2)** registers provide control for the system clock. The **RCC2** register is provided to extend fields that offer additional encodings over the **RCC** register. When used, the **RCC2** register field values are used by the logic over the corresponding field in the **RCC** register. In particular, **RCC2** provides for

a larger assortment of clock configuration options. These registers control the following clock functionality:

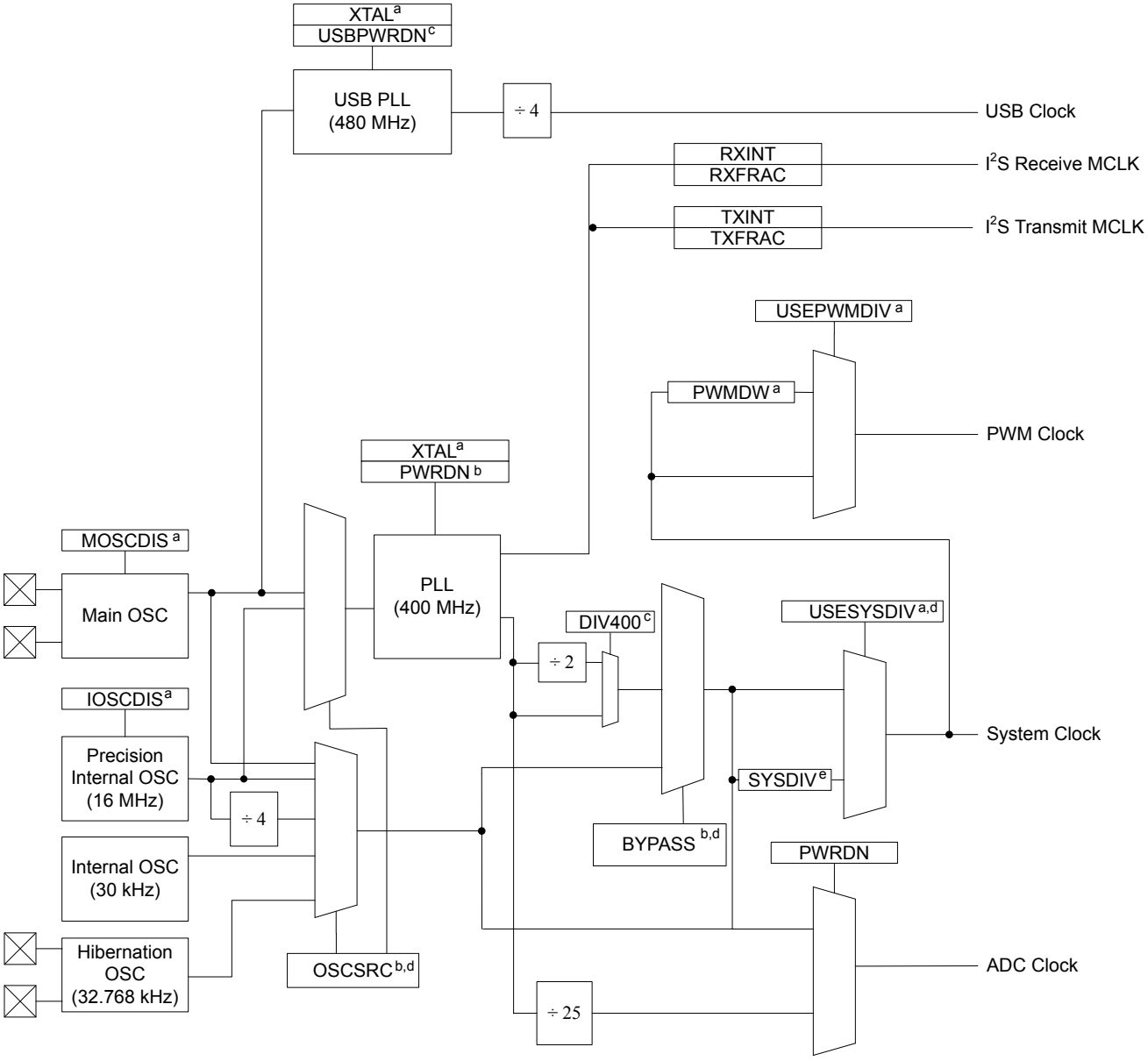
- Source of clocks in sleep and deep-sleep modes
- System clock derived from PLL or other clock source
- Enabling/disabling of oscillators and PLL
- Clock divisors
- Crystal input selection

**Important:** Write the **RCC** register prior to writing the **RCC2** register. If a subsequent write to the **RCC** register is required, include another register access after writing the **RCC** register and before writing the **RCC2** register.

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. When the PLL is enabled, the ADC clock signal is automatically divided down to 16 MHz from the PLL output for proper ADC operation.

**Note:** When the ADC module is in operation, the system clock must be at least 16 MHz.

**Figure 5-5. Main Clock Tree**




a. Control provided by **RCC** register bit/field.
b. Control provided by **RCC** register bit/field or **RCC2** register bit/field, if overridden with **RCC2** register bit USERCC2.
c. Control provided by **RCC2** register bit/field.
d. Also may be controlled by **DSLPCLKCFG** when in deep sleep mode.
e. Control provided by **RCC** register SYSDIV field, **RCC2** register SYSDIV2 field if overridden with USERCC2 bit, or [SYSDIV2,SYSDIV2LSB] if both USERCC2 and DIV400 bits are set.

**Note:** The figure above shows all features available on all Stellaris® Firestorm-class microcontrollers. Not all peripherals may be available on this device.

### *Using the SYSDIV and SYSDIV2 Fields*

In the **RCC** register, the SYSDIV field specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS bit in this register

is configured). When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. Table 5-5 shows how the SYSDIV encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS=0) or another clock source is used (BYPASS=1). The divisor is equivalent to the SYSDIV encoding plus 1. For a list of possible clock sources, see Table 5-4 on page 176.

**Table 5-5. Possible System Clock Frequencies Using the SYSDIV Field**

| SYSDIV | Divisor | Frequency (BYPASS=0) | Frequency (BYPASS=1) | StellarisWare® Parameter[a] |
|---|---|---|---|---|
| 0x0 | /1 | reserved | Clock source frequency/2 | SYSCTL_SYSDIV_1[b] |
| 0x1 | /2 | reserved | Clock source frequency/2 | SYSCTL_SYSDIV_2 |
| 0x2 | /3 | 66.67 MHz | Clock source frequency/3 | SYSCTL_SYSDIV_3 |
| 0x3 | /4 | 50 MHz | Clock source frequency/4 | SYSCTL_SYSDIV_4 |
| 0x4 | /5 | 40 MHz | Clock source frequency/5 | SYSCTL_SYSDIV_5 |
| 0x5 | /6 | 33.33 MHz | Clock source frequency/6 | SYSCTL_SYSDIV_6 |
| 0x6 | /7 | 28.57 MHz | Clock source frequency/7 | SYSCTL_SYSDIV_7 |
| 0x7 | /8 | 25 MHz | Clock source frequency/8 | SYSCTL_SYSDIV_8 |
| 0x8 | /9 | 22.22 MHz | Clock source frequency/9 | SYSCTL_SYSDIV_9 |
| 0x9 | /10 | 20 MHz | Clock source frequency/10 | SYSCTL_SYSDIV_10 |
| 0xA | /11 | 18.18 MHz | Clock source frequency/11 | SYSCTL_SYSDIV_11 |
| 0xB | /12 | 16.67 MHz | Clock source frequency/12 | SYSCTL_SYSDIV_12 |
| 0xC | /13 | 15.38 MHz | Clock source frequency/13 | SYSCTL_SYSDIV_13 |
| 0xD | /14 | 14.29 MHz | Clock source frequency/14 | SYSCTL_SYSDIV_14 |
| 0xE | /15 | 13.33 MHz | Clock source frequency/15 | SYSCTL_SYSDIV_15 |
| 0xF | /16 | 12.5 MHz (default) | Clock source frequency/16 | SYSCTL_SYSDIV_16 |

a. This parameter is used in functions such as SysCtlClockSet() in the Stellaris Peripheral Driver Library.

b. SYSCTL_SYSDIV_1 does not set the USESYSDIV bit. As a result, using this parameter without enabling the PLL results in the system clock having the same frequency as the clock source.

The SYSDIV2 field in the **RCC2** register is 2 bits wider than the SYSDIV field in the **RCC** register so that additional larger divisors up to /64 are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. The divisor is equivalent to the SYSDIV2 encoding plus 1. Table 5-6 shows how the SYSDIV2 encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS2=0) or another clock source is used (BYPASS2=1). For a list of possible clock sources, see Table 5-4 on page 176.

**Table 5-6. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field**

| SYSDIV2 | Divisor | Frequency (BYPASS2=0) | Frequency (BYPASS2=1) | StellarisWare Parameter[a] |
|---|---|---|---|---|
| 0x00 | /1 | reserved | Clock source frequency/2 | SYSCTL_SYSDIV_1[b] |
| 0x01 | /2 | reserved | Clock source frequency/2 | SYSCTL_SYSDIV_2 |
| 0x02 | /3 | 66.67 MHz | Clock source frequency/3 | SYSCTL_SYSDIV_3 |
| 0x03 | /4 | 50 MHz | Clock source frequency/4 | SYSCTL_SYSDIV_4 |
| 0x04 | /5 | 40 MHz | Clock source frequency/5 | SYSCTL_SYSDIV_5 |
| ... | ... | ... | ... | ... |
| 0x09 | /10 | 20 MHz | Clock source frequency/10 | SYSCTL_SYSDIV_10 |
| ... | ... | ... | ... | ... |

**Table 5-6. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field *(continued)***

| SYSDIV2 | Divisor | Frequency (BYPASS2=0) | Frequency (BYPASS2=1) | StellarisWare Parameter[a] |
|---|---|---|---|---|
| 0x3F | /64 | 3.125 MHz | Clock source frequency/64 | SYSCTL_SYSDIV_64 |

a. This parameter is used in functions such as SysCtlClockSet() in the Stellaris Peripheral Driver Library.

b. SYSCTL_SYSDIV_1 does not set the USESYSDIV bit. As a result, using this parameter without enabling the PLL results in the system clock having the same frequency as the clock source.

To allow for additional frequency choices when using the PLL, the DIV400 bit is provided along with the SYSDIV2LSB bit. When the DIV400 bit is set, bit 22 becomes the LSB for SYSDIV2. In this situation, the divisor is equivalent to the (SYSDIV2 encoding with SYSDIV2LSB appended) plus one. Table 5-7 shows the frequency choices when DIV400 is set. When the DIV400 bit is clear, SYSDIV2LSB is ignored, and the system clock frequency is determined as shown in Table 5-6 on page 179.

**Table 5-7. Examples of Possible System Clock Frequencies with DIV400=1**

| SYSDIV2 | SYSDIV2LSB | Divisor | Frequency (BYPASS2=0)[a] | StellarisWare Parameter[b] |
|---|---|---|---|---|
| 0x00 | reserved | /2 | reserved | - |
| 0x01 | 0 | /3 | reserved | - |
| | 1 | /4 | reserved | - |
| 0x02 | 0 | /5 | 80 MHz | SYSCTL_SYSDIV_2_5 |
| | 1 | /6 | 66.67 MHz | SYSCTL_SYSDIV_3 |
| 0x03 | 0 | /7 | reserved | - |
| | 1 | /8 | 50 MHz | SYSCTL_SYSDIV_4 |
| 0x04 | 0 | /9 | 44.44 MHz | SYSCTL_SYSDIV_4_5 |
| | 1 | /10 | 40 MHz | SYSCTL_SYSDIV_5 |
| ... | ... | ... | ... | ... |
| 0x3F | 0 | /127 | 3.15 MHz | SYSCTL_SYSDIV_63_5 |
| | 1 | /128 | 3.125 MHz | SYSCTL_SYSDIV_64 |

a. Note that DIV400 and SYSDIV2LSB are only valid when BYPASS2=0.

b. This parameter is used in functions such as SysCtlClockSet() in the Stellaris Peripheral Driver Library.

### 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 as it is used for internal functions. The PIOSC can only be disabled during Deep-Sleep mode. It can be powered down by setting the IOSCDIS bit in the **RCC** register.

The PIOSC generates a 16-MHz clock with a ±1% accuracy at room temperatures. Across the extended temperature range, the accuracy is ±3%. 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 one of three 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 Crystal Configuration for the Main Oscillator (MOSC)

The main oscillator supports the use of a select number of crystals. If the main oscillator is used by the PLL as a reference clock, the supported range of crystals is 3.579545 to 16.384 MHz, otherwise, the range of supported crystals is 1 to 16.384 MHz.

The XTAL bit in the **RCC** register (see page 198) describes the available crystal choices and default programming values.

Software configures the **RCC** register XTAL field with the crystal number. If the PLL is used in the design, the XTAL field value is internally translated to the PLL settings.

#### 5.2.5.5 Main PLL Frequency Configuration

The main 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 main PLL to drive the output. The PLL operates at 400 MHz, but is divided by two prior to the application of the output divisor, unless the DIV400 bit in the **RCC2** register is set.

To configure the PIOSC to be the clock source for the main PLL, program the OSCRC2 field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x1.

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 202). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency. Table 19-8 on page 791 shows the actual PLL frequency and error for a given crystal choice.

The Crystal Value field (XTAL) in the **Run-Mode Clock Configuration (RCC)** register (see page 198) describes the available crystal choices and default programming of the **PLLCFG** register. Any time the XTAL field changes, the new settings are translated and the internal PLL settings are updated.

#### 5.2.5.6 PLL Modes

- Normal: The PLL multiplies the input clock reference 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 **RCC**/**RCC2** register fields (see page 198 and page 205).

#### 5.2.5.7 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 19-7 on page 790). During the relock time, the affected PLL is not usable as a clock reference.

The PLL is changed by one of the following:

- Change to the XTAL value in the **RCC** register—writes of the same value do not cause a relock.

- Change in the PLL from Power-Down to Normal mode.

A counter clocked by the system clock is used to measure the $T_{READY}$ requirement. If the system clock is the main oscillator and it is running off an 8.192 MHz or slower external oscillator clock, the down counter is set to 0x1200 (that is, ~600 µs at an 8.192 MHz). If the system clock is running off the PIOSC or an external oscillator clock that is faster than 8.192 MHz, the down counter is set to 0x2400. 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 **RCC**/**RCC2** register is switched to use the PLL.

If the main PLL is enabled and the system clock is switched to use the PLL in one step, the system control hardware continues to clock the microcontroller from the oscillator selected by the **RCC**/**RCC2** register until the main PLL is stable ($T_{READY}$ time met), after which it changes to the PLL. Software can use many methods to ensure that the system is clocked from the main PLL, including periodically polling the PLLLRIS bit in the **Raw Interrupt Status (RIS)** register, and enabling the PLL Lock interrupt.

#### 5.2.5.8 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, the following sequence is performed by the hardware:

1. The MOSCFAIL bit in the **Reset Cause (RESC)** register is set.
2. If the internal oscillator (PIOSC) is disabled, it is enabled.
3. The system clock is switched from the main oscillator to the PIOSC.
4. An internal power-on reset is initiated that lasts for 32 PIOSC periods.
5. Reset is de-asserted and the processor is directed to the NMI handler during the reset sequence.

if the MOSCIM bit in the **MOSCCTL** register is set, then the following sequence is performed by the hardware:

1. The system clock is switched from the main oscillator to the PIOSC.
2. The MOFRIS bit in the **RIS** register is set to indicate a MOSC failure.

### 5.2.6 System Control

For power-savings purposes, the **RCGCn**, **SCGCn**, and **DCGCn** registers control the clock gating logic for each 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. There must be a delay of 3 system clocks after a peripheral module clock is enabled in the **RCGC** register before any module registers are accessed.

There are four levels of operation for the microcontroller defined as:

- Run mode

- Sleep mode
- Deep-Sleep mode
- Hibernation mode

The following sections describe the different modes in detail.

**Caution – If the Cortex-M3 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.**

#### 5.2.6.1 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 **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.

#### 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-M3 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 85 for more details.

Peripherals are clocked that are enabled in the **SCGCn** registers when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** registers when the auto-clock gating is disabled. The system clock has the same source and frequency as that during Run mode.

#### 5.2.6.3 Deep-Sleep Mode

In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Run 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 127) 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 85 for more details.

The Cortex-M3 processor core and the memory subsystem are not clocked in Deep-Sleep mode. Peripherals are clocked that are enabled in the **DCGCn** registers when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** registers when auto-clock gating is disabled. The system clock source is specified in the **DSLPCLKCFG** register. When the **DSLPCLKCFG** 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 powers the PLL down and overrides

the SYSDIV field of the active **RCC**/**RCC2** register, to be determined by the DSDIVORIDE setting in the **DSLPCLKCFG** register, up to /16 or /64 respectively. 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 209.

#### 5.2.6.4 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-M3 processor and peripherals outside of the Hibernation module see a normal "power on" sequence and the processor starts running code. Software can determine if the microcontroller has been restarted from Hibernate mode by inspecting the Hibernation module registers. For more information on the operation of Hibernation mode, see “Hibernation Module” on page 263.

## 5.3 Initialization and Configuration

The PLL is configured using direct register writes to the **RCC**/**RCC2** register. If the **RCC2** register is being used, the USERCC2 bit must be set and the appropriate **RCC2** bit/field is used. The steps required to successfully change the PLL-based system clock are:

1. Bypass the PLL and system clock divider by setting the BYPASS bit and clearing the USESYS bit in the **RCC** register, thereby configuring the microcontroller to run off a “raw” clock source and allowing for the new PLL configuration to be validated before switching the system clock to the PLL.
2. Select the crystal value (XTAL) and oscillator source (OSCSRC), and clear the PWRDN bit in **RCC**/**RCC2**. Setting the XTAL field automatically pulls valid PLL configuration data for the appropriate crystal, and clearing the PWRDN bit powers and enables the PLL and its output.
3. Select the desired system divider (SYSDIV) in **RCC**/**RCC2** and set the USESYS bit in **RCC**. The SYSDIV field determines the system frequency for the microcontroller.
4. Wait for the PLL to lock by polling the PLLLRIS bit in the **Raw Interrupt Status (RIS)** register.
5. Enable use of the PLL by clearing the BYPASS bit in **RCC**/**RCC2**.

## 5.4 Register Map

Table 5-8 on page 185 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 290.

**Table 5-8. System Control Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | DID0 | RO | - | Device Identification 0 | 187 |
| 0x004 | DID1 | RO | - | Device Identification 1 | 214 |
| 0x008 | DC0 | RO | 0x00FF.00BF | Device Capabilities 0 | 216 |
| 0x010 | DC1 | RO | - | Device Capabilities 1 | 217 |
| 0x014 | DC2 | RO | 0x000F.5037 | Device Capabilities 2 | 219 |
| 0x018 | DC3 | RO | 0xBFFF.0000 | Device Capabilities 3 | 221 |
| 0x01C | DC4 | RO | 0x0004.F0FF | Device Capabilities 4 | 223 |
| 0x020 | DC5 | RO | 0x0000.0000 | Device Capabilities 5 | 225 |
| 0x024 | DC6 | RO | 0x0000.0000 | Device Capabilities 6 | 226 |
| 0x028 | DC7 | RO | 0xFFFF.FFFF | Device Capabilities 7 | 227 |
| 0x02C | DC8 | RO | 0xFFFF.FFFF | Device Capabilities 8 ADC Channels | 231 |
| 0x030 | PBORCTL | R/W | 0x0000.0002 | Brown-Out Reset Control | 189 |
| 0x040 | SRCR0 | R/W | 0x00000000 | Software Reset Control 0 | 257 |
| 0x044 | SRCR1 | R/W | 0x00000000 | Software Reset Control 1 | 259 |
| 0x048 | SRCR2 | R/W | 0x00000000 | Software Reset Control 2 | 261 |
| 0x050 | RIS | RO | 0x0000.0000 | Raw Interrupt Status | 190 |
| 0x054 | IMC | R/W | 0x0000.0000 | Interrupt Mask Control | 192 |
| 0x058 | MISC | R/W1C | 0x0000.0000 | Masked Interrupt Status and Clear | 194 |
| 0x05C | RESC | R/W | - | Reset Cause | 196 |
| 0x060 | RCC | R/W | 0x0780.3AD1 | Run-Mode Clock Configuration | 198 |
| 0x064 | PLLCFG | RO | - | XTAL to PLL Translation | 202 |
| 0x06C | GPIOHBCTL | R/W | 0x0000.0000 | GPIO High-Performance Bus Control | 203 |
| 0x070 | RCC2 | R/W | 0x07C0.6810 | Run-Mode Clock Configuration 2 | 205 |
| 0x07C | MOSCCTL | R/W | 0x0000.0000 | Main Oscillator Control | 208 |
| 0x100 | RCGC0 | R/W | 0x00000040 | Run Mode Clock Gating Control Register 0 | 237 |
| 0x104 | RCGC1 | R/W | 0x00000000 | Run Mode Clock Gating Control Register 1 | 243 |
| 0x108 | RCGC2 | R/W | 0x00000000 | Run Mode Clock Gating Control Register 2 | 251 |
| 0x110 | SCGC0 | R/W | 0x00000040 | Sleep Mode Clock Gating Control Register 0 | 239 |
| 0x114 | SCGC1 | R/W | 0x00000000 | Sleep Mode Clock Gating Control Register 1 | 245 |
| 0x118 | SCGC2 | R/W | 0x00000000 | Sleep Mode Clock Gating Control Register 2 | 253 |
| 0x120 | DCGC0 | R/W | 0x00000040 | Deep Sleep Mode Clock Gating Control Register 0 | 241 |
| 0x124 | DCGC1 | R/W | 0x00000000 | Deep-Sleep Mode Clock Gating Control Register 1 | 248 |

**Table 5-8. System Control Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x128 | DCGC2 | R/W | 0x00000000 | Deep Sleep Mode Clock Gating Control Register 2 | 255 |
| 0x144 | DSLPCLKCFG | R/W | 0x0780.0000 | Deep Sleep Clock Configuration | 209 |
| 0x150 | PIOSCCAL | R/W | 0x0000.0000 | Precision Internal Oscillator Calibration | 211 |
| 0x154 | PIOSCSTAT | RO | 0x0000.0040 | Precision Internal Oscillator Statistics | 213 |
| 0x190 | DC9 | RO | 0x00FF.00FF | Device Capabilities 9 ADC Digital Comparators | 234 |
| 0x1A0 | NVMSTAT | RO | 0x0000.0001 | Non-Volatile Memory Information | 236 |

## 5.5 Register Descriptions

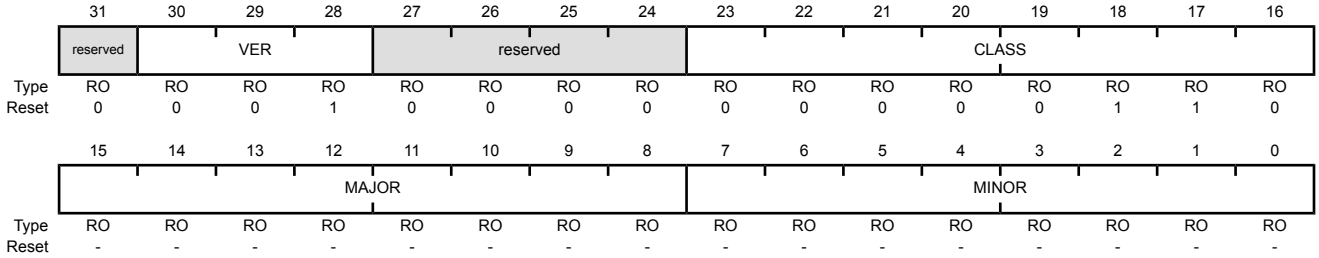
All addresses given are relative to the System Control base address of 0x400F.E000.

## 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.

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>Value Description<br>0x1 Second version of the **DID0** register format. |
| 27:24 | 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. |
| 23:16 | CLASS | RO | 0x06 | 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>Value Description<br>0x06 Stellaris® Firestorm-class microcontrollers |

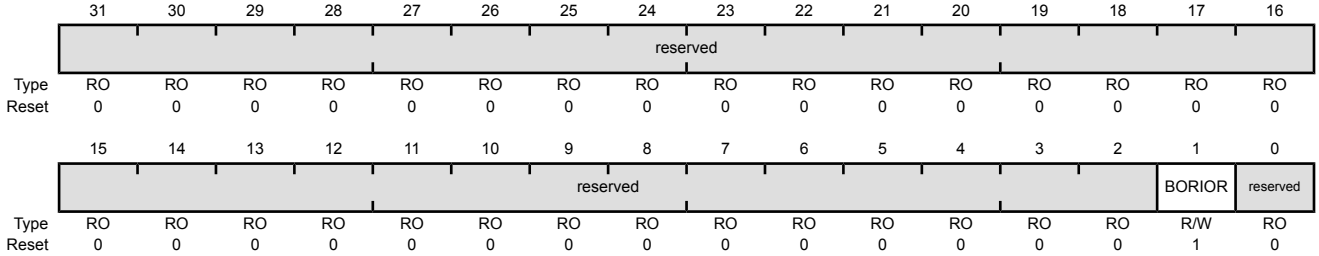
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:8 | MAJOR | RO | - | Major Revision<br>This field specifies the major revision number of the microcontroller. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part 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: Brown-Out Reset Control (PBORCTL), offset 0x030

This register is responsible for controlling reset conditions after initial power-on reset.

Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000
Offset 0x030
Type R/W, reset 0x0000.0002



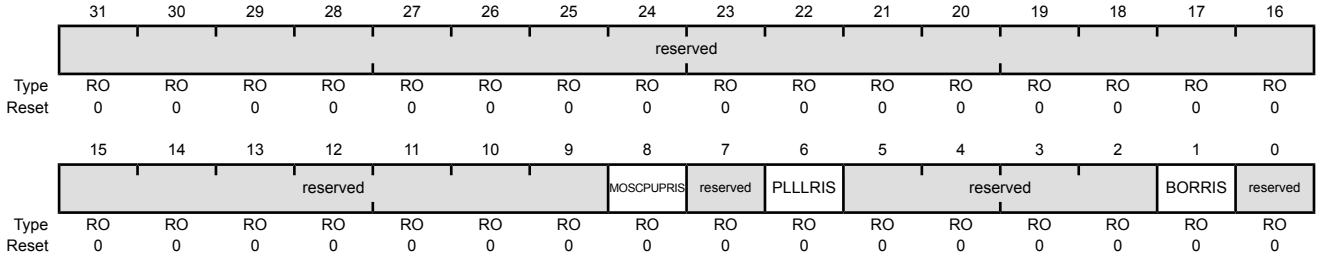

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | BORIOR | R/W | 1 | BOR Interrupt or Reset<br>Value Description<br>0 A Brown Out Event causes an interrupt to be generated to the interrupt controller.<br>1 A Brown Out Event causes a reset of the microcontroller. |
| 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 3: 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>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>0 Sufficient time has not passed for the MOSC to reach the expected frequency.<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>1 The PLL timer has reached $T_{READY}$ indicating that sufficient time has passed for the PLL to lock.<br>0 The PLL timer has not reached $T_{READY}$.<br><br>This bit is cleared by writing a 1 to the `PLLLMIS` bit in the **MISC** register. |
| 5:2 | 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. |

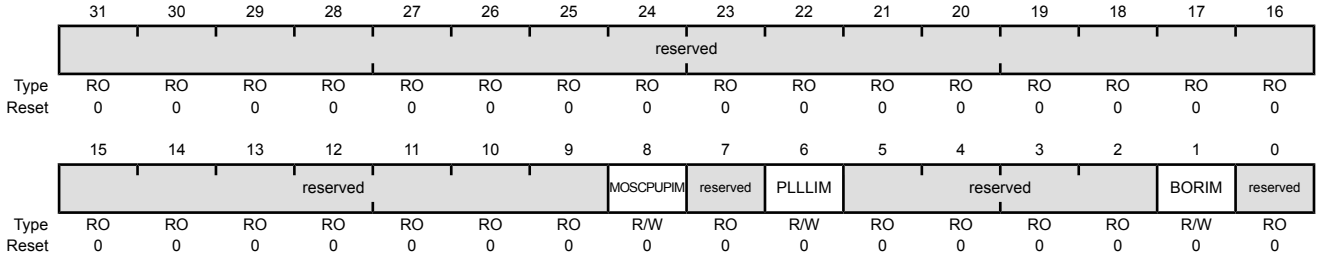
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | BORRIS | RO | 0 | Brown-Out Reset Raw Interrupt Status<br><br>Value Description<br>1 A brown-out condition is currently active.<br>0 A brown-out condition is not currently active.<br><br>Note the BORIOR bit in the **PBORCTL** register must be cleared to cause an interrupt due to a Brown Out Event.<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 4: 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 R/W, 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 | MOSCPUPIM | R/W | 0 | MOSC Power Up Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the **RIS** register is set.<br>0 The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller. |
| 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 | R/W | 0 | PLL Lock Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the PLLLRIS bit in the **RIS** register is set.<br>0 The PLLLRIS interrupt is suppressed and not sent to the interrupt controller. |
| 5:2 | 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. |

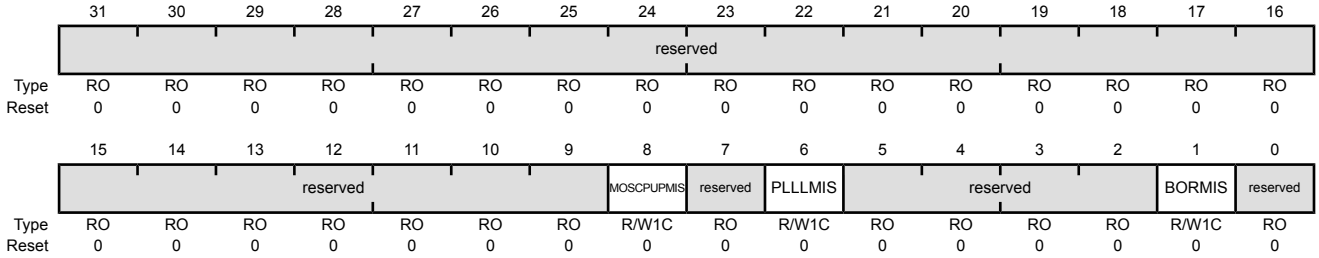
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | BORIM | R/W | 0 | Brown-Out Reset Interrupt Mask<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the BORRIS bit in the **RIS** register is set.<br>0 The BORRIS interrupt is suppressed and 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 5: 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 R/W1C, thus writing a 1 to a bit clears the corresponding raw interrupt bit in the **RIS** register (see page 190).

Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000
Offset 0x058
Type R/W1C, 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 | MOSCPUPMIS | R/W1C | 0 | MOSC Power Up Masked Interrupt Status<br>Value Description<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.<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. |
| 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 | R/W1C | 0 | PLL Lock Masked Interrupt Status<br>Value Description<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.<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. |

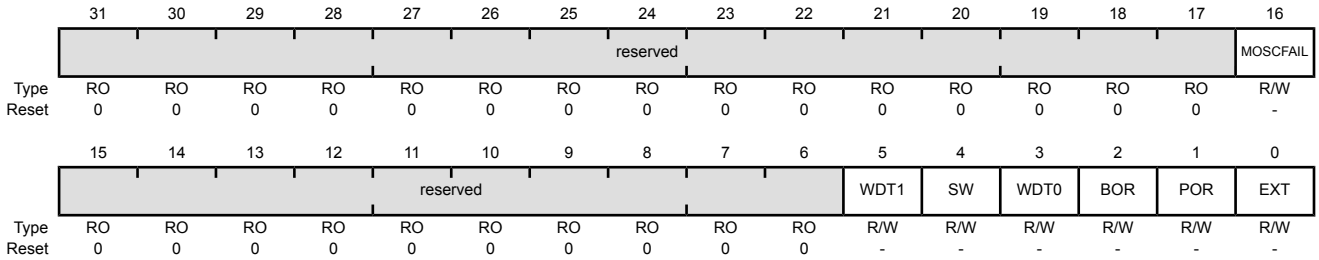
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:2 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | BORMIS | R/W1C | 0 | BOR Masked Interrupt Status<br>Value Description<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.<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. |
| 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: 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, except when an power-on reset is the cause, in which case, all bits other than POR in the **RESC** register are cleared.

Reset Cause (RESC)

Base 0x400F.E000
Offset 0x05C
Type R/W, reset -




| 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 | R/W | - | MOSC Failure Reset<br><br>Value Description<br>1 When read, this bit indicates that the MOSC circuit was enabled for clock validation and failed, generating a reset event.<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. |
| 15:6 | 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 | WDT1 | R/W | - | Watchdog Timer 1 Reset<br><br>Value Description<br>1 When read, this bit indicates that Watchdog Timer 1 timed out and generated a reset.<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. |

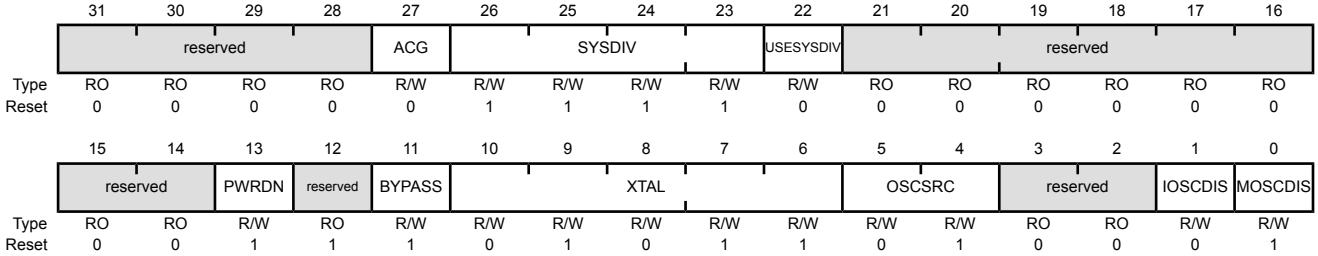
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | SW | R/W | - | Software Reset<br><br>Value Description<br>1 When read, this bit indicates that a software reset has caused a reset event.<br>0 When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset.<br>Writing a 0 to this bit clears it. |
| 3 | WDT0 | R/W | - | Watchdog Timer 0 Reset<br><br>Value Description<br>1 When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset.<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. |
| 2 | BOR | R/W | - | Brown-Out Reset<br><br>Value Description<br>1 When read, this bit indicates that a brown-out reset has caused a reset event.<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. |
| 1 | POR | R/W | - | Power-On Reset<br><br>Value Description<br>1 When read, this bit indicates that a power-on reset has caused a reset event.<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. |
| 0 | EXT | R/W | - | External Reset<br><br>Value Description<br>1 When read, this bit indicates that an external reset ($\overline{\text{RST}}$ assertion) has caused a reset event.<br>0 When read, this bit indicates that an external reset ($\overline{\text{RST}}$ assertion) has not caused a reset event since the previous power-on reset.<br>Writing a 0 to this bit clears it. |

## Register 7: Run-Mode Clock Configuration (RCC), offset 0x060

The bits in this register configure the system clock and oscillators.

Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type R/W, reset 0x0780.3AD1




| 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 | ACG | R/W | 0 | 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>1 The **SCGCn** or **DCGCn** registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The **SCGCn** and **DCGCn** registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode.<br>0 The **Run-Mode Clock Gating Control (RCGCn)** registers are used when the microcontroller enters a sleep mode.<br>The **RCGCn** registers are always used to control the clocks in Run mode. |
| 26:23 | SYSDIV | R/W | 0xF | System Clock Divisor<br>Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS bit in this register is configured). See Table 5-5 on page 179 for bit encodings.<br>If the SYSDIV value is less than MINSYSDIV (see page 217), and the PLL is being used, then the MINSYSDIV value is used as the divisor.<br>If the PLL is not being used, the SYSDIV value can be less than MINSYSDIV. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 22 | USESYSDIV | R/W | 0 | Enable System Clock Divider<br><br>Value Description<br>1 The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.<br>If the USERCC2 bit in the **RCC2** register is set, then the SYSDIV2 field in the **RCC2** register is used as the system clock divider rather than the SYSDIV field in this register.<br>0 The system clock is used undivided. |
| 21:14 | 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 | PWRDN | R/W | 1 | PLL Power Down<br><br>Value Description<br>1 The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.<br>0 The PLL is operating normally. |
| 12 | 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. |
| 11 | BYPASS | R/W | 1 | PLL Bypass<br><br>Value Description<br>1 The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV.<br>0 The system clock is the PLL output clock divided by the divisor specified by SYSDIV.<br><br>See Table 5-5 on page 179 for programming guidelines.<br><br>**Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 10:6 | XTAL | R/W | 0x0B | Crystal Value<br>This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below. Depending on the crystal used, the PLL frequency may not be exactly 400 MHz, see Table 19-8 on page 791 for more information. |

| Value | Crystal Frequency (MHz) Not Using the PLL | Crystal Frequency (MHz) Using the PLL |
|---|---|---|
| 0x00 | 1.000 MHz | reserved |
| 0x01 | 1.8432 MHz | reserved |
| 0x02 | 2.000 MHz | reserved |
| 0x03 | 2.4576 MHz | reserved |
| 0x04 | 3.579545 MHz | |
| 0x05 | 3.6864 MHz | |
| 0x06 | 4 MHz | |
| 0x07 | 4.096 MHz | |
| 0x08 | 4.9152 MHz | |
| 0x09 | 5 MHz | |
| 0x0A | 5.12 MHz | |
| 0x0B | 6 MHz (reset value) | |
| 0x0C | 6.144 MHz | |
| 0x0D | 7.3728 MHz | |
| 0x0E | 8 MHz | |
| 0x0F | 8.192 MHz | |
| 0x10 | 10.0 MHz | |
| 0x11 | 12.0 MHz | |
| 0x12 | 12.288 MHz | |
| 0x13 | 13.56 MHz | |
| 0x14 | 14.31818 MHz | |
| 0x15 | 16.0 MHz | |
| 0x16 | 16.384 MHz | |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | OSCSRC | R/W | 0x1 | Oscillator Source<br>Selects the input source for the OSC. The values are:<br><br>Value Input Source<br>0x0 MOSC<br>Main oscillator<br>0x1 PIOSC<br>Precision internal oscillator<br>(default)<br>0x2 PIOSC/4<br>Precision internal oscillator / 4<br>0x3 30 kHz<br>30-kHz internal oscillator<br><br>For additional oscillator sources, see the **RCC2** 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 | IOSCDIS | R/W | 0 | Precision Internal Oscillator Disable<br><br>Value Description<br>1 The precision internal oscillator (PIOSC) is disabled.<br>0 The precision internal oscillator is enabled. |
| 0 | MOSCDIS | R/W | 1 | Main Oscillator Disable<br><br>Value Description<br>1 The main oscillator is disabled (default).<br>0 The main oscillator is enabled. |

## Register 8: XTAL to PLL Translation (PLLCFG), offset 0x064

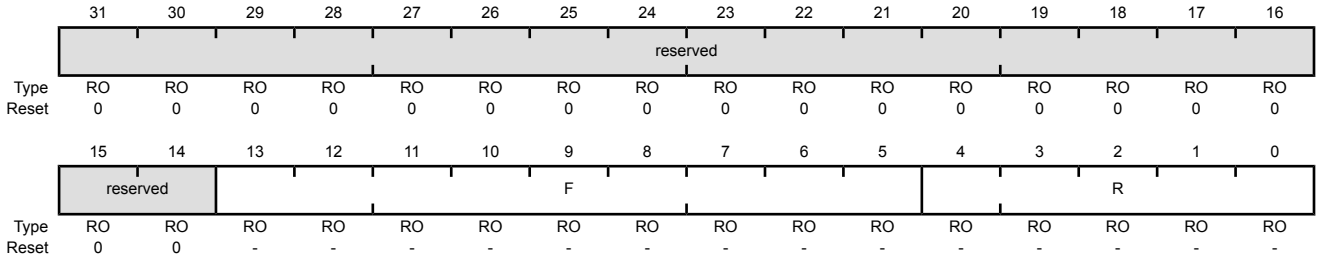
This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 198).

The PLL frequency is calculated using the **PLLCFG** field values, as follows:

```
PLLFreq = OSCFreq * F / (R + 1)
```

XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000
Offset 0x064
Type RO, reset -



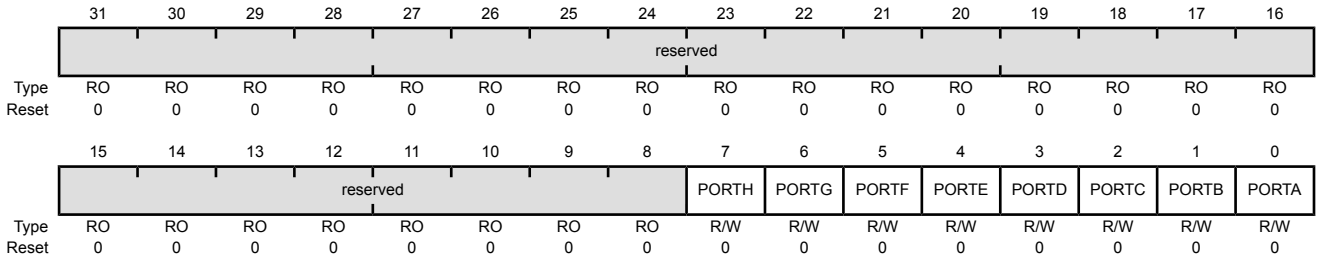

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:14 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 13:5 | F | RO | - | PLL F Value<br>This field specifies the value supplied to the PLL's F input. |
| 4:0 | R | RO | - | PLL R Value<br>This field specifies the value supplied to the PLL's R input. |

## Register 9: GPIO High-Performance Bus Control (GPIOHBCTL), offset 0x06C

This register controls which internal bus is used to access each GPIO port. When a bit is clear, the corresponding GPIO port is accessed across the legacy Advanced Peripheral Bus (APB) bus and through the APB memory aperture. When a bit is set, the corresponding port is accessed across the Advanced High-Performance Bus (AHB) bus and through the AHB memory aperture. Each GPIO port can be individually configured to use AHB or APB, but may be accessed only through one aperture. The AHB bus provides better back-to-back access performance than the APB bus. The address aperture in the memory map changes for the ports that are enabled for AHB access (see Table 9-7 on page 406).

GPIO High-Performance Bus Control (GPIOHBCTL)

Base 0x400F.E000
Offset 0x06C
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | 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. |
| 7 | PORTH | R/W | 0 | Port H Advanced High-Performance Bus<br>This bit defines the memory aperture for Port H.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 6 | PORTG | R/W | 0 | Port G Advanced High-Performance Bus<br>This bit defines the memory aperture for Port G.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 5 | PORTF | R/W | 0 | Port F Advanced High-Performance Bus<br>This bit defines the memory aperture for Port F.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | PORTE | R/W | 0 | Port E Advanced High-Performance Bus<br>This bit defines the memory aperture for Port E.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 3 | PORTD | R/W | 0 | Port D Advanced High-Performance Bus<br>This bit defines the memory aperture for Port D.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 2 | PORTC | R/W | 0 | Port C Advanced High-Performance Bus<br>This bit defines the memory aperture for Port C.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 1 | PORTB | R/W | 0 | Port B Advanced High-Performance Bus<br>This bit defines the memory aperture for Port B.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 0 | PORTA | R/W | 0 | Port A Advanced High-Performance Bus<br>This bit defines the memory aperture for Port A.<br><br>Value Description<br>1 Advanced High-Performance Bus (AHB)<br>0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |

## Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070

This register overrides the **RCC** equivalent register fields, as shown in Table 5-9, when the USERCC2 bit is set, allowing the extended capabilities of the **RCC2** register to be used while also providing a means to be backward-compatible to previous parts. Each **RCC2** field that supersedes an **RCC** field is located at the same LSB bit position; however, some **RCC2** fields are larger than the corresponding **RCC** field.

**Table 5-9. RCC2 Fields that Override RCC Fields**

| RCC2 Field... | Overrides RCC Field |
|---|---|
| SYSDIV2, bits[28:23] | SYSDIV, bits[26:23] |
| PWRDN2, bit[13] | PWRDN, bit[13] |
| BYPASS2, bit[11] | BYPASS, bit[11] |
| OSCSRC2, bits[6:4] | OSCSRC, bits[5:4] |

Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type R/W, reset 0x07C0.6810

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | USERCC2 | DIV400 | reserved | SYSDIV2 | | | | | | SYSDIV2LSB | reserved | | | | | |
| Type | R/W | R/W | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | PWRDN2 | reserved | BYPASS2 | reserved | | | | OSCSRC2 | | | reserved | | | |
| Type | RO | RO | R/W | RO | R/W | RO | RO | RO | RO | R/W | R/W | R/W | RO | RO | RO | RO |
| Reset | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | USERCC2 | R/W | 0 | Use **RCC2**<br><br>Value Description<br>1 The **RCC2** register fields override the **RCC** register fields.<br>0 The **RCC** register fields are used, and the fields in **RCC2** are ignored. |
| 30 | DIV400 | R/W | 0 | Divide PLL as 400 MHz vs. 200 MHz<br>This bit, along with the SYSDIV2LSB bit, allows additional frequency choices.<br><br>Value Description<br>1 Append the SYSDIV2LSB bit to the SYSDIV2 field to create a 7 bit divisor using the 400 MHz PLL output, see Table 5-7 on page 180.<br>0 Use SYSDIV2 as is and apply to 200 MHz predivided PLL output. See Table 5-6 on page 179 for programming guidelines. |
| 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 28:23 | SYSDIV2 | R/W | 0x0F | System Clock Divisor 2<br>Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS2 bit is configured). SYSDIV2 is used for the divisor when both the USESYSDIV bit in the **RCC** register and the USERCC2 bit in this register are set. See Table 5-6 on page 179 for programming guidelines. |
| 22 | SYSDIV2LSB | R/W | 1 | Additional LSB for SYSDIV2<br>When DIV400 is set, this bit becomes the LSB of SYSDIV2. If DIV400 is clear, this bit is not used. See Table 5-6 on page 179 for programming guidelines.<br>This bit can only be set or cleared when DIV400 is set. |
| 21:14 | 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 | PWRDN2 | R/W | 1 | Power-Down PLL 2<br>Value Description<br>1 The PLL is powered down.<br>0 The PLL operates normally. |
| 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 | BYPASS2 | R/W | 1 | PLL Bypass 2<br>Value Description<br>1 The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV2.<br>0 The system clock is the PLL output clock divided by the divisor specified by SYSDIV2.<br>See Table 5-6 on page 179 for programming guidelines.<br>**Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly. |
| 10:7 | 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. |

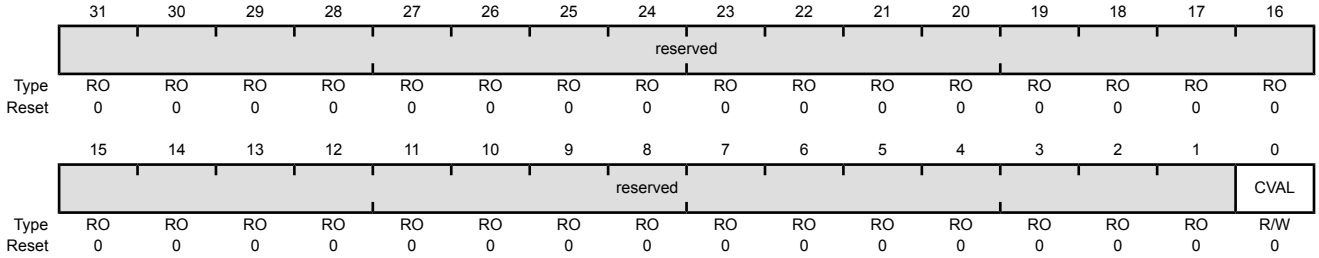
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6:4 | OSCSRC2 | R/W | 0x1 | Oscillator Source 2<br>Selects the input source for the OSC. The values are:<br><br>Value Description<br>0x0 MOSC<br>Main oscillator<br>0x1 PIOSC<br>Precision internal oscillator<br>0x2 PIOSC/4<br>Precision internal oscillator / 4<br>0x3 30 kHz<br>30-kHz internal oscillator<br>0x4-0x6 Reserved<br>0x7 32.768 kHz<br>32.768-kHz external oscillator |
| 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 11: Main Oscillator Control (MOSCCTL), offset 0x07C

This register provides the ability to enable the MOSC clock verification circuit. 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.

Main Oscillator Control (MOSCCTL)

Base 0x400F.E000
Offset 0x07C
Type R/W, 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 | CVAL | R/W | 0 | Clock Validation for MOSC |

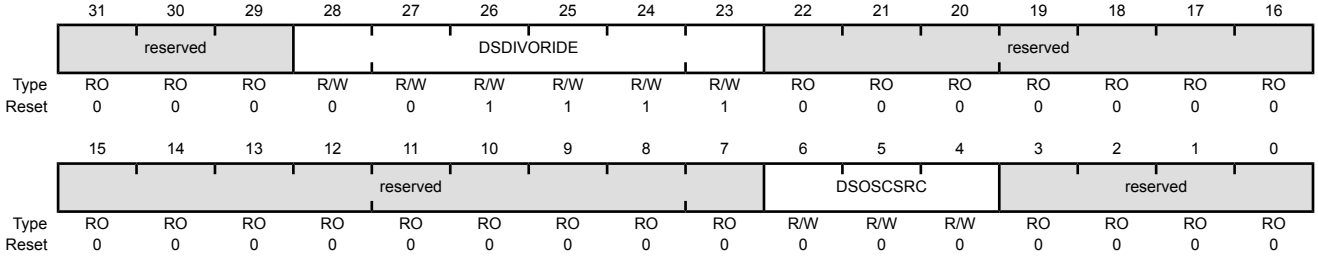
| Value | Description |
|---|---|
| 1 | The MOSC monitor circuit is enabled. |
| 0 | The MOSC monitor circuit is disabled. |

## Register 12: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144

This register provides configuration information for the hardware control of Deep Sleep Mode.

Deep Sleep Clock Configuration (DSLPCLKCFG)

Base 0x400F.E000
Offset 0x144
Type R/W, reset 0x0780.0000




| 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. |
| 28:23 | DSDIVORIDE | R/W | 0x0F | Divider Field Override<br>If Deep-Sleep mode is enabled when the PLL is running, the PLL is disabled. This 6-bit field contains a system divider field that overrides the SYSDIV field in the **RCC** register or the SYSDIV2 field in the **RCC2** register during Deep Sleep. This divider is applied to the source selected by the **DSOSCSRC** field.<br>Value Description<br>0x0 /1<br>0x1 /2<br>0x2 /3<br>0x3 /4<br>... ...<br>0x3F /64 |
| 22:7 | 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. |

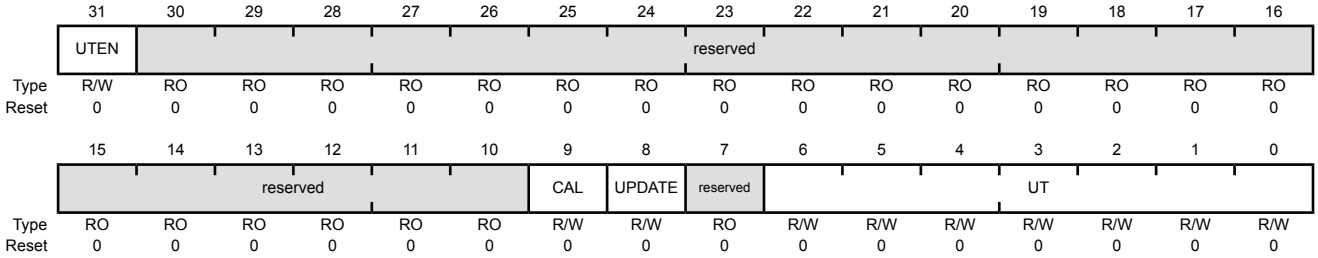
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6:4 | DSOSCSRC | R/W | 0x0 | Clock Source<br>Specifies the clock source during Deep-Sleep mode.<br><br>Value Description<br>0x0 MOSC<br>Use the main oscillator as the source.<br>**Note:** If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode.<br>0x1 PIOSC<br>Use the precision internal 16-MHz oscillator as the source.<br>0x2 Reserved<br>0x3 30 kHz<br>Use the 30-kHz internal oscillator as the source.<br>0x4-0x6 Reserved<br>0x7 32.768 kHz<br>Use the Hibernation module 32.768-kHz external oscillator as the source. |
| 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 13: 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | UTEN | R/W | 0 | Use User Trim Value<br><br>Value Description<br>1 The trim value in bits[6:0] of this register are used for any update trim operation.<br>0 The factory calibration value is used for an 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 | R/W | 0 | Start Calibration<br><br>Value Description<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>0 No action.<br><br>This bit is auto-cleared after it is set. |
| 8 | UPDATE | R/W | 0 | Update Trim<br><br>Value Description<br>1 Updates the PIOSC trim value with the UT bit or the DT bit in the **PIOSCSTAT** register. Used with UTEN.<br>0 No action.<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. |

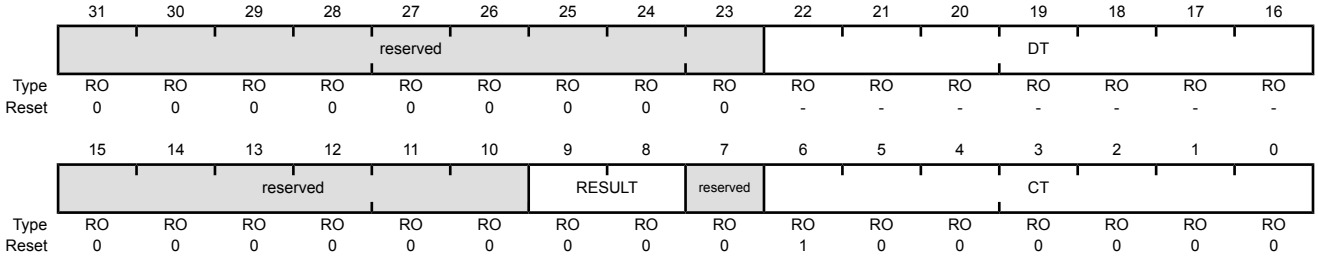
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6:0 | UT | R/W | 0x0 | User Trim Value<br>User trim value that can be loaded into the PIOSC.<br>Refer to “Main PLL Frequency Configuration” on page 181 for more information on calibrating the PIOSC. |

## Register 14: 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 0x0000.0040



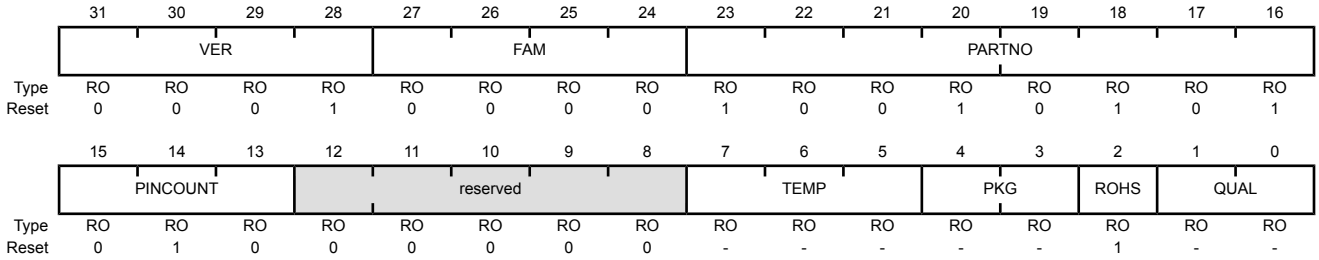

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:23 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 22:16 | DT | RO | - | 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 15: 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 -




| 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><br>Value Description<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 Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x0 Stellaris family of microcontollers, that is, all devices with external part numbers starting with LM3S. |
| 23:16 | PARTNO | RO | 0x95 | Part Number<br>This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved):<br><br>Value Description<br>0x95 LM3S1G58 |
| 15:13 | PINCOUNT | RO | 0x2 | 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>0x2 100-pin package |

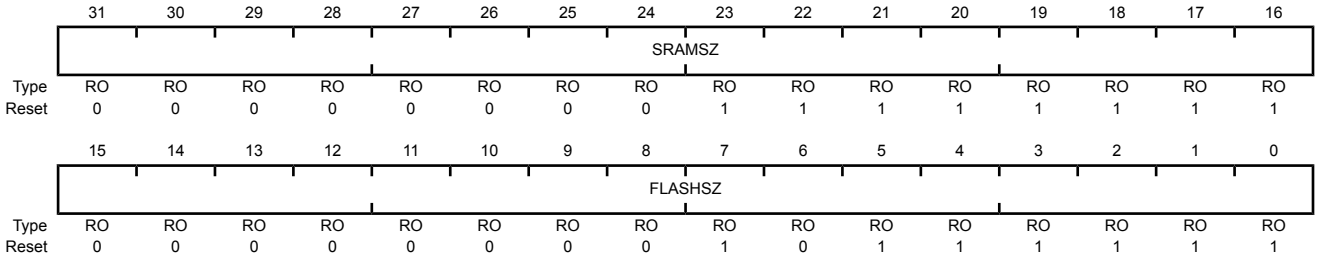
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12:8 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:5 | TEMP | RO | - | Temperature Range<br>This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved):<br>Value Description<br>0x0 Commercial temperature range (0°C to 70°C)<br>0x1 Industrial temperature range (-40°C to 85°C)<br>0x2 Extended temperature range (-40°C to 105°C) |
| 4:3 | PKG | RO | - | Package Type<br>This field specifies the package type. The value is encoded as follows (all other encodings are reserved):<br>Value Description<br>0x0 SOIC package<br>0x1 LQFP package<br>0x2 BGA package |
| 2 | ROHS | RO | 1 | RoHS-Compliance<br>This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant. |
| 1:0 | QUAL | RO | - | Qualification Status<br>This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved):<br>Value Description<br>0x0 Engineering Sample (unqualified)<br>0x1 Pilot Production (unqualified)<br>0x2 Fully Qualified |

## Register 16: Device Capabilities 0 (DC0), offset 0x008

This register is predefined by the part and can be used to verify features.

Device Capabilities 0 (DC0)

Base 0x400F.E000
Offset 0x008
Type RO, reset 0x00FF.00BF



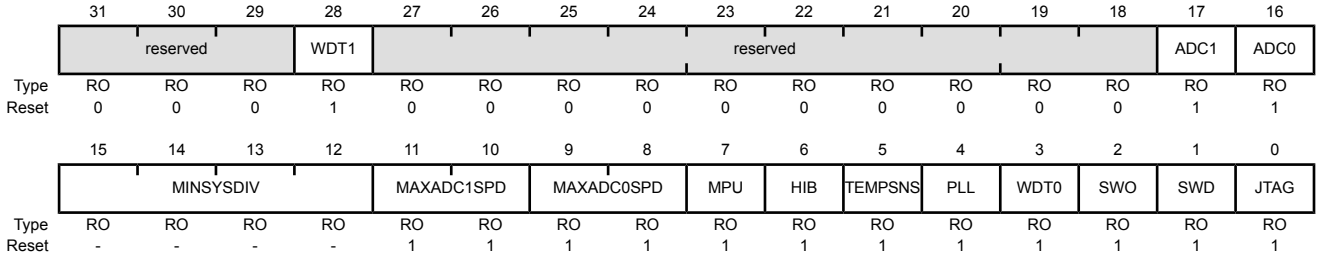

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | SRAMSZ | RO | 0x00FF | SRAM Size<br>Indicates the size of the on-chip SRAM memory.<br><br>Value Description<br>0x00FF 64 KB of SRAM |
| 15:0 | FLASHSZ | RO | 0x00BF | Flash Size<br>Indicates the size of the on-chip flash memory.<br><br>Value Description<br>0x00BF 384 KB of Flash |

## Register 17: Device Capabilities 1 (DC1), offset 0x010

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

Device Capabilities 1 (DC1)

Base 0x400F.E000
Offset 0x010
Type RO, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of 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 | WDT1 | RO | 1 | Watchdog Timer 1 Present<br>When set, indicates that watchdog timer 1 is present. |
| 27:18 | 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 | ADC1 | RO | 1 | ADC Module 1 Present<br>When set, indicates that ADC module 1 is present. |
| 16 | ADC0 | RO | 1 | ADC Module 0 Present<br>When set, indicates that ADC module 0 is present |
| 15:12 | MINSYSDIV | RO | - | System Clock Divider<br>Minimum 4-bit divider value for system clock. The reset value is hardware-dependent. See the RCC register for how to change the system clock divisor using the SYSDIV bit.<br>Value Description<br>0x1 Specifies an 80-MHz CPU clock with a PLL divider of 2.5.<br>0x2 Specifies a 66.67-MHz CPU clock with a PLL divider of 3.<br>0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4.<br>0x7 Specifies a 25-MHz clock with a PLL divider of 8.<br>0x9 Specifies a 20-MHz clock with a PLL divider of 10. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:10 | MAXADC1SPD | RO | 0x3 | Max ADC1 Speed<br>This field indicates the maximum rate at which the ADC samples data.<br><br>Value Description<br>0x3 1M samples/second |
| 9:8 | MAXADC0SPD | RO | 0x3 | Max ADC0 Speed<br>This field indicates the maximum rate at which the ADC samples data.<br><br>Value Description<br>0x3 1M samples/second |
| 7 | MPU | RO | 1 | MPU Present<br>When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the "Cortex-M3 Peripherals" chapter for details on the MPU. |
| 6 | HIB | RO | 1 | Hibernation Module Present<br>When set, indicates that the Hibernation module is present. |
| 5 | TEMPSNS | RO | 1 | Temp Sensor Present<br>When set, indicates that the on-chip temperature sensor is present. |
| 4 | PLL | RO | 1 | PLL Present<br>When set, indicates that the on-chip Phase Locked Loop (PLL) is present. |
| 3 | WDT0 | RO | 1 | Watchdog Timer 0 Present<br>When set, indicates that watchdog timer 0 is present. |
| 2 | SWO | RO | 1 | SWO Trace Port Present<br>When set, indicates that the Serial Wire Output (SWO) trace port is present. |
| 1 | SWD | RO | 1 | SWD Present<br>When set, indicates that the Serial Wire Debugger (SWD) is present. |
| 0 | JTAG | RO | 1 | JTAG Present<br>When set, indicates that the JTAG debugger interface is present. |

## Register 18: Device Capabilities 2 (DC2), offset 0x014

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

Device Capabilities 2 (DC2)

Base 0x400F.E000
Offset 0x014
Type RO, reset 0x000F.5037

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| 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 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | I2C1 | reserved | I2C0 | reserved | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |

| 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 | TIMER3 | RO | 1 | Timer Module 3 Present<br>When set, indicates that General-Purpose Timer module 3 is present. |
| 18 | TIMER2 | RO | 1 | Timer Module 2 Present<br>When set, indicates that General-Purpose Timer module 2 is present. |
| 17 | TIMER1 | RO | 1 | Timer Module 1 Present<br>When set, indicates that General-Purpose Timer module 1 is present. |
| 16 | TIMER0 | RO | 1 | Timer Module 0 Present<br>When set, indicates that General-Purpose Timer module 0 is present. |
| 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 | I2C1 | RO | 1 | I2C Module 1 Present<br>When set, indicates that I2C module 1 is present. |
| 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 | I2C0 | RO | 1 | I2C Module 0 Present<br>When set, indicates that I2C module 0 is present. |
| 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 | SSI1 | RO | 1 | SSI Module 1 Present<br>When set, indicates that SSI module 1 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | SSI0 | RO | 1 | SSI Module 0 Present<br>When set, indicates that SSI module 0 is present. |
| 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 | UART2 | RO | 1 | UART Module 2 Present<br>When set, indicates that UART module 2 is present. |
| 1 | UART1 | RO | 1 | UART Module 1 Present<br>When set, indicates that UART module 1 is present. |
| 0 | UART0 | RO | 1 | UART Module 0 Present<br>When set, indicates that UART module 0 is present. |

## Register 19: Device Capabilities 3 (DC3), offset 0x018

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

Device Capabilities 3 (DC3)

Base 0x400F.E000
Offset 0x018
Type RO, reset 0xBFFF.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 32KHZ | reserved | CCP5 | CCP4 | CCP3 | CCP2 | CCP1 | CCP0 | ADC0AIN7 | ADC0AIN6 | ADC0AIN5 | ADC0AIN4 | ADC0AIN3 | ADC0AIN2 | ADC0AIN1 | ADC0AIN0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 0 | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 | 32KHZ | RO | 1 | 32KHz Input Clock Available<br>When set, indicates an even CCP pin is present and can be used as a 32-KHz input clock. |
| 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 | CCP5 | RO | 1 | CCP5 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 5 is present. |
| 28 | CCP4 | RO | 1 | CCP4 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 4 is present. |
| 27 | CCP3 | RO | 1 | CCP3 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 3 is present. |
| 26 | CCP2 | RO | 1 | CCP2 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 2 is present. |
| 25 | CCP1 | RO | 1 | CCP1 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 1 is present. |
| 24 | CCP0 | RO | 1 | CCP0 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 0 is present. |
| 23 | ADC0AIN7 | RO | 1 | ADC Module 0 AIN7 Pin Present<br>When set, indicates that ADC module 0 input pin 7 is present. |
| 22 | ADC0AIN6 | RO | 1 | ADC Module 0 AIN6 Pin Present<br>When set, indicates that ADC module 0 input pin 6 is present. |
| 21 | ADC0AIN5 | RO | 1 | ADC Module 0 AIN5 Pin Present<br>When set, indicates that ADC module 0 input pin 5 is present. |

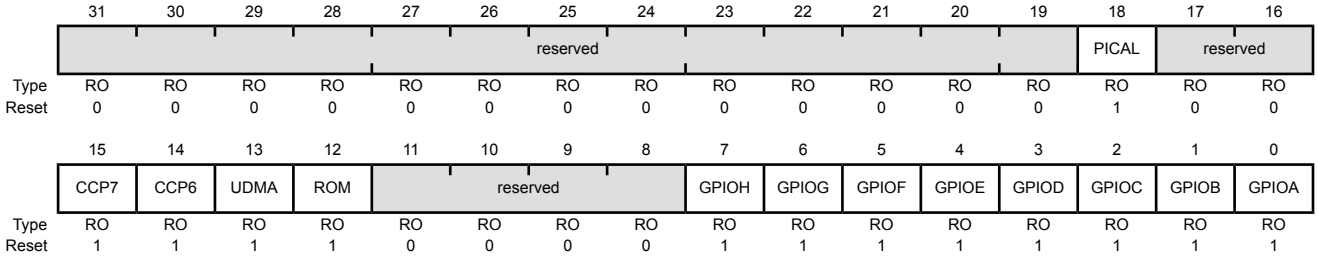
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 20 | ADC0AIN4 | RO | 1 | ADC Module 0 AIN4 Pin Present<br>When set, indicates that ADC module 0 input pin 4 is present. |
| 19 | ADC0AIN3 | RO | 1 | ADC Module 0 AIN3 Pin Present<br>When set, indicates that ADC module 0 input pin 3 is present. |
| 18 | ADC0AIN2 | RO | 1 | ADC Module 0 AIN2 Pin Present<br>When set, indicates that ADC module 0 input pin 2 is present. |
| 17 | ADC0AIN1 | RO | 1 | ADC Module 0 AIN1 Pin Present<br>When set, indicates that ADC module 0 input pin 1 is present. |
| 16 | ADC0AIN0 | RO | 1 | ADC Module 0 AIN0 Pin Present<br>When set, indicates that ADC module 0 input pin 0 is present. |
| 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 20: Device Capabilities 4 (DC4), offset 0x01C

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

Device Capabilities 4 (DC4)

Base 0x400F.E000
Offset 0x01C
Type RO, reset 0x0004.F0FF




| 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 | PICAL | RO | 1 | PIOSC Calibrate<br>When set, indicates that the PIOSC can be calibrated. |
| 17:16 | 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 | CCP7 | RO | 1 | CCP7 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 7 is present. |
| 14 | CCP6 | RO | 1 | CCP6 Pin Present<br>When set, indicates that Capture/Compare/PWM pin 6 is present. |
| 13 | UDMA | RO | 1 | Micro-DMA Module Present<br>When set, indicates that the micro-DMA module present. |
| 12 | ROM | RO | 1 | Internal Code ROM Present<br>When set, indicates that internal code ROM is present. |
| 11:8 | 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 | GPIOH | RO | 1 | GPIO Port H Present<br>When set, indicates that GPIO Port H is present. |
| 6 | GPIOG | RO | 1 | GPIO Port G Present<br>When set, indicates that GPIO Port G is present. |
| 5 | GPIOF | RO | 1 | GPIO Port F Present<br>When set, indicates that GPIO Port F is present. |

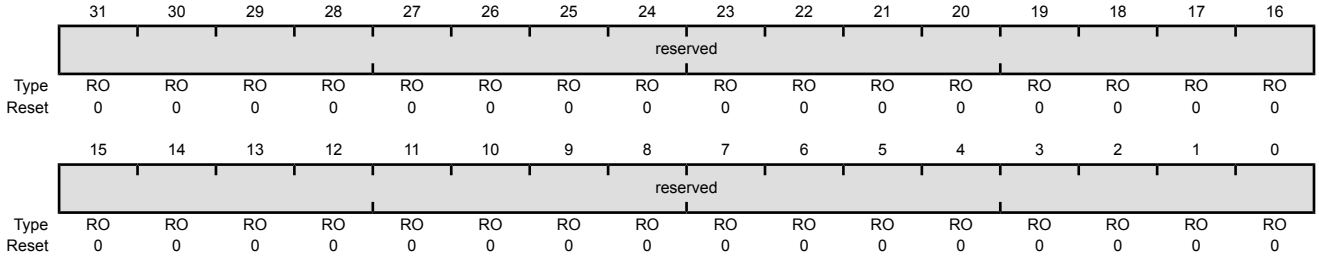
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | GPIOE | RO | 1 | GPIO Port E Present<br>When set, indicates that GPIO Port E is present. |
| 3 | GPIOD | RO | 1 | GPIO Port D Present<br>When set, indicates that GPIO Port D is present. |
| 2 | GPIOC | RO | 1 | GPIO Port C Present<br>When set, indicates that GPIO Port C is present. |
| 1 | GPIOB | RO | 1 | GPIO Port B Present<br>When set, indicates that GPIO Port B is present. |
| 0 | GPIOA | RO | 1 | GPIO Port A Present<br>When set, indicates that GPIO Port A is present. |

## Register 21: Device Capabilities 5 (DC5), offset 0x020

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

Device Capabilities 5 (DC5)

Base 0x400F.E000
Offset 0x020
Type RO, reset 0x0000.0000



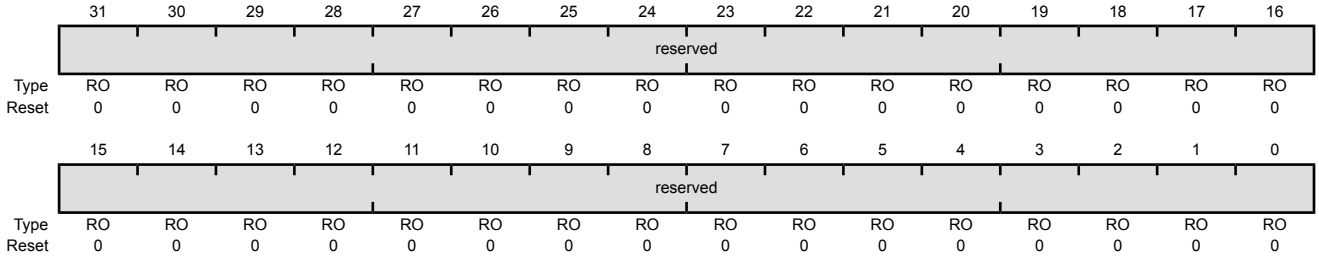

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

## Register 22: Device Capabilities 6 (DC6), offset 0x024

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

Device Capabilities 6 (DC6)

Base 0x400F.E000
Offset 0x024
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | 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 23: Device Capabilities 7 (DC7), offset 0x028

This register is predefined by the part and can be used to verify uDMA channel features. A 1 indicates the channel is available on this device; a 0 that the channel is only available on other devices in the family. Most channels have primary and secondary assignments. If the primary function is not available on this microcontroller, the secondary function becomes the primary function. If the secondary function is not available, the primary function is the only option.

Device Capabilities 7 (DC7)

Base 0x400F.E000
Offset 0x028
Type RO, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | DMACH30 | DMACH29 | DMACH28 | DMACH27 | DMACH26 | DMACH25 | DMACH24 | DMACH23 | DMACH22 | DMACH21 | DMACH20 | DMACH19 | DMACH18 | DMACH17 | DMACH16 |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DMACH15 | DMACH14 | DMACH13 | DMACH12 | DMACH11 | DMACH10 | DMACH9 | DMACH8 | DMACH7 | DMACH6 | DMACH5 | DMACH4 | DMACH3 | DMACH2 | DMACH1 | DMACH0 |
| 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 | reserved | RO | 1 | Reserved<br>Reserved for uDMA channel 31. |
| 30 | DMACH30 | RO | 1 | SW<br>When set, indicates uDMA channel 30 is available for software transfers. |
| 29 | DMACH29 | RO | 1 | I2S0_TX / CAN1_TX<br>When set, indicates uDMA channel 29 is available and connected to the transmit path of I2S module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of CAN module 1 transmit. |
| 28 | DMACH28 | RO | 1 | I2S0_RX / CAN1_RX<br>When set, indicates uDMA channel 28 is available and connected to the receive path of I2S module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of CAN module 1 receive. |
| 27 | DMACH27 | RO | 1 | CAN1_TX / ADC1_SS3<br>When set, indicates uDMA channel 27 is available and connected to the transmit path of CAN module 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of ADC module 1 Sample Sequencer 3. |
| 26 | DMACH26 | RO | 1 | CAN1_RX / ADC1_SS2<br>When set, indicates uDMA channel 26 is available and connected to the receive path of CAN module 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of ADC module 1 Sample Sequencer 2. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 25 | DMACH25 | RO | 1 | SSI1_TX / ADC1_SS1<br>When set, indicates uDMA channel 25 is available and connected to the transmit path of SSI module 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of ADC module 1 Sample Sequencer 1. |
| 24 | DMACH24 | RO | 1 | SSI1_RX / ADC1_SS0<br>When set, indicates uDMA channel 24 is available and connected to the receive path of SSI module 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of ADC module 1 Sample Sequencer 0. |
| 23 | DMACH23 | RO | 1 | UART1_TX / CAN2_TX<br>When set, indicates uDMA channel 23 is available and connected to the transmit path of UART module 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of CAN module 2 transmit. |
| 22 | DMACH22 | RO | 1 | UART1_RX / CAN2_RX<br>When set, indicates uDMA channel 22 is available and connected to the receive path of UART module 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of CAN module 2 receive. |
| 21 | DMACH21 | RO | 1 | Timer1B / EPI0_WFIFO<br>When set, indicates uDMA channel 21 is available and connected to Timer 1B. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of EPI module 0 write FIFO (WRIFO). |
| 20 | DMACH20 | RO | 1 | Timer1A / EPI0_NBRFIFO<br>When set, indicates uDMA channel 20 is available and connected to Timer 1A. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of EPI module 0 non-blocking read FIFO (NBRFIFO). |
| 19 | DMACH19 | RO | 1 | Timer0B / Timer1B<br>When set, indicates uDMA channel 19 is available and connected to Timer 0B. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 1B. |
| 18 | DMACH18 | RO | 1 | Timer0A / Timer1A<br>When set, indicates uDMA channel 18 is available and connected to Timer 0A. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 1A. |
| 17 | DMACH17 | RO | 1 | ADC0_SS3<br>When set, indicates uDMA channel 17 is available and connected to ADC module 0 Sample Sequencer 3. |
| 16 | DMACH16 | RO | 1 | ADC0_SS2<br>When set, indicates uDMA channel 16 is available and connected to ADC module 0 Sample Sequencer 2. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | DMACH15 | RO | 1 | ADC0_SS1 / Timer2B<br>When set, indicates uDMA channel 15 is available and connected to ADC module 0 Sample Sequencer 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 2B. |
| 14 | DMACH14 | RO | 1 | ADC0_SS0 / Timer2A<br>When set, indicates uDMA channel 14 is available and connected to ADC module 0 Sample Sequencer 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 2A. |
| 13 | DMACH13 | RO | 1 | CAN0_TX / UART2_TX<br>When set, indicates uDMA channel 13 is available and connected to the transmit path of CAN module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of UART module 2 transmit. |
| 12 | DMACH12 | RO | 1 | CAN0_RX / UART2_RX<br>When set, indicates uDMA channel 12 is available and connected to the receive path of CAN module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of UART module 2 receive. |
| 11 | DMACH11 | RO | 1 | SSI0_TX / SSI1_TX<br>When set, indicates uDMA channel 11 is available and connected to the transmit path of SSI module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of SSI module 1 transmit. |
| 10 | DMACH10 | RO | 1 | SSI0_RX / SSI1_RX<br>When set, indicates uDMA channel 10 is available and connected to the receive path of SSI module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of SSI module 1 receive. |
| 9 | DMACH9 | RO | 1 | UART0_TX / UART1_TX<br>When set, indicates uDMA channel 9 is available and connected to the transmit path of UART module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of UART module 1 transmit. |
| 8 | DMACH8 | RO | 1 | UART0_RX / UART1_RX<br>When set, indicates uDMA channel 8 is available and connected to the receive path of UART module 0. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of UART module 1 receive. |
| 7 | DMACH7 | RO | 1 | ETH_TX / Timer2B<br>When set, indicates uDMA channel 7 is available and connected to the transmit path of the Ethernet module. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 2B. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | DMACH6 | RO | 1 | ETH_RX / Timer2A<br>When set, indicates uDMA channel 6 is available and connected to the receive path of the Ethernet module. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 2A. |
| 5 | DMACH5 | RO | 1 | USB_EP3_TX / Timer2B<br>When set, indicates uDMA channel 5 is available and connected to the transmit path of USB endpoint 3. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 2B. |
| 4 | DMACH4 | RO | 1 | USB_EP3_RX / Timer2A<br>When set, indicates uDMA channel 4 is available and connected to the receive path of USB endpoint 3. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 2A. |
| 3 | DMACH3 | RO | 1 | USB_EP2_TX / Timer3B<br>When set, indicates uDMA channel 3 is available and connected to the transmit path of USB endpoint 2. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 3B. |
| 2 | DMACH2 | RO | 1 | USB_EP2_RX / Timer3A<br>When set, indicates uDMA channel 2 is available and connected to the receive path of USB endpoint 2. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of Timer 3A. |
| 1 | DMACH1 | RO | 1 | USB_EP1_TX / UART2_TX<br>When set, indicates uDMA channel 1 is available and connected to the transmit path of USB endpoint 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of UART module 2 transmit. |
| 0 | DMACH0 | RO | 1 | USB_EP1_RX / UART2_RX<br>When set, indicates uDMA channel 0 is available and connected to the receive path of USB endpoint 1. If the corresponding bit in the **DMACHASGN** register is set, the channel is connected instead to the secondary channel assignment of UART module 2 receive. |

## Register 24: Device Capabilities 8 ADC Channels (DC8), offset 0x02C

This register is predefined by the part and can be used to verify features.

Device Capabilities 8 ADC Channels (DC8)

Base 0x400F.E000
Offset 0x02C
Type RO, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADC1AIN15 | ADC1AIN14 | ADC1AIN13 | ADC1AIN12 | ADC1AIN11 | ADC1AIN10 | ADC1AIN9 | ADC1AIN8 | ADC1AIN7 | ADC1AIN6 | ADC1AIN5 | ADC1AIN4 | ADC1AIN3 | ADC1AIN2 | ADC1AIN1 | ADC1AIN0 |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADC0AIN15 | ADC0AIN14 | ADC0AIN13 | ADC0AIN12 | ADC0AIN11 | ADC0AIN10 | ADC0AIN9 | ADC0AIN8 | ADC0AIN7 | ADC0AIN6 | ADC0AIN5 | ADC0AIN4 | ADC0AIN3 | ADC0AIN2 | ADC0AIN1 | ADC0AIN0 |
| 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 | ADC1AIN15 | RO | 1 | ADC Module 1 AIN15 Pin Present<br>When set, indicates that ADC module 1 input pin 15 is present. |
| 30 | ADC1AIN14 | RO | 1 | ADC Module 1 AIN14 Pin Present<br>When set, indicates that ADC module 1 input pin 14 is present. |
| 29 | ADC1AIN13 | RO | 1 | ADC Module 1 AIN13 Pin Present<br>When set, indicates that ADC module 1 input pin 13 is present. |
| 28 | ADC1AIN12 | RO | 1 | ADC Module 1 AIN12 Pin Present<br>When set, indicates that ADC module 1 input pin 12 is present. |
| 27 | ADC1AIN11 | RO | 1 | ADC Module 1 AIN11 Pin Present<br>When set, indicates that ADC module 1 input pin 11 is present. |
| 26 | ADC1AIN10 | RO | 1 | ADC Module 1 AIN10 Pin Present<br>When set, indicates that ADC module 1 input pin 10 is present. |
| 25 | ADC1AIN9 | RO | 1 | ADC Module 1 AIN9 Pin Present<br>When set, indicates that ADC module 1 input pin 9 is present. |
| 24 | ADC1AIN8 | RO | 1 | ADC Module 1 AIN8 Pin Present<br>When set, indicates that ADC module 1 input pin 8 is present. |
| 23 | ADC1AIN7 | RO | 1 | ADC Module 1 AIN7 Pin Present<br>When set, indicates that ADC module 1 input pin 7 is present. |
| 22 | ADC1AIN6 | RO | 1 | ADC Module 1 AIN6 Pin Present<br>When set, indicates that ADC module 1 input pin 6 is present. |
| 21 | ADC1AIN5 | RO | 1 | ADC Module 1 AIN5 Pin Present<br>When set, indicates that ADC module 1 input pin 5 is present. |
| 20 | ADC1AIN4 | RO | 1 | ADC Module 1 AIN4 Pin Present<br>When set, indicates that ADC module 1 input pin 4 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 19 | ADC1AIN3 | RO | 1 | ADC Module 1 AIN3 Pin Present<br>When set, indicates that ADC module 1 input pin 3 is present. |
| 18 | ADC1AIN2 | RO | 1 | ADC Module 1 AIN2 Pin Present<br>When set, indicates that ADC module 1 input pin 2 is present. |
| 17 | ADC1AIN1 | RO | 1 | ADC Module 1 AIN1 Pin Present<br>When set, indicates that ADC module 1 input pin 1 is present. |
| 16 | ADC1AIN0 | RO | 1 | ADC Module 1 AIN0 Pin Present<br>When set, indicates that ADC module 1 input pin 0 is present. |
| 15 | ADC0AIN15 | RO | 1 | ADC Module 0 AIN15 Pin Present<br>When set, indicates that ADC module 0 input pin 15 is present. |
| 14 | ADC0AIN14 | RO | 1 | ADC Module 0 AIN14 Pin Present<br>When set, indicates that ADC module 0 input pin 14 is present. |
| 13 | ADC0AIN13 | RO | 1 | ADC Module 0 AIN13 Pin Present<br>When set, indicates that ADC module 0 input pin 13 is present. |
| 12 | ADC0AIN12 | RO | 1 | ADC Module 0 AIN12 Pin Present<br>When set, indicates that ADC module 0 input pin 12 is present. |
| 11 | ADC0AIN11 | RO | 1 | ADC Module 0 AIN11 Pin Present<br>When set, indicates that ADC module 0 input pin 11 is present. |
| 10 | ADC0AIN10 | RO | 1 | ADC Module 0 AIN10 Pin Present<br>When set, indicates that ADC module 0 input pin 10 is present. |
| 9 | ADC0AIN9 | RO | 1 | ADC Module 0 AIN9 Pin Present<br>When set, indicates that ADC module 0 input pin 9 is present. |
| 8 | ADC0AIN8 | RO | 1 | ADC Module 0 AIN8 Pin Present<br>When set, indicates that ADC module 0 input pin 8 is present. |
| 7 | ADC0AIN7 | RO | 1 | ADC Module 0 AIN7 Pin Present<br>When set, indicates that ADC module 0 input pin 7 is present. |
| 6 | ADC0AIN6 | RO | 1 | ADC Module 0 AIN6 Pin Present<br>When set, indicates that ADC module 0 input pin 6 is present. |
| 5 | ADC0AIN5 | RO | 1 | ADC Module 0 AIN5 Pin Present<br>When set, indicates that ADC module 0 input pin 5 is present. |
| 4 | ADC0AIN4 | RO | 1 | ADC Module 0 AIN4 Pin Present<br>When set, indicates that ADC module 0 input pin 4 is present. |
| 3 | ADC0AIN3 | RO | 1 | ADC Module 0 AIN3 Pin Present<br>When set, indicates that ADC module 0 input pin 3 is present. |
| 2 | ADC0AIN2 | RO | 1 | ADC Module 0 AIN2 Pin Present<br>When set, indicates that ADC module 0 input pin 2 is present. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | ADC0AIN1 | RO | 1 | ADC Module 0 AIN1 Pin Present<br>When set, indicates that ADC module 0 input pin 1 is present. |
| 0 | ADC0AIN0 | RO | 1 | ADC Module 0 AIN0 Pin Present<br>When set, indicates that ADC module 0 input pin 0 is present. |

## Register 25: Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190

This register is predefined by the part and can be used to verify features.

Device Capabilities 9 ADC Digital Comparators (DC9)

Base 0x400F.E000
Offset 0x190
Type RO, reset 0x00FF.00FF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | ADC1DC7 | ADC1DC6 | ADC1DC5 | ADC1DC4 | ADC1DC3 | ADC1DC2 | ADC1DC1 | ADC1DC0 |
| 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | ADC0DC7 | ADC0DC6 | ADC0DC5 | ADC0DC4 | ADC0DC3 | ADC0DC2 | ADC0DC1 | ADC0DC0 |
| 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 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:24 | 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 | ADC1DC7 | RO | 1 | ADC1 DC7 Present<br>When set, indicates that ADC module 1 Digital Comparator 7 is present. |
| 22 | ADC1DC6 | RO | 1 | ADC1 DC6 Present<br>When set, indicates that ADC module 1 Digital Comparator 6 is present. |
| 21 | ADC1DC5 | RO | 1 | ADC1 DC5 Present<br>When set, indicates that ADC module 1 Digital Comparator 5 is present. |
| 20 | ADC1DC4 | RO | 1 | ADC1 DC4 Present<br>When set, indicates that ADC module 1 Digital Comparator 4 is present. |
| 19 | ADC1DC3 | RO | 1 | ADC1 DC3 Present<br>When set, indicates that ADC module 1 Digital Comparator 3 is present. |
| 18 | ADC1DC2 | RO | 1 | ADC1 DC2 Present<br>When set, indicates that ADC module 1 Digital Comparator 2 is present. |
| 17 | ADC1DC1 | RO | 1 | ADC1 DC1 Present<br>When set, indicates that ADC module 1 Digital Comparator 1 is present. |
| 16 | ADC1DC0 | RO | 1 | ADC1 DC0 Present<br>When set, indicates that ADC module 1 Digital Comparator 0 is present. |
| 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. |
| 7 | ADC0DC7 | RO | 1 | ADC0 DC7 Present<br>When set, indicates that ADC module 0 Digital Comparator 7 is present. |

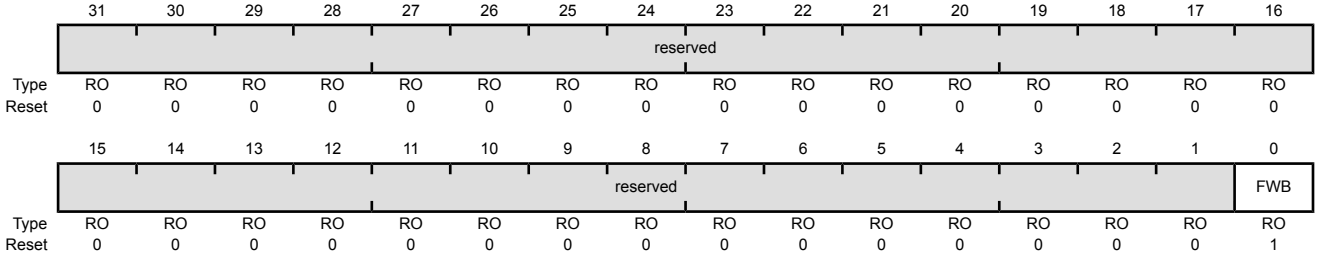
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 6 | ADC0DC6 | RO | 1 | ADC0 DC6 Present<br>When set, indicates that ADC module 0 Digital Comparator 6 is present. |
| 5 | ADC0DC5 | RO | 1 | ADC0 DC5 Present<br>When set, indicates that ADC module 0 Digital Comparator 5 is present. |
| 4 | ADC0DC4 | RO | 1 | ADC0 DC4 Present<br>When set, indicates that ADC module 0 Digital Comparator 4 is present. |
| 3 | ADC0DC3 | RO | 1 | ADC0 DC3 Present<br>When set, indicates that ADC module 0 Digital Comparator 3 is present. |
| 2 | ADC0DC2 | RO | 1 | ADC0 DC2 Present<br>When set, indicates that ADC module 0 Digital Comparator 2 is present. |
| 1 | ADC0DC1 | RO | 1 | ADC0 DC1 Present<br>When set, indicates that ADC module 0 Digital Comparator 1 is present. |
| 0 | ADC0DC0 | RO | 1 | ADC0 DC0 Present<br>When set, indicates that ADC module 0 Digital Comparator 0 is present. |

## Register 26: 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 | 1 | 32 Word Flash Write Buffer Active<br>When set, indicates that the 32 word Flash memory write buffer feature is active. |

## Register 27: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Run Mode Clock Gating Control Register 0 (RCGC0)

Base 0x400F.E000
Offset 0x100
Type R/W, reset 0x00000040

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | WDT1 | reserved | | | | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| 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 | | | | MAXADC1SPD | | MAXADC0SPD | | reserved | HIB | reserved | | WDT0 | reserved | | |
| Type | RO | RO | RO | RO | R/W | R/W | R/W | R/W | RO | R/W | RO | RO | R/W | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of 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 | WDT1 | R/W | 0 | WDT1 Clock Gating Control<br>This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 27:18 | 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 | ADC1 | R/W | 0 | ADC1 Clock Gating Control<br>This bit controls the clock gating for SAR ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 16 | ADC0 | R/W | 0 | ADC0 Clock Gating Control<br>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| 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. |
| 11:10 | MAXADC1SPD | R/W | 0 | ADC1 Sample Speed<br>This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved):<br>Value Description<br>0x3 1M samples/second<br>0x2 500K samples/second<br>0x1 250K samples/second<br>0x0 125K samples/second |
| 9:8 | MAXADC0SPD | R/W | 0 | ADC0 Sample Speed<br>This field sets the rate at which ADC0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved):<br>Value Description<br>0x3 1M samples/second<br>0x2 500K samples/second<br>0x1 250K samples/second<br>0x0 125K samples/second |
| 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 | HIB | R/W | 1 | HIB Clock Gating Control<br>This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 5:4 | 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 | R/W | 0 | WDT0 Clock Gating Control<br>This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 28: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

This register controls the clock gating logic in Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Sleep Mode Clock Gating Control Register 0 (SCGC0)

Base 0x400F.E000
Offset 0x110
Type R/W, reset 0x00000040

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | WDT1 | reserved | | | | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| 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 | | | | MAXADC1SPD | | MAXADC0SPD | | reserved | HIB | reserved | | WDT0 | reserved | | |
| Type | RO | RO | RO | RO | R/W | R/W | R/W | R/W | RO | R/W | RO | RO | R/W | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of 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 | WDT1 | R/W | 0 | WDT1 Clock Gating Control<br>This bit controls the clock gating for Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 27:18 | 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 | ADC1 | R/W | 0 | ADC1 Clock Gating Control<br>This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 16 | ADC0 | R/W | 0 | ADC0 Clock Gating Control<br>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| 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. |
| 11:10 | MAXADC1SPD | R/W | 0 | ADC1 Sample Speed<br>This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved):<br>Value Description<br>0x3 1M samples/second<br>0x2 500K samples/second<br>0x1 250K samples/second<br>0x0 125K samples/second |
| 9:8 | MAXADC0SPD | R/W | 0 | ADC0 Sample Speed<br>This field sets the rate at which ADC module 0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved):<br>Value Description<br>0x3 1M samples/second<br>0x2 500K samples/second<br>0x1 250K samples/second<br>0x0 125K samples/second |
| 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 | HIB | R/W | 1 | HIB Clock Gating Control<br>This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 5:4 | 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 | R/W | 0 | WDT0 Clock Gating Control<br>This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 29: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

This register controls the clock gating logic in Deep-Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

Base 0x400F.E000
Offset 0x120
Type R/W, reset 0x00000040

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | WDT1 | reserved | | | | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| 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 | | | | | | | | | HIB | reserved | | WDT0 | reserved | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | RO | RO | R/W | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of 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 | WDT1 | R/W | 0 | WDT1 Clock Gating Control<br>This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 27:18 | 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 | ADC1 | R/W | 0 | ADC1 Clock Gating Control<br>This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 16 | ADC0 | R/W | 0 | ADC0 Clock Gating Control<br>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15:7 | 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 | HIB | R/W | 1 | HIB Clock Gating Control<br>This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 5:4 | 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 | R/W | 0 | WDT0 Clock Gating Control<br>This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 30: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Run Mode Clock Gating Control Register 1 (RCGC1)

Base 0x400F.E000
Offset 0x104
Type R/W, reset 0x00000000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| 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 | I2C1 | reserved | I2C0 | reserved | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | R/W | R/W | R/W |
| 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 | 0 | Software should not rely on the value of 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 | TIMER3 | R/W | 0 | Timer 3 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 18 | TIMER2 | R/W | 0 | Timer 2 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 17 | TIMER1 | R/W | 0 | Timer 1 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 16 | TIMER0 | R/W | 0 | Timer 0 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

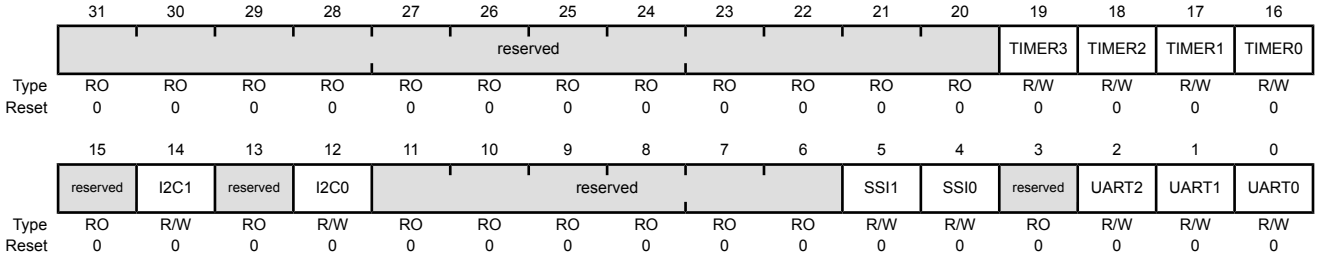
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 15 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 14 | I2C1 | R/W | 0 | I2C1 Clock Gating Control<br>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | I2C0 | R/W | 0 | I2C0 Clock Gating Control<br>This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | SSI1 | R/W | 0 | SSI1 Clock Gating Control<br>This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 4 | SSI0 | R/W | 0 | SSI0 Clock Gating Control<br>This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | UART2 | R/W | 0 | UART2 Clock Gating Control<br>This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 1 | UART1 | R/W | 0 | UART1 Clock Gating Control<br>This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 0 | UART0 | R/W | 0 | UART0 Clock Gating Control<br>This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

## Register 31: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

This register controls the clock gating logic in Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Sleep Mode Clock Gating Control Register 1 (SCGC1)

Base 0x400F.E000
Offset 0x114
Type R/W, reset 0x00000000




| 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 | TIMER3 | R/W | 0 | Timer 3 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 18 | TIMER2 | R/W | 0 | Timer 2 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 17 | TIMER1 | R/W | 0 | Timer 1 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | TIMER0 | R/W | 0 | Timer 0 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | I2C1 | R/W | 0 | I2C1 Clock Gating Control<br>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | I2C0 | R/W | 0 | I2C0 Clock Gating Control<br>This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | SSI1 | R/W | 0 | SSI1 Clock Gating Control<br>This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 4 | SSI0 | R/W | 0 | SSI0 Clock Gating Control<br>This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | UART2 | R/W | 0 | UART2 Clock Gating Control<br>This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 1 | UART1 | R/W | 0 | UART1 Clock Gating Control<br>This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | UART0 | R/W | 0 | UART0 Clock Gating Control<br>This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

## Register 32: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

This register controls the clock gating logic in Deep-Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1)

Base 0x400F.E000
Offset 0x124
Type R/W, reset 0x00000000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| 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 | I2C1 | reserved | I2C0 | reserved | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | R/W | R/W | R/W |
| 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 | 0 | Software should not rely on the value of 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 | TIMER3 | R/W | 0 | Timer 3 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 18 | TIMER2 | R/W | 0 | Timer 2 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 17 | TIMER1 | R/W | 0 | Timer 1 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | TIMER0 | R/W | 0 | Timer 0 Clock Gating Control<br>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | I2C1 | R/W | 0 | I2C1 Clock Gating Control<br>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | I2C0 | R/W | 0 | I2C0 Clock Gating Control<br>This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | SSI1 | R/W | 0 | SSI1 Clock Gating Control<br>This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 4 | SSI0 | R/W | 0 | SSI0 Clock Gating Control<br>This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | UART2 | R/W | 0 | UART2 Clock Gating Control<br>This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 1 | UART1 | R/W | 0 | UART1 Clock Gating Control<br>This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | UART0 | R/W | 0 | UART0 Clock Gating Control<br>This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

## Register 33: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000
Offset 0x108
Type R/W, reset 0x00000000

| | 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 | | UDMA | reserved | | | | | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| Type | RO | RO | R/W | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 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 | UDMA | R/W | 0 | Micro-DMA Clock Gating Control<br>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 12:8 | 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 | GPIOH | R/W | 0 | Port H Clock Gating Control<br>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 6 | GPIOG | R/W | 0 | Port G Clock Gating Control<br>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

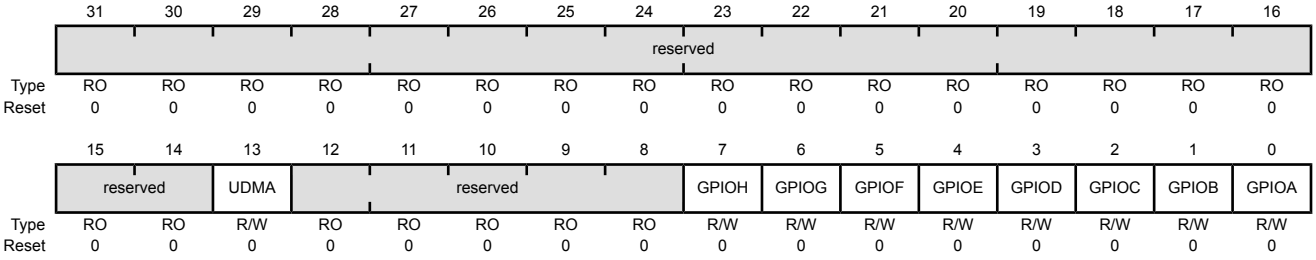
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | GPIOF | R/W | 0 | Port F Clock Gating Control<br>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 4 | GPIOE | R/W | 0 | Port E Clock Gating Control<br>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 3 | GPIOD | R/W | 0 | Port D Clock Gating Control<br>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 2 | GPIOC | R/W | 0 | Port C Clock Gating Control<br>This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 1 | GPIOB | R/W | 0 | Port B Clock Gating Control<br>This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 0 | GPIOA | R/W | 0 | Port A Clock Gating Control<br>This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

## Register 34: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

This register controls the clock gating logic in Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Sleep Mode Clock Gating Control Register 2 (SCGC2)

Base 0x400F.E000
Offset 0x118
Type R/W, reset 0x00000000




| 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 | UDMA | R/W | 0 | Micro-DMA Clock Gating Control<br>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 12:8 | 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 | GPIOH | R/W | 0 | Port H Clock Gating Control<br>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 6 | GPIOG | R/W | 0 | Port G Clock Gating Control<br>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | GPIOF | R/W | 0 | Port F Clock Gating Control<br>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 4 | GPIOE | R/W | 0 | Port E Clock Gating Control<br>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 3 | GPIOD | R/W | 0 | Port D Clock Gating Control<br>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 2 | GPIOC | R/W | 0 | Port C Clock Gating Control<br>This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 1 | GPIOB | R/W | 0 | Port B Clock Gating Control<br>This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 0 | GPIOA | R/W | 0 | Port A Clock Gating Control<br>This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

## Register 35: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

This register controls the clock gating logic in Deep-Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

Base 0x400F.E000
Offset 0x128
Type R/W, reset 0x00000000

| 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 | | UDMA | reserved | | | | | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| RO | RO | R/W | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | UDMA | R/W | 0 | Micro-DMA Clock Gating Control<br>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 12:8 | 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 | GPIOH | R/W | 0 | Port H Clock Gating Control<br>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 6 | GPIOG | R/W | 0 | Port G Clock Gating Control<br>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

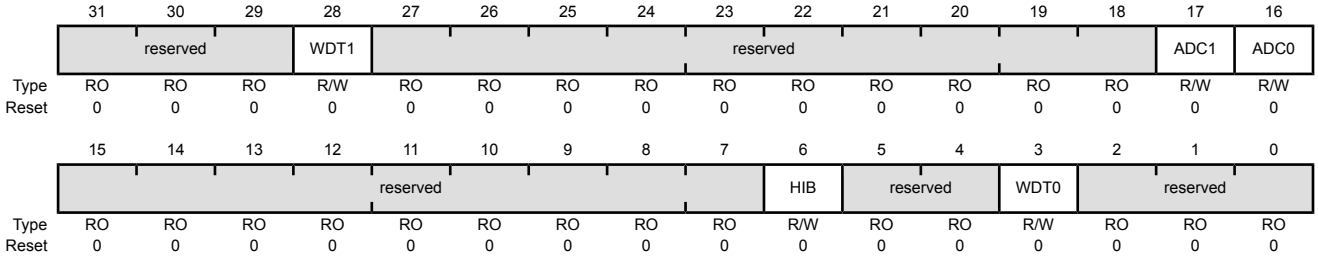
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | GPIOF | R/W | 0 | Port F Clock Gating Control<br>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 4 | GPIOE | R/W | 0 | Port E Clock Gating Control<br>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 3 | GPIOD | R/W | 0 | Port D Clock Gating Control<br>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 2 | GPIOC | R/W | 0 | Port C Clock Gating Control<br>This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 1 | GPIOB | R/W | 0 | Port B Clock Gating Control<br>This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 0 | GPIOA | R/W | 0 | Port A Clock Gating Control<br>This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |

## Register 36: Software Reset Control 0 (SRCR0), offset 0x040

This register allows individual modules to be reset. Writes to this register are masked by the bits in the **Device Capabilities 1 (DC1)** register.

Software Reset Control 0 (SRCR0)

Base 0x400F.E000
Offset 0x040
Type R/W, reset 0x00000000



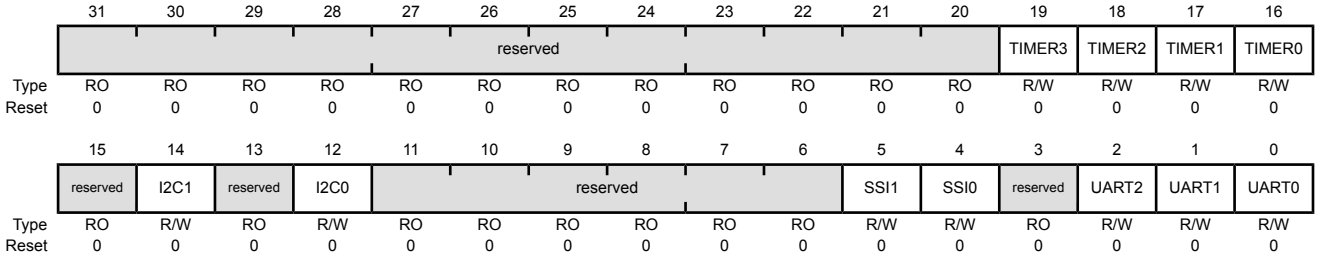

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | WDT1 | R/W | 0 | WDT1 Reset Control<br>When this bit is set, Watchdog Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 27:18 | 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 | ADC1 | R/W | 0 | ADC1 Reset Control<br>When this bit is set, ADC module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 16 | ADC0 | R/W | 0 | ADC0 Reset Control<br>When this bit is set, ADC module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 15:7 | 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 | HIB | R/W | 0 | HIB Reset Control<br>When this bit is set, the Hibernation module is reset. All internal data is lost and the registers are returned to their reset states.This bit must be manually cleared after being set. |
| 5:4 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | WDT0 | R/W | 0 | WDT0 Reset Control<br>When this bit is set, Watchdog Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 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 37: Software Reset Control 1 (SRCR1), offset 0x044

This register allows individual modules to be reset. Writes to this register are masked by the bits in the **Device Capabilities 2 (DC2)** register.

Software Reset Control 1 (SRCR1)
Base 0x400F.E000
Offset 0x044
Type R/W, reset 0x00000000




| 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 | TIMER3 | R/W | 0 | Timer 3 Reset Control<br>Timer 3 Reset Control. When this bit is set, General-Purpose Timer module 3 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 18 | TIMER2 | R/W | 0 | Timer 2 Reset Control<br>When this bit is set, General-Purpose Timer module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 17 | TIMER1 | R/W | 0 | Timer 1 Reset Control<br>When this bit is set, General-Purpose Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 16 | TIMER0 | R/W | 0 | Timer 0 Reset Control<br>When this bit is set, General-Purpose Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 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 | I2C1 | R/W | 0 | I2C1 Reset Control<br>When this bit is set, I2C module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 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. |

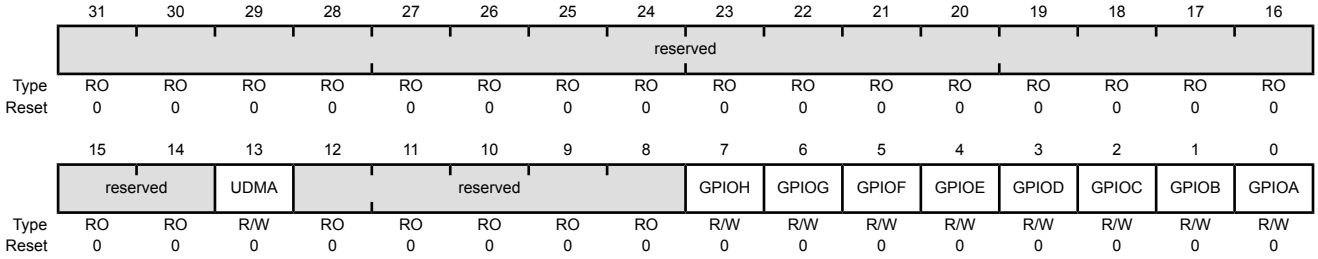
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | I2C0 | R/W | 0 | I2C0 Reset Control<br>When this bit is set, I2C module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 11:6 | 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 | SSI1 | R/W | 0 | SSI1 Reset Control<br>When this bit is set, SSI module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 4 | SSI0 | R/W | 0 | SSI0 Reset Control<br>When this bit is set, SSI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 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 | UART2 | R/W | 0 | UART2 Reset Control<br>When this bit is set, UART module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 1 | UART1 | R/W | 0 | UART1 Reset Control<br>When this bit is set, UART module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 0 | UART0 | R/W | 0 | UART0 Reset Control<br>When this bit is set, UART module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

## Register 38: Software Reset Control 2 (SRCR2), offset 0x048

This register allows individual modules to be reset. Writes to this register are masked by the bits in the **Device Capabilities 4 (DC4)** register.

Software Reset Control 2 (SRCR2)

Base 0x400F.E000
Offset 0x048
Type R/W, reset 0x00000000




| 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 | UDMA | R/W | 0 | Micro-DMA Reset Control<br>When this bit is set, uDMA module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 12:8 | 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 | GPIOH | R/W | 0 | Port H Reset Control<br>When this bit is set, Port H module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 6 | GPIOG | R/W | 0 | Port G Reset Control<br>When this bit is set, Port G module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 5 | GPIOF | R/W | 0 | Port F Reset Control<br>When this bit is set, Port F module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 4 | GPIOE | R/W | 0 | Port E Reset Control<br>When this bit is set, Port E module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 3 | GPIOD | R/W | 0 | Port D Reset Control<br>When this bit is set, Port D module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2 | GPIOC | R/W | 0 | Port C Reset Control<br>When this bit is set, Port C module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 1 | GPIOB | R/W | 0 | Port B Reset Control<br>When this bit is set, Port B module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 0 | GPIOA | R/W | 0 | Port A Reset Control<br>When this bit is set, Port A module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

# 6 Hibernation Module

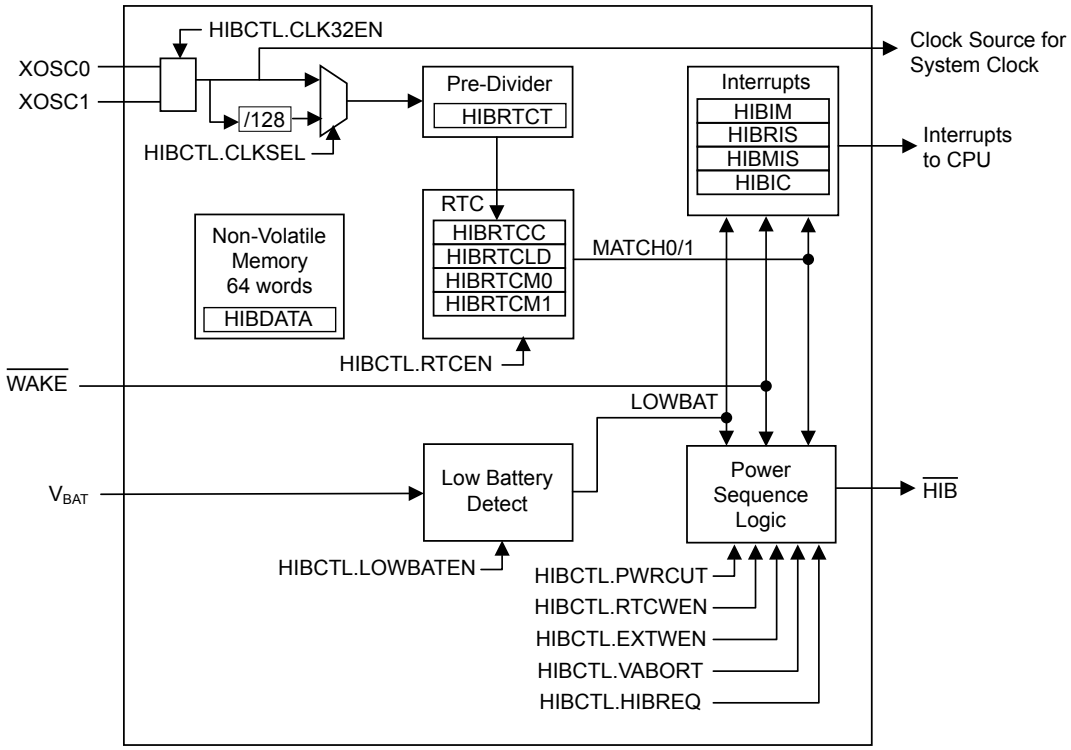
The Hibernation Module manages removal and restoration of power to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal or at a certain time using the built-in Real-Time Clock (RTC). The Hibernation module can be independently supplied from a battery or an auxiliary power supply.

The Hibernation module has the following features:

- 32-bit real-time counter (RTC)
  - Two 32-bit RTC match registers for timed wake-up and interrupt generation
  - RTC predivider trim for making fine adjustments to the clock rate
- Two mechanisms for power control
  - System power control using discrete external regulator
  - On-chip power control using internal switches under register control
- Dedicated pin for waking using an external signal
- RTC operational and hibernation memory valid as long as $V_{BAT}$ is valid
- Low-battery detection, signaling, and interrupt generation
- Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal; 32.768-kHz external oscillator can be used for main controller clock
- 64 32-bit words of non-volatile memory to save state during hibernation
- Programmable interrupts for RTC match, external wake, and low battery events

## 6.1 Block Diagram

**Figure 6-1. Hibernation Module Block Diagram**




## 6.2 Signal Description

The following table lists the external signals of the Hibernation module and describes the function of each. These signals have dedicated functions and are not alternate functions for any GPIO signals.

**Table 6-1. Signals for Hibernate (100LQFP)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| HIB | 51 | fixed | O | OD | An output that indicates the processor is in Hibernate mode. |
| VBAT | 55 | 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 | 50 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| XOSC0 | 52 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| XOSC1 | 53 | fixed | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 6-2. Signals for Hibernate (108BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| HIB | M12 | fixed | O | OD | An output that indicates the processor is in Hibernate mode. |
| VBAT | L12 | 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 | M10 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| XOSC0 | K11 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| XOSC1 | K12 | fixed | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 6.3 Functional Description

**Important:** The Hibernate module must have either the RTC function or the External Wake function enabled to ensure proper operation of the microcontroller. See “Initialization” on page 270.

The Hibernation module provides two mechanisms for power control:

- The first mechanism controls the power to the microcontroller with a control signal (HIB) that signals an external voltage regulator to turn on or off.
- The second mechanism uses internal switches to control power to the Cortex-M3 as well as to most analog and digital functions while retaining I/O pin power (VDD3ON mode).

The Hibernation module power source is determined dynamically. The supply voltage of the Hibernation module is the larger of the main voltage source ($V_{DD}$) or the battery/auxilliary voltage source ($V_{BAT}$). The Hibernation module also has an independent clock source to maintain a real-time clock (RTC) when the system clock is powered down. Once in hibernation, the module signals an external voltage regulator to turn the power back on when an external pin (WAKE) 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 when this occurs.

When waking from hibernation, the HIB signal is deasserted. The return of $V_{DD}$ causes a POR to be executed. The time from when the 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 ($T_{IRPOR}$).

### 6.3.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, certain 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 certain Hibernation registers or between a write followed by a read to those same registers. 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 affected register. The following registers are subject to this timing restriction:

- **Hibernation RTC Counter (HIBRTCC)**
- **Hibernation RTC Match 0 (HIBRTCM0)**
- **Hibernation RTC Match 1 (HIBRTCM1)**
- **Hibernation RTC Load (HIBRTCLD)**
- **Hibernation RTC Trim (HIBRTCT)**
- **Hibernation Data (HIBDATA)**

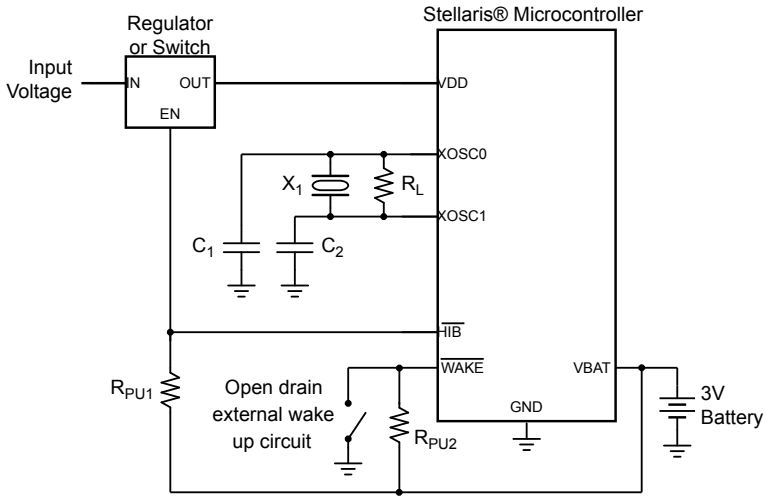
Back-to-back reads from Hibernation module registers have no timing restrictions. Reads are performed at the full peripheral clock rate.

### 6.3.2 Hibernation Clock Source

In systems where the Hibernation module is used to put the microcontroller into hibernation, the module must be clocked by an external source that is independent from the main system clock, even if the RTC feature is not used. An external oscillator or crystal is used for this purpose. To use a crystal, a 4.194304-MHz crystal is connected to the XOSC0 and XOSC1 pins. This clock signal is divided by 128 internally to produce a 32.768-kHz Hibernation clock reference. 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 6-2 on page 267 and Figure 6-3 on page 267. Note that these diagrams only show the connection to the Hibernation pins and not to the full system. See “Hibernation Module” on page 793 for specific values.

The Hibernation clock source is enabled by setting the CLK32EN bit of the **HIBCTL** register. The type of clock source is selected by clearing the CLKSEL bit for a 4.194304-MHz crystal and setting the CLKSEL bit for a 32.768-kHz oscillator. 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 oscillator is used for the clock source, no delay is needed.

**Figure 6-2. Using a Crystal as the Hibernation Clock Source**




**Note:** $X_1$ = Crystal frequency is $f_{XOSC_XTAL}$.

$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

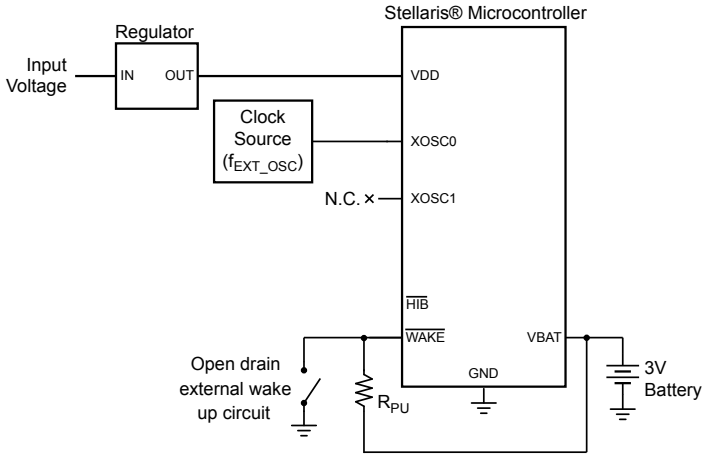
$R_L$ = Load resistor is $R_{XOSC_LOAD}$.

$R_{PU1}$ = Pull-up resistor 1 (value and voltage source ($V_{BAT}$ or Input Voltage) determined by regulator or switch enable input characteristics).

$R_{PU2}$ = Pull-up resistor 2 is 200 kΩ

See “Hibernation Module” on page 793 for specific parameter values.

**Figure 6-3. Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode**




**Note:** $R_{PU}$ = Pull-up resistor is 1 MΩ

If the application does not require the use of the Hibernation module, refer to “Connections for Unused Signals” on page 782 for more information on how to connect the unused signals. In this situation, the HIB bit in the **Run Mode Clock Gating Control Register 0 (RCGC0)** register must be cleared, disabling the system clock to the Hibernation module and Hibernation module registers are not accessible.

### 6.3.3 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. The module can monitor the voltage level of the battery and detect when the voltage drops below $V_{LOWBAT}$. The module can also be configured so that it does not go into Hibernate mode if the battery voltage drops below this threshold. Battery voltage is not measured while in Hibernate mode.

The Hibernation module can be configured to detect a low battery condition by setting the LOWBATEN bit of the **HIBCTL** register. In this configuration, the LOWBAT bit of the **Hibernation Raw Interrupt Status (HIBRIS)** register is set when the battery level is low. If the VABORT bit in the **HIBCTL** register is also set, then the module is prevented from entering Hibernation 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 269).

Note that the Hibernation module draws power from whichever source ($V_{BAT}$ or $V_{DD}$) has the higher voltage. Therefore, it is important to design the circuit to ensure that $V_{DD}$ is higher that $V_{BAT}$ under nominal conditions or else the Hibernation module draws power from the battery even when $V_{DD}$ is available.

### 6.3.4 Real-Time Clock

The Hibernation module includes a 32-bit counter that increments once per second with the proper configuration (see “Hibernation Clock Source” on page 266). The 32.768-kHz clock signal, either directly from the 32.768-kHz oscillator or from the 4.194304-MHz crystal divided by 128, is fed into a predivider register that counts down the 32.768-kHz clock ticks to achieve a once per second clock rate for the RTC. The 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 to divide the input clock. 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.

The Hibernation module includes two 32-bit match registers that are compared to the value of the RTC counter. The match registers can be used to wake the processor from Hibernation mode or to generate an interrupt to the processor if it is not in hibernation.

The RTC must be enabled with the RTCEN bit of the **HIBCTL** register. The value of the RTC can be set at any time by writing to the **HIBRTCLD** register. The predivider trim can be adjusted by reading and writing the **HIBRTCT** register. The predivider uses this register once every 64 seconds to adjust the clock rate. The two match registers can be set by writing to the **HIBRTCM0** and **HIBRTCM1** registers. The RTC can be configured to generate interrupts by using the interrupt registers (see “Interrupts and Status” on page 269).

### 6.3.5 Non-Volatile Memory

The Hibernation module contains 64 32-bit words of memory that are powered from the battery or 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. The non-volatile memory can be accessed through the **HIBDATA** registers.

### 6.3.6 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 $V_{DC}$ or powered down with the same regulator controlled by $\overline{\text{HIB}}$. See "Hibernation Module" on page 793 for more details.

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 (which could be a battery or an auxiliary power source) 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.

### 6.3.7 Power Control Using VDD3ON Mode

The Hibernation module may also be configured to cut power to all internal modules. While in this state, all pins are configured as inputs. In the VDD3ON mode, the regulator should maintain 3.3 V power to the microcontroller during Hibernate. This power control mode is enabled by setting the VDD3ON bit in **HIBCTL**.

### 6.3.8 Initiating Hibernate

Prior to initiating hibernation, a wake-up condition must be configured, either from the external $\overline{\text{WAKE}}$ pin, or by using an RTC match. Hibernation mode is initiated when the HIBREQ bit of the **HIBCTL** register is set. If a Flash memory write operation is in progress, an interlock feature holds off the transition into Hibernation mode until the write has completed.

The Hibernation module is configured to wake from the external $\overline{\text{WAKE}}$ pin by setting the PINWEN bit of the **HIBCTL** register. It is configured to wake from RTC match by setting the RTCWEN bit. Either one or both of these bits must be set prior to going into hibernation. Note that the $\overline{\text{WAKE}}$ pin uses the Hibernation module's internal power supply as the logic 1 reference.

Upon either external wake-up or RTC match, the Hibernation module delays coming out of hibernation until $V_{DD}$ is above the minimum specified voltage, see Table 19-2 on page 785.

When the Hibernation module wakes, the microcontroller performs a normal power-on reset. 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 269) and by looking for state data in the non-volatile memory (see "Non-Volatile Memory" on page 268).

### 6.3.9 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

All of the interrupts 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 the wake condition was caused by the WAKE signal or the RTC match.

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.

## 6.4 Initialization and Configuration

The Hibernation module has several different configurations. The following sections show the recommended programming sequence for various scenarios. The examples below assume that a 32.768-kHz oscillator is used, and thus always set the CLKSEL bit of the **HIBCTL** register. If a 4.194304-MHz crystal is used instead, then the CLKSEL bit remains cleared. Because the Hibernation module runs at 32.768 kHz 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 certain registers (see “Register Access Timing” on page 265). The registers that require a delay are listed in a note in “Register Map” on page 272 as well as in each register description.

### 6.4.1 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 237.

If a 4.194304-MHz crystal is used as the Hibernation module clock source, perform the following steps:

1. Write 0x40 to the **HIBCTL** register at offset 0x10 to enable the crystal and select the divide-by-128 input path.
2. Wait until the WC interrupt in the **HIBMIS** register has been triggered before performing any other operations with the Hibernation module.

If a 32.678-kHz single-ended oscillator is used as the Hibernation module clock source, then perform the following steps:

1. Write 0x44 to the **HIBCTL** register at offset 0x10 to enable the oscillator input and bypass the on-chip oscillator.
2. No delay is necessary.

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.

Table 6-3 on page 270 illustrates how the clocks function with various bit setting both in normal operation and in hibernation.

**Table 6-3. Hibernation Module Clock Operation**

| CLK32EN | PINWEN | RTCWEN | CLKSEL | RTCEN | Result Normal Operation | Result Hibernation |
|---|---|---|---|---|---|---|
| 0 | X | X | X | X | Hibernation module disabled | Hibernation module disabled |

**Table 6-3. Hibernation Module Clock Operation *(continued)***

| CLK32EN | PINWEN | RTCWEN | CLKSEL | RTCEN | Result Normal Operation | Result Hibernation |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | RTC match capability enabled. Module clocked from 4.184304-MHz crystal. | No hibernation |
| 1 | 0 | 0 | 1 | 1 | RTC match capability enabled. Module clocked from 32.768-kHz oscillator. | No hibernation |
| 1 | 0 | 1 | X | 1 | Module clocked from selected source | RTC match for wake-up event |
| 1 | 1 | 0 | X | 0 | Module clocked from selected source | Clock is powered down during hibernation and powered up again on external wake-up event. |
| 1 | 1 | 0 | X | 1 | Module clocked from selected source | Clock is powered up during hibernation for RTC. Wake up on external event. |
| 1 | 1 | 1 | X | 1 | Module clocked from selected source | RTC match or external wake-up event, whichever occurs first. |

### 6.4.2 RTC Match Functionality (No Hibernation)

Use the following steps to implement the RTC match functionality of the Hibernation module:

1. Write the required RTC match value to one of the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Set the required RTC match interrupt mask in the RTCALT0 and RTCALT1 bits (bits 1:0) in the **HIBIM** register at offset 0x014.
4. Write 0x0000.0041 to the **HIBCTL** register at offset 0x010 to enable the RTC to begin counting.

### 6.4.3 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 the required RTC match value to the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.
4. Set the RTC Match Wake-Up and start the hibernation sequence by writing 0x0000.004F to the **HIBCTL** register at offset 0x010.

### 6.4.4 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 any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.
2. Enable the external wake and start the hibernation sequence by writing 0x0000.0056 to the **HIBCTL** register at offset 0x010.

Note that in this mode, if the RTC is disabled, then the Hibernation clock source is powered down during Hibernation mode and is powered up again on the external wake event to save power during hibernation. If the RTC is enabled before hibernation, it continues to operate during hibernation.

### 6.4.5 RTC or External Wake-Up from Hibernation

1. Write the required RTC match value to the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.
4. Set the RTC Match/External Wake-Up and start the hibernation sequence by writing 0x0000.005F to the **HIBCTL** register at offset 0x010.

### 6.4.6 Register Reset

The Hibernation module handles resets according to the following conditions:

- Cold Reset

  When the Hibernation module has no voltage applied to either $V_{DD}$ or $V_{BAT}$, and a voltage is subsequently applied to either $V_{DD}$ or $V_{BAT}$, all Hibernation module registers are reset to the value in Table 6-4 on page 273.

- Reset During Hibernation Module Disable

  When the module has either not been enabled or has been disabled by software, the reset is passed through to the Hibernation module circuitry, and the internal state of the module is reset. Non-volatile memory contents are not reset to zero and contents after reset are indeterminate.

- Reset While Hibernation Module is in Hibernation Mode

  While in Hibernation mode, or while transitioning from Hibernation mode to run mode, the reset generated by the POR circuitry of the microcontroller is suppressed, and the state of the Hibernation module's registers is unaffected.

- Reset While Hibernation Module is in Normal Mode

  While in normal mode (not hibernating), any reset is suppressed if either the RTCEN or the PINWEN bit is set in the **HIBCTL** register, and the content/state of the control and data registers is unaffected.

  Software must initialize any control or data registers in this condition. Therefore, software is the only mechanism to set or clear the CLK32EN bit and real-time clock operation, or to clear contents of the data memory. The only state that must be cleared by a reset operation while not in Hibernation mode is any state that prevents software from managing the interface.

  **Note:** If $V_{DD}$ drops below operational range while in normal mode (not hibernating), all Hibernation module registers are reset to the value in Table 6-4 on page 273, regardless of whether the proper voltage is applied to $V_{BAT}$.

## 6.5 Register Map

Table 6-4 on page 273 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 237). There must be a delay of 3

system clocks after the Hibernation module clock is enabled before any Hibernation module registers are accessed.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** 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 265.

**Important:** Reset values apply only to a cold reset. Once configured, the Hibernate module ignores any system reset, other than power on reset, as long as $V_{BAT}$ is present.

**Table 6-4. Hibernation Module Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | HIBRTCC | RO | 0x0000.0000 | Hibernation RTC Counter | 274 |
| 0x004 | HIBRTCM0 | R/W | 0xFFFF.FFFF | Hibernation RTC Match 0 | 275 |
| 0x008 | HIBRTCM1 | R/W | 0xFFFF.FFFF | Hibernation RTC Match 1 | 276 |
| 0x00C | HIBRTCLD | R/W | 0xFFFF.FFFF | Hibernation RTC Load | 277 |
| 0x010 | HIBCTL | R/W | 0x8000.0000 | Hibernation Control | 278 |
| 0x014 | HIBIM | R/W | 0x0000.0000 | Hibernation Interrupt Mask | 281 |
| 0x018 | HIBRIS | RO | 0x0000.0000 | Hibernation Raw Interrupt Status | 283 |
| 0x01C | HIBMIS | RO | 0x0000.0000 | Hibernation Masked Interrupt Status | 285 |
| 0x020 | HIBIC | R/W1C | 0x0000.0000 | Hibernation Interrupt Clear | 287 |
| 0x024 | HIBRTCT | R/W | 0x0000.7FFF | Hibernation RTC Trim | 288 |
| 0x030-0x12C | HIBDATA | R/W | - | Hibernation Data | 289 |

## 6.6 Register Descriptions

The remainder of this section lists and describes the Hibernation module registers, in numerical order by address offset.

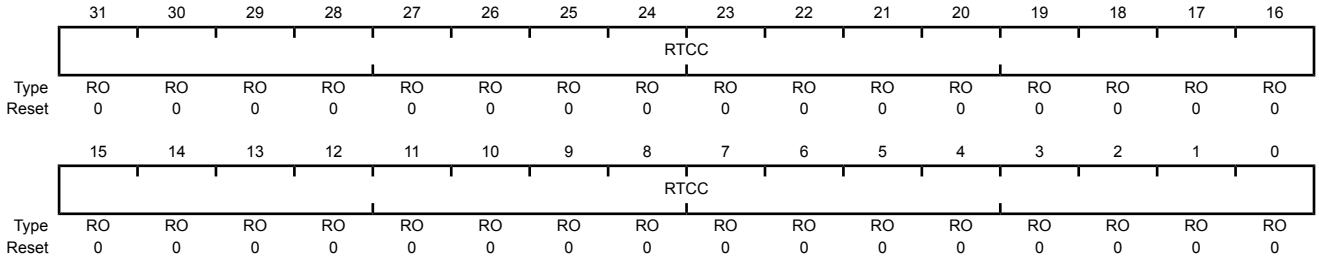
## Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000

This register is the current 32-bit value of the RTC counter.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** 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 265.

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. |

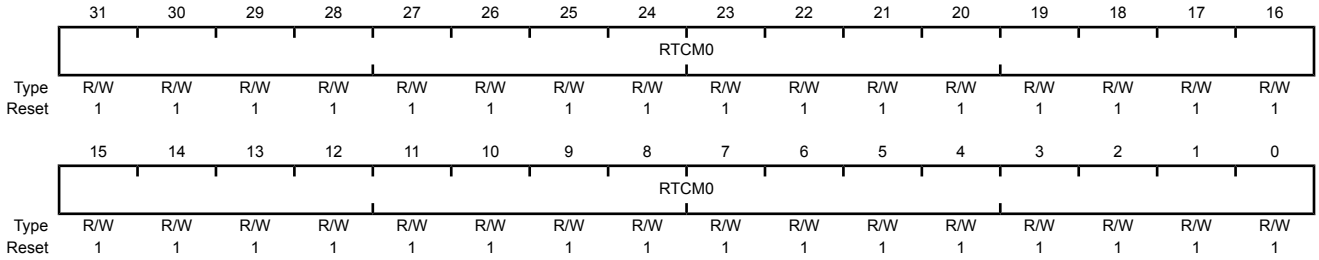
## Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004

This register is the 32-bit match 0 register for the RTC counter.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** 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 265.

Hibernation RTC Match 0 (HIBRTCM0)

Base 0x400F.C000
Offset 0x004
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTCM0 | R/W | 0xFFFF.FFFF | RTC Match 0<br>A write loads the value into the RTC match register.<br>A read returns the current match value. |

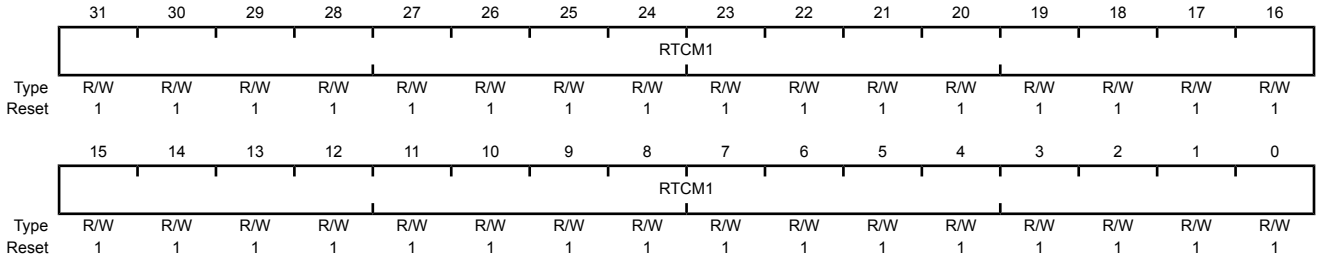
## Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008

This register is the 32-bit match 1 register for the RTC counter.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** 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 265.

Hibernation RTC Match 1 (HIBRTCM1)

Base 0x400F.C000
Offset 0x008
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTCM1 | R/W | 0xFFFF.FFFF | RTC Match 1<br>A write loads the value into the RTC match register.<br>A read returns the current match value. |

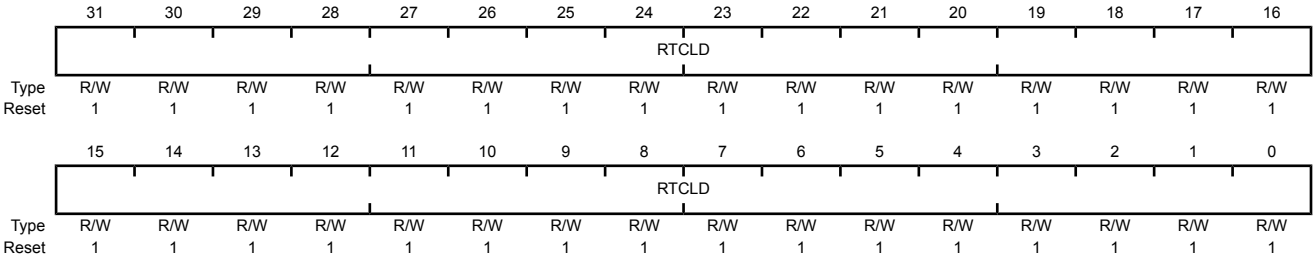
## Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C

This register is used to load a 32-bit value loaded into the RTC counter. The load occurs immediately upon this register being written.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** 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 265.

Hibernation RTC Load (HIBRTCLD)

Base 0x400F.C000
Offset 0x00C
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTCLD | R/W | 0xFFFF.FFFF | RTC Load<br>A write loads the current value into the RTC counter (RTCC).<br>A read returns the 32-bit load value. |

## Register 5: 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.

Hibernation Control (HIBCTL)

Base 0x400F.C000
Offset 0x010
Type R/W, reset 0x8000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | WRC | reserved | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| 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 | | | | | | | VDD3ON | VABORT | CLK32EN | LOWBATEN | PINWEN | RTCWEN | CLKSEL | HIBREQ | RTCEN |
| Type | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 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.

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.

| | | | | |
|---|---|---|---|---|
| 30:9 | 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. |
| 8 | VDD3ON | R/W | 0 | VDD Powered |

| Value | Description |
|---|---|
| 1 | The internal switches control the power to the on-chip modules (VDD3ON mode). |
| 0 | The internal switches are not used. The $\overline{\text{HIB}}$ signal should be used to control an external switch or regulator. |

Note that regardless of the status of the VDD3ON bit, the $\overline{\text{HIB}}$ signal is asserted during Hibernate mode. Thus, when VDD3ON is set, the $\overline{\text{HIB}}$ signal should not be connected to the 3.3V regulator, and the 3.3V power source should remain connected.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | VABORT | R/W | 0 | Power Cut Abort Enable<br><br>Value: 1 — When this bit is set, the battery voltage level is checked before entering hibernation. If $V_{BAT}$ is less than $V_{LOWBAT}$, the microcontroller does not go into hibernation.<br>Value: 0 — The microcontroller goes into hibernation regardless of the voltage level of the battery. |
| 6 | CLK32EN | R/W | 0 | Clocking Enable<br>This bit must be enabled to use the Hibernation module.<br><br>Value: 1 — The Hibernation module clock source is enabled.<br>Value: 0 — The Hibernation module clock source is disabled. |
| 5 | LOWBATEN | R/W | 0 | Low Battery Monitoring Enable<br><br>Value: 1 — Low battery voltage detection is enabled. When this bit is set, the battery voltage level is checked before entering hibernation. If $V_{BAT}$ is less than $V_{LOWBAT}$, the LOWBAT bit in the **HIBRIS** register is set.<br>Value: 0 — Low battery monitoring is disabled. |
| 4 | PINWEN | R/W | 0 | External WAKE Pin Enable<br><br>Value: 1 — An assertion of the WAKE pin takes the microcontroller out of hibernation.<br>Value: 0 — The status of the WAKE pin has no effect on hibernation. |
| 3 | RTCWEN | R/W | 0 | RTC Wake-up Enable<br><br>Value: 1 — An RTC match event (the value the **HIBRTCC** register matches the value of the **HIBRTCM0** or **HIBRTCM1** register) takes the microcontroller out of hibernation.<br>Value: 0 — An RTC match event has no effect on hibernation. |
| 2 | CLKSEL | R/W | 0 | Hibernation Module Clock Select<br><br>Value: 1 — Use raw output. Use this value for a 32.768-kHz oscillator.<br>Value: 0 — Use Divide-by-128 output. Use this value for a 4.194304-MHz crystal. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | HIBREQ | R/W | 0 | Hibernation Request<br><br>Value: 1 — Set this bit to initiate hibernation.<br>Value: 0 — No hibernation request.<br><br>After a wake-up event, this bit is automatically cleared by hardware. |
| 0 | RTCEN | R/W | 0 | RTC Timer Enable<br><br>Value: 1 — The Hibernation module RTC is enabled. The RTC remains active during hibernation.<br>Value: 0 — The Hibernation module RTC is disabled. When this bit is clear and PINWEN is set, enabling an external wake event, the RTC stops during hibernation to save power. |

## Register 6: 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.

Hibernation Interrupt Mask (HIBIM)

Base 0x400F.C000
Offset 0x014
Type R/W, reset 0x0000.0000

| Bit | 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 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | EXTW | LOWBAT | RTCALT1 | RTCALT0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| 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 | EXTW | R/W | 0 | External Wake-Up Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the EXTW bit in the **HIBRIS** register is set.<br>0 The EXTW interrupt is suppressed and not sent to the interrupt controller. |
| 2 | LOWBAT | R/W | 0 | Low Battery Voltage Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the LOWBAT bit in the **HIBRIS** register is set.<br>0 The LOWBAT interrupt is suppressed and not sent to the interrupt controller. |
| 1 | RTCALT1 | R/W | 0 | RTC Alert 1 Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the RTCALT1 bit in the **HIBRIS** register is set.<br>0 The RTCALT1 interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RTCALT0 | R/W | 0 | RTC Alert 0 Interrupt Mask |

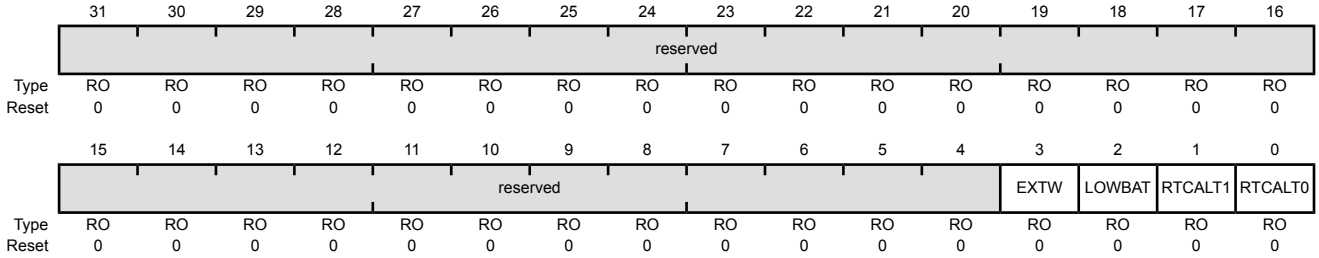
| Value | Description |
|---|---|
| 1 | An interrupt is sent to the interrupt controller when the RTCALT0 bit in the **HIBRIS** register is set. |
| 0 | The RTCALT0 interrupt is suppressed and not sent to the interrupt controller. |

## Register 7: 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 |
|---|---|---|---|---|
| 31:4 | 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 | EXTW | RO | 0 | External Wake-Up Raw Interrupt Status<br><br>Value Description<br>1 The $\overline{\text{WAKE}}$ pin has been asserted.<br>0 The $\overline{\text{WAKE}}$ pin has not been asserted.<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 Raw Interrupt Status<br><br>Value Description<br>1 The battery voltage dropped below $V_{LOWBAT}$.<br>0 The battery voltage has not dropped below $V_{LOWBAT}$.<br><br>This bit is cleared by writing a 1 to the LOWBAT bit in the **HIBIC** register. |
| 1 | RTCALT1 | RO | 0 | RTC Alert 1 Raw Interrupt Status<br><br>Value Description<br>1 The value of the **HIBRTCC** register matches the value in the **HIBRTCM1** register.<br>0 No match<br><br>This bit is cleared by writing a 1 to the RTCALT1 bit in the **HIBIC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RTCALT0 | RO | 0 | RTC Alert 0 Raw Interrupt Status |

| Value | Description |
|---|---|
| 1 | The value of the **HIBRTCC** register matches the value in the **HIBRTCM0** register. |
| 0 | No match |

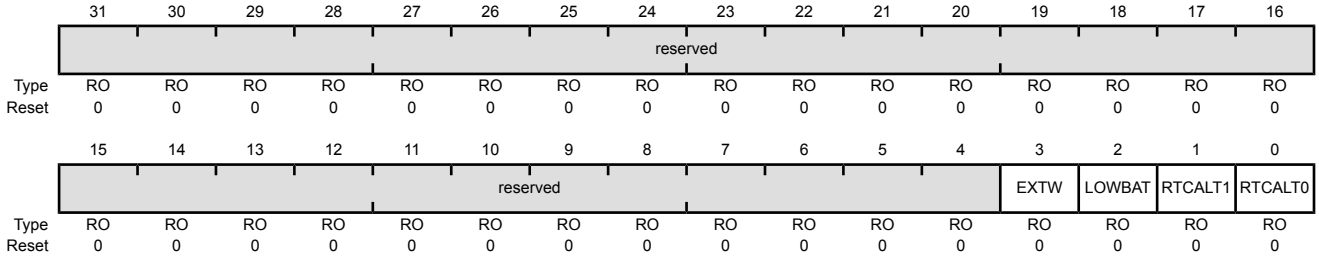
This bit is cleared by writing a 1 to the RTCALT0 bit in the **HIBIC** register.

## Register 8: 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 |
|---|---|---|---|---|
| 31:4 | 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 | EXTW | RO | 0 | External Wake-Up Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to a $\overline{\text{WAKE}}$ pin assertion.<br>0 An external wake-up interrupt has not occurred or is masked.<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>1 An unmasked interrupt was signaled due to a low battery voltage condition.<br>0 A low battery voltage interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the LOWBAT bit in the **HIBIC** register. |
| 1 | RTCALT1 | RO | 0 | RTC Alert 1 Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to an RTC match.<br>0 An RTC match interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the RTCALT1 bit in the **HIBIC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RTCALT0 | RO | 0 | RTC Alert 0 Masked Interrupt Status |

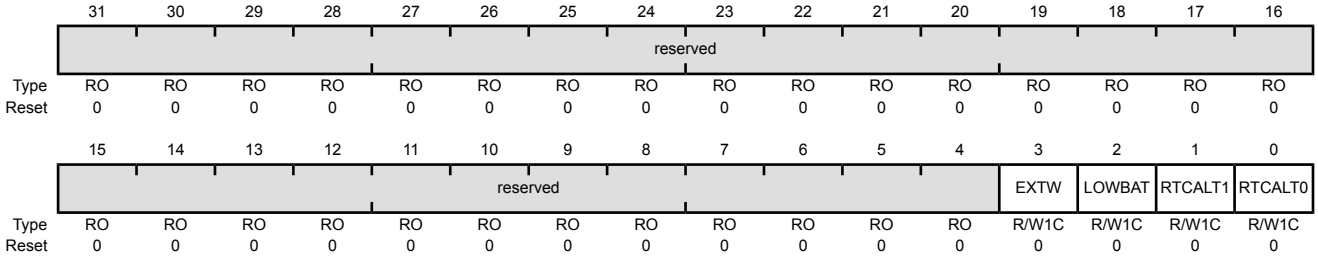
| Value | Description |
|---|---|
| 1 | An unmasked interrupt was signaled due to an RTC match. |
| 0 | An RTC match interrupt has not occurred or is masked. |

This bit is cleared by writing a 1 to the RTCALT0 bit in the **HIBIC** register.

## Register 9: 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.

Hibernation Interrupt Clear (HIBIC)
Base 0x400F.C000
Offset 0x020
Type R/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 | EXTW | R/W1C | 0 | External Wake-Up Masked Interrupt Clear<br>Writing a 1 to this bit clears the EXTW bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return an indeterminate value. |
| 2 | LOWBAT | R/W1C | 0 | Low Battery Voltage Masked Interrupt Clear<br>Writing a 1 to this bit clears the LOWBAT bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return an indeterminate value. |
| 1 | RTCALT1 | R/W1C | 0 | RTC Alert1 Masked Interrupt Clear<br>Writing a 1 to this bit clears the RTCALT1 bit in the **HIBRIS** and **HIBMIS** registers.<br>Reads return an indeterminate value. |
| 0 | RTCALT0 | R/W1C | 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 an indeterminate value. |

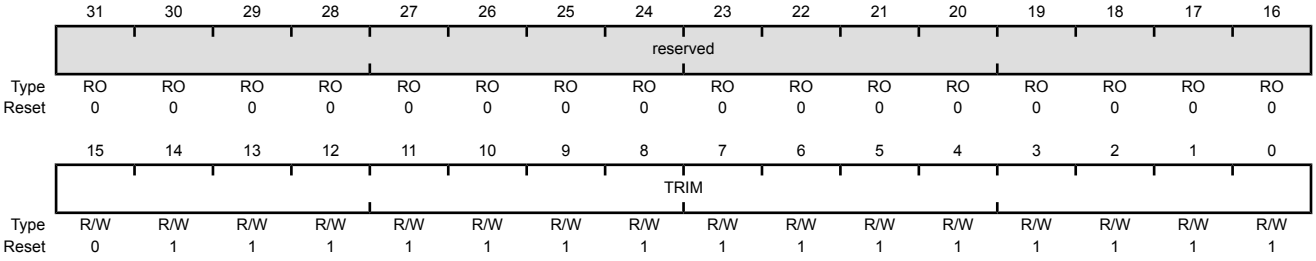
## Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024

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 63 seconds.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** 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 265.

Hibernation RTC Trim (HIBRTCT)

Base 0x400F.C000
Offset 0x024
Type R/W, 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 | R/W | 0x7FFF | RTC Trim Value<br>This value is loaded into the RTC predivider every 64 seconds. 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. |

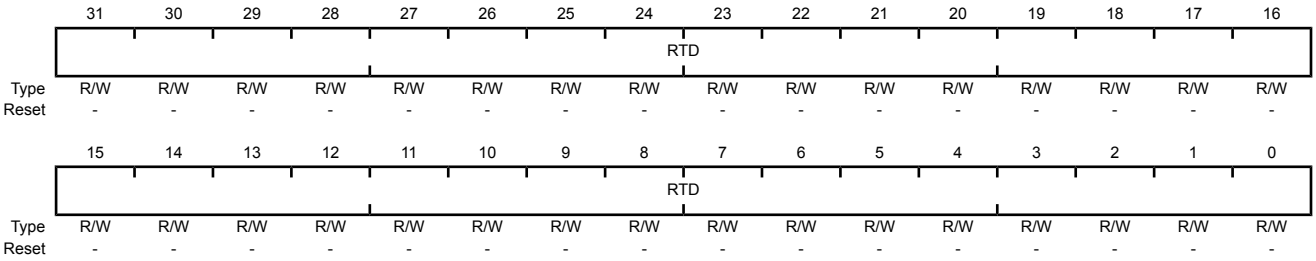
## Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C

This address space is implemented as a 64x32-bit memory (256 bytes). It can be loaded by the system processor in order to store any non-volatile state data and does not lose power during a power cut operation.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** 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 265.

Hibernation Data (HIBDATA)

Base 0x400F.C000
Offset 0x030-0x12C
Type R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | RTD | R/W | - | Hibernation Module NV Data |

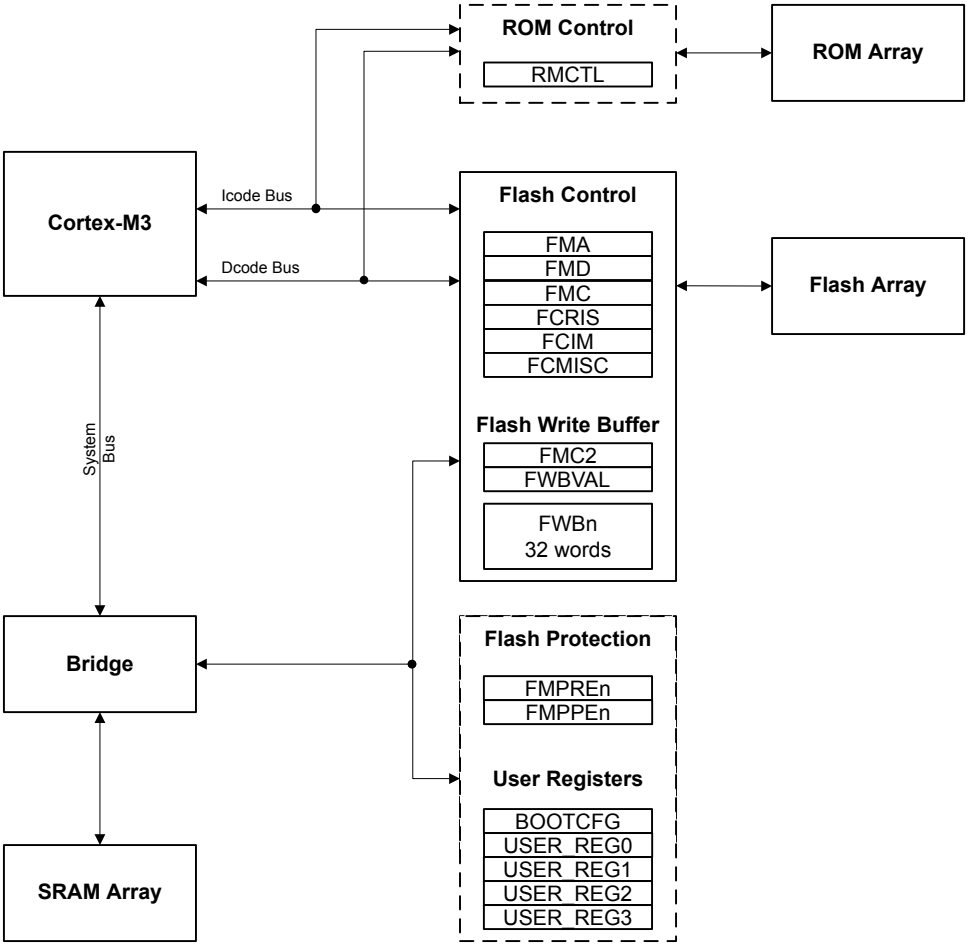
# 7 Internal Memory

The LM3S1G58 microcontroller comes with 64 KB of bit-banded SRAM, internal ROM,and 384 KB of Flash memory. The Flash memory controller provides a user-friendly interface, making Flash memory programming a simple task. Flash memory protection can be applied to the Flash memory on a 2-KB block basis.

## 7.1 Block Diagram

Figure 7-1 on page 290 illustrates the internal memory blocks and control logic. The dashed boxes in the figure indicate registers residing in the System Control module.

**Figure 7-1. Internal Memory Block Diagram**




## 7.2 Functional Description

This section describes the functionality of the SRAM, ROM, and Flash memories.

**Note:** The µDMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the µDMA controller.

### 7.2.1 SRAM

The internal SRAM of the Stellaris® 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 71.

**Note:** The SRAM is implemented using two 32-bit wide SRAM banks (separate SRAM arrays). The banks are partitioned such that one bank contains all even words (the even bank) and the other contains all odd words (the odd bank). A write access that is followed immediately by a read access to the same bank incurs a stall of a single clock cycle. However, a write to one bank followed by a read of the other bank can occur in successive clock cycles without incurring any delay.

### 7.2.2 ROM

The internal ROM of the Stellaris device is located at address 0x0100.0000 of the device memory map. Detailed information on the ROM contents can be found in the *Stellaris® ROM User's Guide*.

The ROM contains the following components:

- Stellaris Boot Loader and vector table
- Stellaris 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 memory is empty) 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). Advance Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government and Cyclic Redundancy Check (CRC) is a technique to validate a span of data has the same contents as when previously checked.

#### 7.2.2.1 Boot Loader Overview

The Stellaris Boot Loader is used to download code to the Flash memory of a device without the use of a debug interface. When the core is 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 in Ports A-H as configured in the **Boot Configuration (BOOTCFG)** register.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The BA bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the EN bit is clear, 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 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 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 as it does not enable the PLL. The following serial interfaces can be used:

- UART0
- SSI0
- $I^2C0$

For simplicity, both the data format and communication protocol are identical for all serial interfaces.

See the *Stellaris® Boot Loader User's Guide* for information on the boot loader software.

#### 7.2.2.2 Stellaris Peripheral Driver Library

The Stellaris 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 *Stellaris® ROM User's Guide*. See the "Using the ROM" chapter of the *Stellaris® Peripheral Driver Library User's Guide* 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 will 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-M3 vector table in the ROM.

DriverLib functions are described in detail in the *Stellaris® Peripheral Driver Library User's Guide*.

Additional APIs are available for graphics and USB functions, but are not preloaded into ROM. The Stellaris Graphics Library provides a set of graphics primitives and a widget set for creating graphical user interfaces on Stellaris microcontroller-based boards that have a graphical display (for more information, see the *Stellaris® Graphics Library User's Guide*).

#### 7.2.2.3 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 pre-arranged 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 *Stellaris® ROM User's Guide* for more information on AES.

#### 7.2.2.4 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 (e.g. XOR all bits) because it catches changes more readily. See the *Stellaris® ROM User's Guide* for more information on CRC.

### 7.2.3 Flash Memory

At system clock speeds of 50 MHz and below, the Flash memory is read in a single cycle. The Flash memory is organized as a set of 1-KB blocks that can be individually erased. An individual 32-bit word can be programmed to change bits from 1 to 0. In addition, a write buffer provides the ability to concurrently program 32 continuous words in Flash memory. Erasing a block causes the entire contents of the block to be reset to all 1s. The 1-KB blocks are paired into sets of 2-KB blocks that can be individually protected. The protection allows blocks to 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.

**Caution – The Stellaris Flash memory array has ECC which uses a test port into the Flash memory to continually scan the array for ECC errors and to correct any that are detected. This operation is transparent to the microcontroller. The BIST must scan the entire memory array occasionally to ensure integrity, taking about five minutes to do so. In systems where the microcontroller is frequently powered for less than five minutes, power should be removed from the microcontroller in a controlled manner to ensure proper operation. This controlled manner can either be through entering Hibernation mode or software can request permission to power down the part using the USDREQ bit in the Flash Control (FCTL) register and wait to receive an acknowledge from the USDACK bit prior to removing power. If the microcontroller is powered down using this controlled method, the BIST engine keeps track of where it was in the memory array and it always scans the complete array after any aggregate of five minutes powered-on, regardless of the number of intervening power cycles. If the microcontroller is powered down before five minutes of being powered up, BIST starts again from wherever it left off before the last controlled power-down or from 0 if there never was a controlled power down. An occasional short power down is not a concern, but the microcontroller should not always be powered down frequently in an uncontrolled manner. The microcontroller can be power-cycled as frequently as necessary if it is powered-down in a controlled manner.**

#### 7.2.3.1 Prefetch Buffer

The Flash memory controller has a prefetch buffer that is automatically used when the CPU frequency is greater than 50 MHz. In this mode, the Flash memory operates at half of the system clock. The prefetch buffer fetches two 32-bit words per clock allowing instructions to be fetched with no wait states while code is executing linearly. The fetch buffer includes a branch speculation mechanism

that recognizes a branch and avoids extra wait states by not reading the next word pair. Also, short loop branches often stay in the buffer. As a result, some branches can be executed with no wait states. Other branches incur a single wait state.

### 7.2.3.2 Flash Memory Protection

The user is provided two forms of Flash memory protection per 2-KB Flash memory block in six pairs 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)**: If a bit is set, the corresponding block may be programmed (written) or erased. If a bit is cleared, the corresponding block may not be changed.
- **Flash Memory Protection Read Enable (FMPREn)**: If a bit is set, the corresponding block may be executed or read by software or debuggers. If a bit is cleared, the corresponding block may only be executed, and contents of the memory block are prohibited from being read as data.

The policies may be combined as shown in Table 7-1 on page 294.

**Table 7-1. 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 set) is prohibited and generates a bus fault. A Flash memory access that attempts to program or erase a program-protected block (**FMPPEn** bit is set) 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 power-on reset sequence. The changes are committed using the **Flash Memory Control (FMC)** register. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 297.

### 7.2.3.3 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.
- Access Interrupt - signals when a program or erase action has been attempted on a 2-kB block of memory that is protected by its corresponding **FMPPEn** bit.

The interrupt events that can trigger a controller-level interrupt are defined in the **Flash Controller Masked Interrupt Status (FCMIS)** register (see page 306) 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 305).

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 307).

#### 7.2.3.4 Flash Memory Programming

The Stellaris 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 162.

During a Flash memory operation (write, page erase, or mass erase) access to the Flash memory is inhibited. As a result, instruction and literal fetches 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.

---

**Caution – The Flash memory is divided into sectors of electrically separated address ranges of 4 KB each, aligned on 4 KB boundaries. Erase/program operations on a 1-KB page have an electrical effect on the other three 1-KB pages within the sector. A specific 1-KB page must be erased after 6 total erase/program cycles occur to the other pages within its 4-KB sector. The following sequence of operations on a 4-KB sector of Flash memory (Page 0..3) provides an example:**

- **Page 3 is erase and programmed with values.**
- **Page 0, Page 1, and Page 2 are erased and then programmed with values. At this point Page 3 has been affected by 3 erase/program cycles.**
- **Page 0, Page 1, and Page 2 are again erased and then programmed with values. At this point Page 3 has been affected by 6 erase/program cycles.**
- **If the contents of Page 3 must continue to be valid, Page 3 must be erased and reprogrammed before any other page in this sector has another erase or program operation.**

---

***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.
4. Poll the **FMC** register until the WRITE bit is cleared.

---

**Important:** To ensure proper operation, two writes to the same word must be separated by an ERASE. The following two sequences are allowed:

- ERASE -> PROGRAM value -> PROGRAM 0x0000.0000
- ERASE -> PROGRAM value -> ERASE

The following sequence is NOT allowed:

- ERASE -> PROGRAM value -> PROGRAM value

---

***To perform an erase of a 1-KB page***

1. Write the page address to the **FMA** register.
2. Write the Flash memory write key and the ERASE bit (a value of 0xA442.0002) 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 (a value of 0xA442.0004) 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.

#### 7.2.3.5 32-Word Flash Memory Write Buffer

A 32-word write buffer provides the capability to perform faster write accesses to the Flash memory by concurrently programing 32 words with a single buffered Flash memory write operation. The buffered Flash memory write operation takes the same amount of time as the single word write operation controlled by bit 0 in the **FMC** register. 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 (a value of 0xA442.0001) to the **FMC2** register.
4. Poll the **FMC2** register until the WRBUF bit is cleared or wait for the PMIS interrupt to be signaled.

### 7.2.3.6 Nonvolatile Register Programming

This section discusses how to update registers that are resident within the Flash memory itself. 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 write operation. The register contents are unaffected by any reset condition except power-on reset, which returns the register contents to 0xFFFF.FFFF. By committing the register values using the COMT bit in the **FMC** register, the register contents become nonvolatile 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 162.

With the exception of the **Boot Configuration (BOOTCFG)** register, the settings in these registers can be tested before committing them to Flash memory. For the **BOOTCFG** register, the data to be written is loaded into the **FMD** register before it is committed. The **FMD** register is read only and does not allow the **BOOTCFG** operation to be tried before committing it to nonvolatile memory.

**Important:** The Flash memory resident registers can only have bits changed from 1 to 0 by user programming and can only be committed once. After being committed, these registers can only be restored to their factory default values only by performing the sequence described in “Recovering a "Locked" Microcontroller” on page 162. The mass erase of the main Flash memory array caused by the sequence is performed prior to restoring these registers.

In addition, the **USER_REG0**, **USER_REG1**, **USER_REG2**, **USER_REG3**, and **BOOTCFG** registers each use bit 31 (NW) to indicate that they have not been committed and bits in the register may be changed from 1 to 0. Table 7-2 on page 297 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 value of 0xA442.0008. After writing the COMT bit, the user may poll the **FMC** register to wait for the commit operation to complete.

**Table 7-2. 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 |
| FMPPE0 | 0x0000.0001 | FMPPE0 |
| FMPPE1 | 0x0000.0003 | FMPPE1 |
| FMPPE2 | 0x0000.0005 | FMPPE2 |
| FMPPE3 | 0x0000.0007 | FMPPE3 |
| FMPRE4 | 0x0000.0009 | FMPRE4 |
| FMPRE5 | 0x0000.000B | FMPRE5 |
| 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 |

## 7.3 Register Map

Table 7-3 on page 298 lists the ROM Controller register and the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, **FCMISC**, **FMC2**, **FWBVAL**, and **FWBn** register offsets are relative to the Flash memory control base address of 0x400F.D000. The ROM and Flash memory protection register offsets are relative to the System Control base address of 0x400F.E000.

**Table 7-3. Flash Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **Flash Memory Registers (Flash Control Offset)** | | | | | |
| 0x000 | FMA | R/W | 0x0000.0000 | Flash Memory Address | 300 |
| 0x004 | FMD | R/W | 0x0000.0000 | Flash Memory Data | 301 |
| 0x008 | FMC | R/W | 0x0000.0000 | Flash Memory Control | 302 |
| 0x00C | FCRIS | RO | 0x0000.0000 | Flash Controller Raw Interrupt Status | 305 |
| 0x010 | FCIM | R/W | 0x0000.0000 | Flash Controller Interrupt Mask | 306 |
| 0x014 | FCMISC | R/W1C | 0x0000.0000 | Flash Controller Masked Interrupt Status and Clear | 307 |
| 0x020 | FMC2 | R/W | 0x0000.0000 | Flash Memory Control 2 | 308 |
| 0x030 | FWBVAL | R/W | 0x0000.0000 | Flash Write Buffer Valid | 309 |
| 0x0F8 | FCTL | R/W | 0x0000.0000 | Flash Control | 310 |
| 0x100 - 0x17C | FWBn | R/W | 0x0000.0000 | Flash Write Buffer n | 311 |
| **Memory Registers (System Control Offset)** | | | | | |
| 0x0F0 | RMCTL | R/W1C | - | ROM Control | 312 |
| 0x130 | FMPRE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 0 | 313 |
| 0x200 | FMPRE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 0 | 313 |
| 0x134 | FMPPE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 0 | 314 |
| 0x400 | FMPPE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 0 | 314 |
| 0x1D0 | BOOTCFG | R/W | 0xFFFF.FFFE | Boot Configuration | 315 |
| 0x1E0 | USER_REG0 | R/W | 0xFFFF.FFFF | User Register 0 | 317 |
| 0x1E4 | USER_REG1 | R/W | 0xFFFF.FFFF | User Register 1 | 318 |
| 0x1E8 | USER_REG2 | R/W | 0xFFFF.FFFF | User Register 2 | 319 |
| 0x1EC | USER_REG3 | R/W | 0xFFFF.FFFF | User Register 3 | 320 |
| 0x204 | FMPRE1 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 1 | 321 |
| 0x208 | FMPRE2 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 2 | 322 |
| 0x20C | FMPRE3 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 3 | 323 |
| 0x210 | FMPRE4 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 4 | 324 |

**Table 7-3. Flash Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x214 | FMPRE5 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 5 | 325 |
| 0x218 | FMPRE6 | R/W | 0x0000.0000 | Flash Memory Protection Read Enable 6 | 326 |
| 0x21C | FMPRE7 | R/W | 0x0000.0000 | Flash Memory Protection Read Enable 7 | 327 |
| 0x404 | FMPPE1 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 1 | 328 |
| 0x408 | FMPPE2 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 2 | 329 |
| 0x40C | FMPPE3 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 3 | 330 |
| 0x410 | FMPPE4 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 4 | 331 |
| 0x414 | FMPPE5 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 5 | 332 |
| 0x418 | FMPPE6 | R/W | 0x0000.0000 | Flash Memory Protection Program Enable 6 | 333 |
| 0x41C | FMPPE7 | R/W | 0x0000.0000 | Flash Memory Protection Program Enable 7 | 334 |

## 7.4 Flash Memory Register Descriptions (Flash Control Offset)

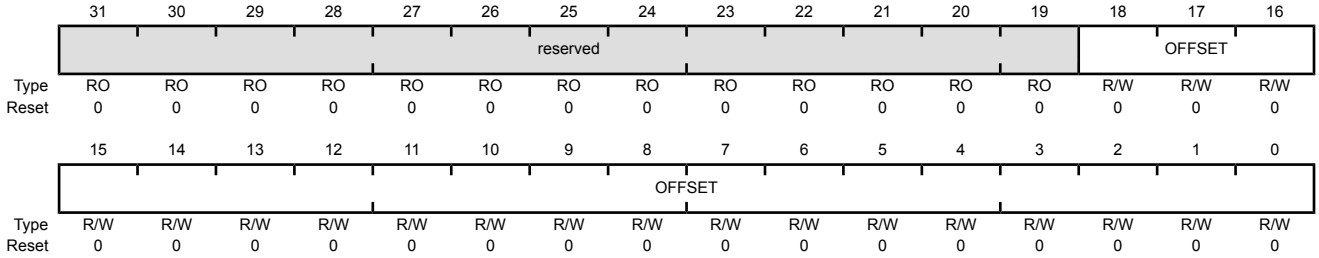
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

## 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, this register contains a 1 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 R/W, reset 0x0000.0000



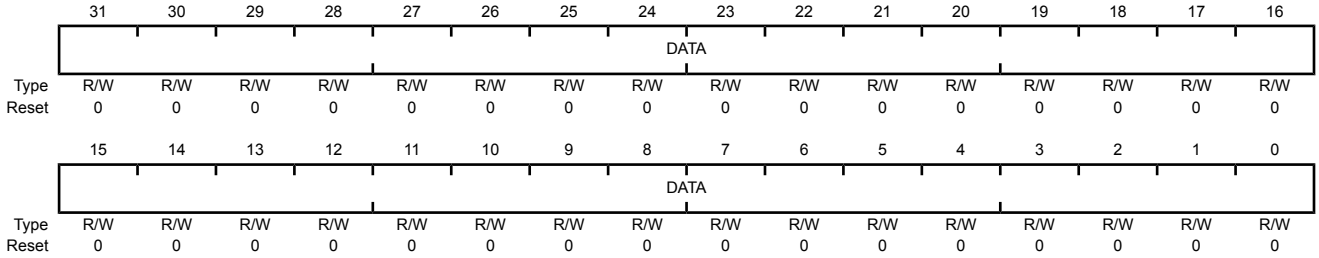

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:19 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 18:0 | OFFSET | R/W | 0x0 | Address Offset<br>Address offset in Flash memory where operation is performed, except for nonvolatile registers (see “Nonvolatile Register Programming” on page 297 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 or read during the read 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | R/W | 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 300). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 301) is written to the specified address.

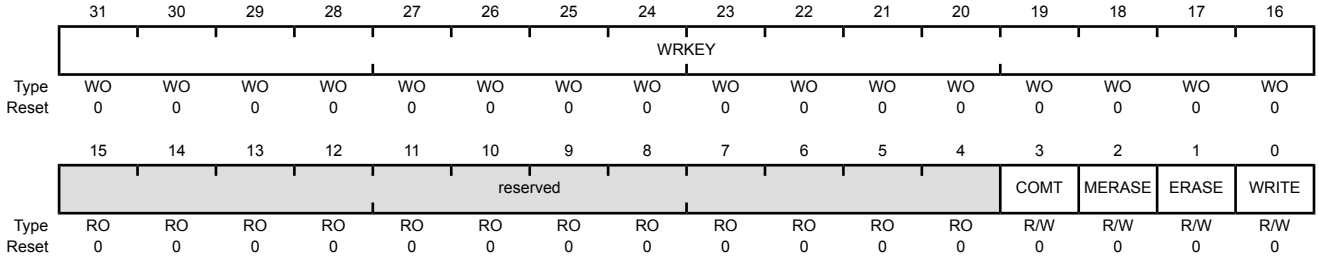
This register must be the final register written and initiates the memory operation. The four control bits in the lower byte of this register are used to initiate memory operations.

Care must be taken not to set multiple control bits as the results of such an operation are unpredictable.

**Caution – If any of bits [15:4] are written to 1, the device may become inoperable. These bits should always be written to 0. In all registers, the value of a reserved bit should be preserved across a read-modify-write operation.**

Flash Memory Control (FMC)

Base 0x400F.D000
Offset 0x008
Type R/W, 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 must be written into this field for a Flash memory write to occur. 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 | R/W | 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>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>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><br>See "Nonvolatile Register Programming" on page 297 for more information on programming Flash-memory-resident registers. |
| 2 | MERASE | R/W | 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>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>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><br>For information on erase time, see "Flash Memory " on page 795. |
| 1 | ERASE | R/W | 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>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>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><br>For information on erase time, see "Flash Memory " on page 795. |

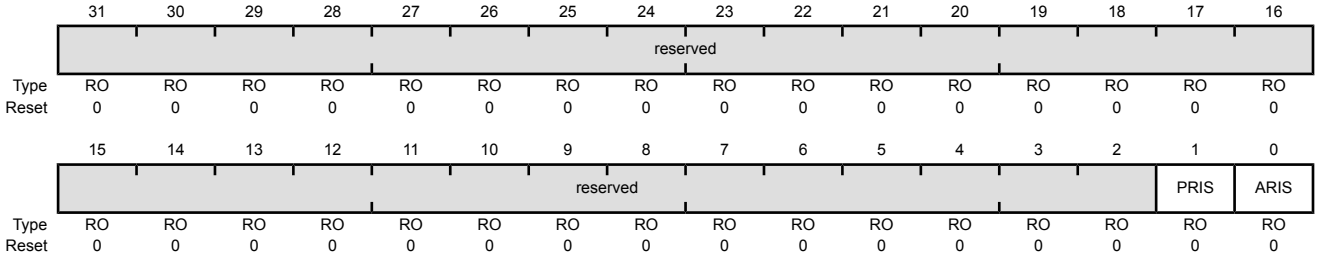
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | WRITE | R/W | 0 | Write a Word into Flash Memory<br>This bit is used to write a word into Flash memory and to monitor the progress of that process.<br><br>Value Description<br>1 Set this bit to write the data stored in the **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>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><br>For information on programming time, see “Flash Memory ” on page 795. |

## 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



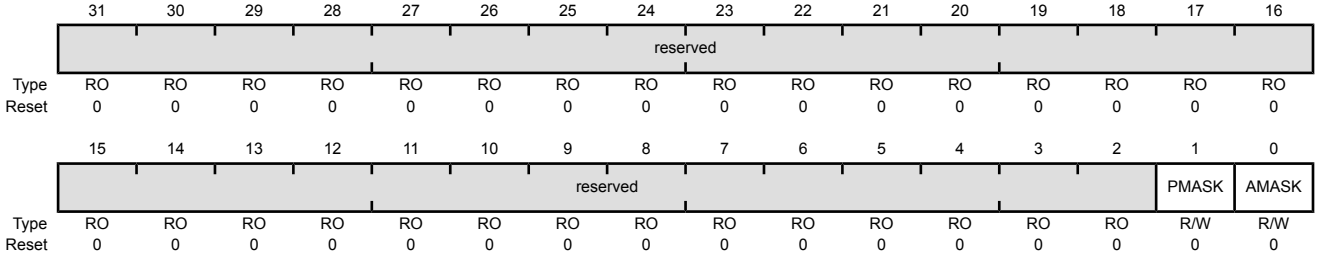

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | 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 302 and page 308).<br><br>Value Description<br>1 The programming or erase cycle has completed.<br>0 The programming or erase cycle has not 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. |
| 0 | ARIS | RO | 0 | Access Raw Interrupt Status<br><br>Value Description<br>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.<br>0 No access has tried to improperly program or erase the Flash memory.<br><br>This status is sent to the interrupt controller when the AMASK bit in the **FCIM** register is set.<br>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 R/W, reset 0x0000.0000



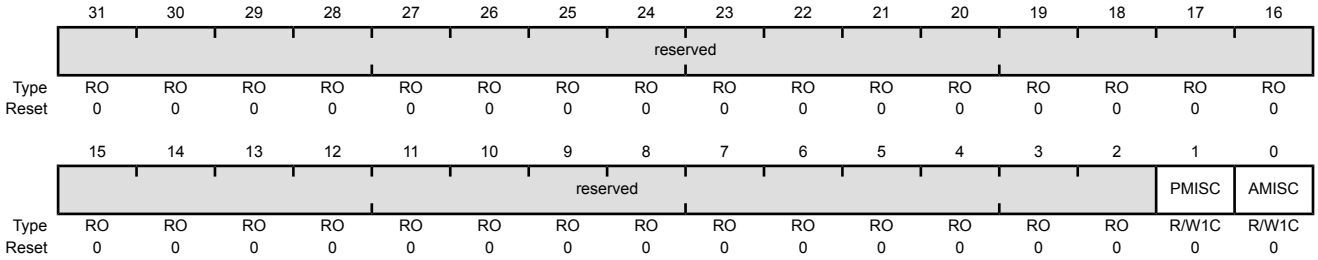

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | PMASK | R/W | 0 | Programming Interrupt Mask<br>This bit controls the reporting of the programming raw interrupt status to the interrupt controller.<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the PRIS bit is set.<br>0 The PRIS interrupt is suppressed and not sent to the interrupt controller. |
| 0 | AMASK | R/W | 0 | Access Interrupt Mask<br>This bit controls the reporting of the access raw interrupt status to the interrupt controller.<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the ARIS bit is set.<br>0 The ARIS interrupt is suppressed and not sent to the interrupt controller. |

## 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 R/W1C, 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 | PMISC | R/W1C | 0 | Programming Masked Interrupt Status and Clear<br>Value Description<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 305).<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. |
| 0 | AMISC | R/W1C | 0 | Access Masked Interrupt Status and Clear<br>Value Description<br>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 305).<br>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. |

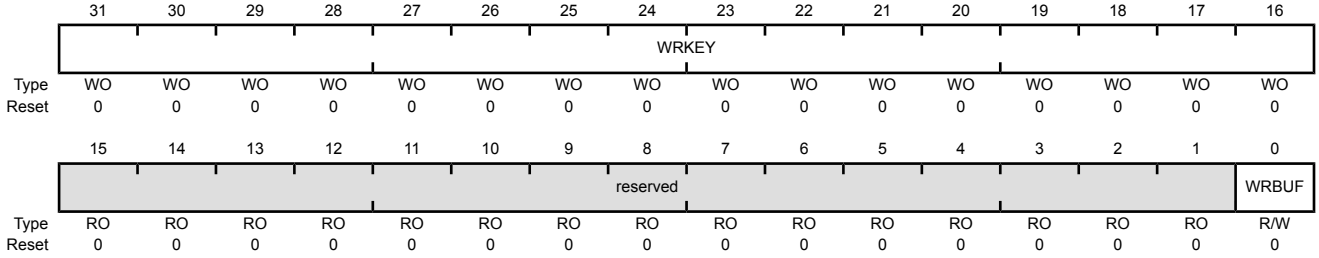
## Register 7: Flash Memory Control 2 (FMC2), offset 0x020

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 300). 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 R/W, 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 must be written into this field for a write to occur. 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 | R/W | 0 | Buffered Flash Memory Write<br>This bit is used to start a buffered write to Flash memory.<br>Value Description<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>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>For information on programming time, see “Flash Memory ” on page 795. |

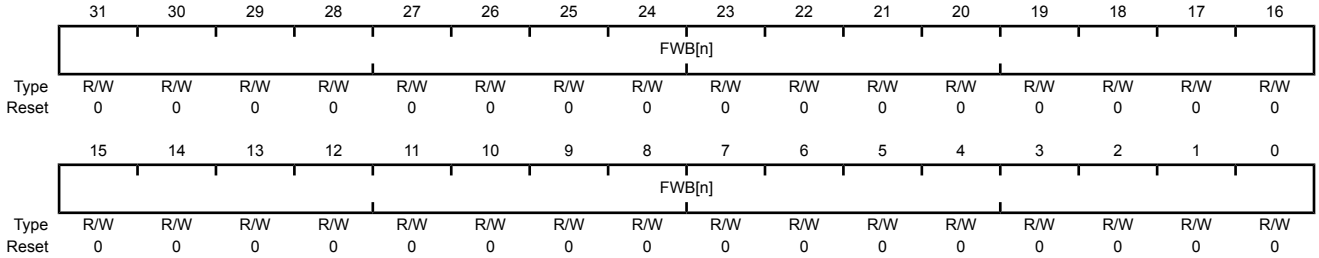
## Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030

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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | FWB[n] | R/W | 0x0 | Flash Memory Write Buffer |

| Value | Description |
|---|---|
| 1 | The corresponding **FWBn** register has been updated since the last buffer write operation and is ready to be written to Flash memory. |
| 0 | The corresponding **FWBn** register has no new data to be written. |

Bit 0 corresponds to **FWB0**, offset 0x100, and bit 31 corresponds to **FWB31**, offset 0x13C.

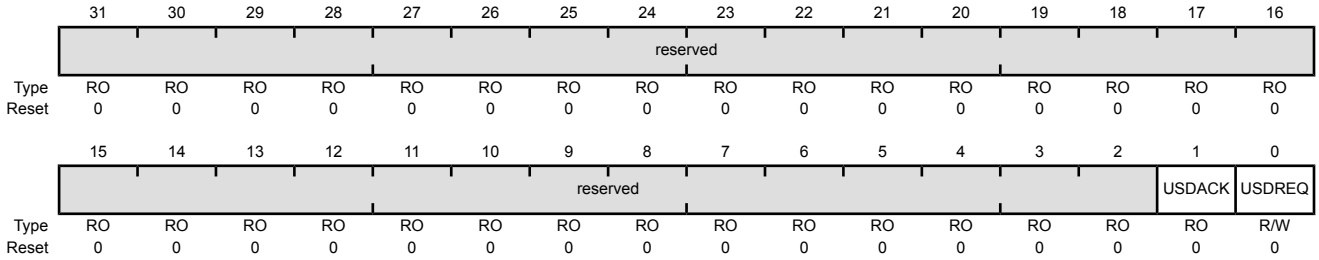
## Register 9: Flash Control (FCTL), offset 0x0F8

This register is used to ensure that the microcontroller is powered down in a controlled fashion in systems where power is cycled more frequently than once every five minutes. The USDREQ bit should be set to indicate that power is going to be turned off. Software should poll the USDACK bit to determine when it is acceptable to power down.

Note that this power-down process is not required if the microcontroller enters hibernation mode prior to power being removed.

Flash Control (FCTL)

Base 0x400F.D000
Offset 0x0F8
Type R/W, reset 0x0000.0000



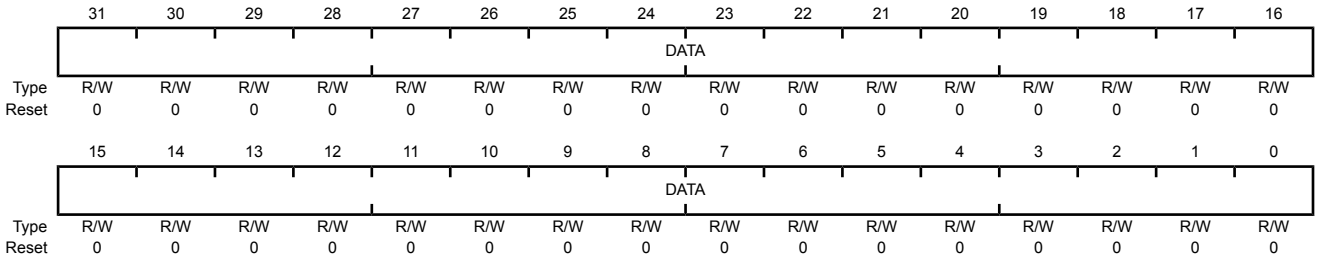

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | USDACK | RO | 0 | User Shut Down Acknowledge<br><br>Value Description<br>1 The microcontroller can be powered down.<br>0 The microcontroller cannot yet be powered down.<br><br>This bit should be set within 50 ms of setting the USDREQ bit. |
| 0 | USDREQ | R/W | 0 | User Shut Down Request<br><br>Value Description<br>1 Requests permission to power down the microcontroller.<br>0 No effect. |

### 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | DATA | R/W | 0x0000.0000 | Data<br>Data to be written into the Flash memory. |

## 7.5 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 11: ROM Control (RMCTL), offset 0x0F0

This register provides control of the ROM controller state. This register offset is relative to the System Control base address of 0x400F.E000.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The BA bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the EN bit is clear, 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 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.

ROM Control (RMCTL)

Base 0x400F.E000
Offset 0x0F0
Type R/W1C, 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 | | | | | | | | | | | | | | | BA |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C |
| 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 | 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 | BA | R/W1C | 1 | Boot Alias<br><br>Value Description<br>1 The microcontroller's ROM appears at address 0x0.<br>0 The Flash memory is at address 0x0.<br><br>This bit is cleared by writing a 1 to this bit position. |

## Register 12: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

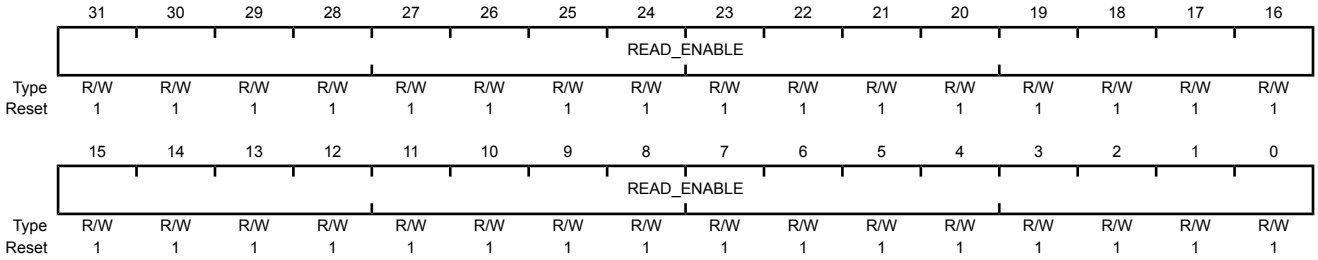
**Note:** This register is aliased for backwards compatability.

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 0 (FMPRE0)

Base 0x400F.E000
Offset 0x130 and 0x200
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | R/W | 0xFFFFFFFF | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value: 0xFFFFFFFF — Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB. |

## Register 13: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

**Note:** This register is aliased for backwards compatability.

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 0 (FMPPE0)

Base 0x400F.E000
Offset 0x134 and 0x400
Type R/W, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | PROG_ENABLE | R/W | 0xFFFFFFFF | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value: 0xFFFFFFFF — Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB. |

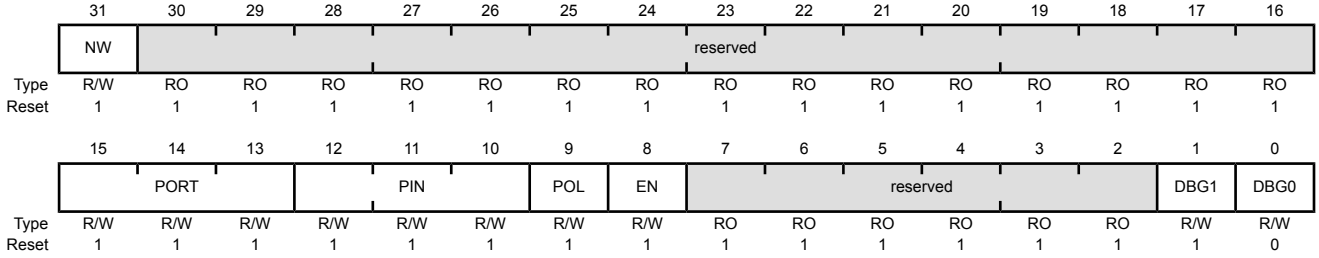
## Register 14: Boot Configuration (BOOTCFG), offset 0x1D0

**Note:** Offset is relative to System Control base address of 0x400FE000.

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. Upon 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-H as configured by the bits in this register. If the EN bit is set or the specified pin does not have the required polarity, the system control module checks address 0x000.0004 to see if the Flash memory has a valid reset vector. If the data at address 0x0000.0004 is 0xFFFF.FFFF, then it is assumed that the Flash memory has not yet been programmed, and the core executes the ROM Boot Loader. The DBG0 bit (bit 0) is set to 0 from the factory and the DBG1 bit (bit 1) is set to 1, which enables external debuggers. Clearing the DBG1 bit disables any external debugger access to the device permanently, starting with the next power-up cycle of the device. The NW bit (bit 31) indicates that the register has not yet been committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. 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.

Boot Configuration (BOOTCFG)

Base 0x400F.E000
Offset 0x1D0
Type R/W, reset 0xFFFF.FFFE




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NW | R/W | 1 | Not Written<br>When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again. |
| 30:16 | reserved | RO | 0x7FFF | Software should not rely on the value of 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 |
|---|---|---|---|---|
| 15:13 | PORT | R/W | 0x7 | Boot GPIO Port<br>This field selects the port of the GPIO port pin that enables the ROM boot loader at reset.<br><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 | R/W | 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><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 | R/W | 0x1 | 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. |
| 8 | EN | R/W | 0x1 | 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. |
| 7:2 | reserved | RO | 0x3F | Software should not rely on the value of 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 | R/W | 1 | Debug Control 1<br>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available. |
| 0 | DBG0 | R/W | 0x0 | Debug Control 0<br>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available. |

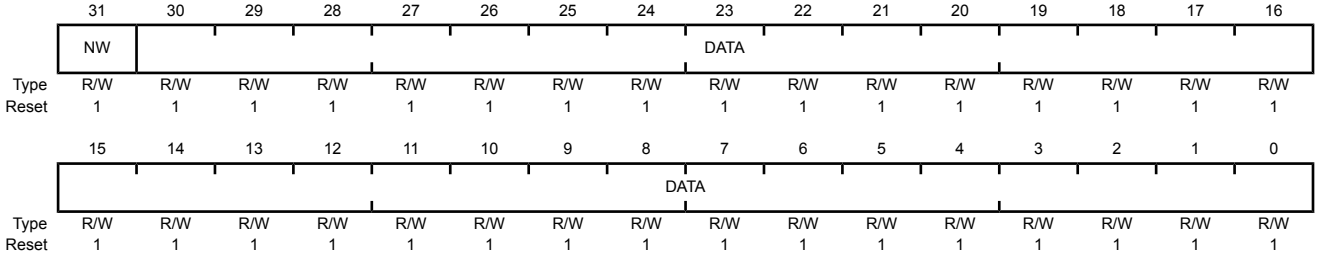
## Register 15: User Register 0 (USER_REG0), offset 0x1E0

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be committed once. Bit 31 indicates that the register is available to be committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device. 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 0 (USER_REG0)

Base 0x400F.E000
Offset 0x1E0
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NW | R/W | 1 | Not Written<br>When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again. |
| 30:0 | DATA | R/W | 0x7FFFFFFF | User Data<br>Contains the user data value. This field is initialized to all 1s and can only be committed once. |

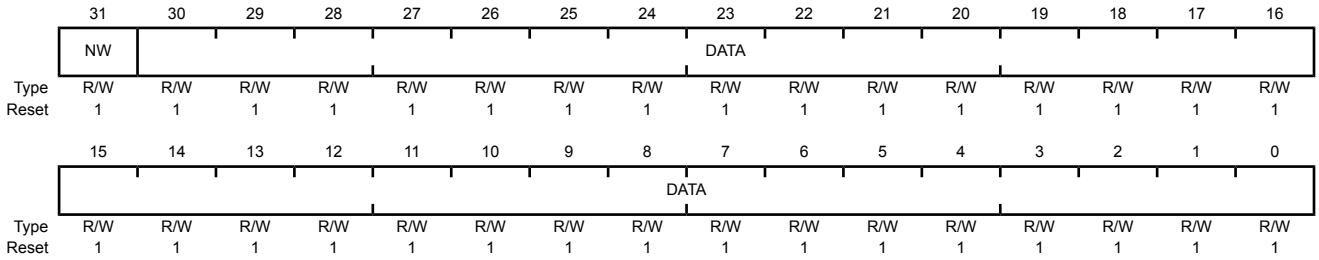
## Register 16: User Register 1 (USER_REG1), offset 0x1E4

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

User Register 1 (USER_REG1)

Base 0x400F.E000
Offset 0x1E4
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NW | R/W | 1 | Not Written<br>When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again. |
| 30:0 | DATA | R/W | 0x7FFFFFFF | User Data<br>Contains the user data value. This field is initialized to all 1s and can only be committed once. |

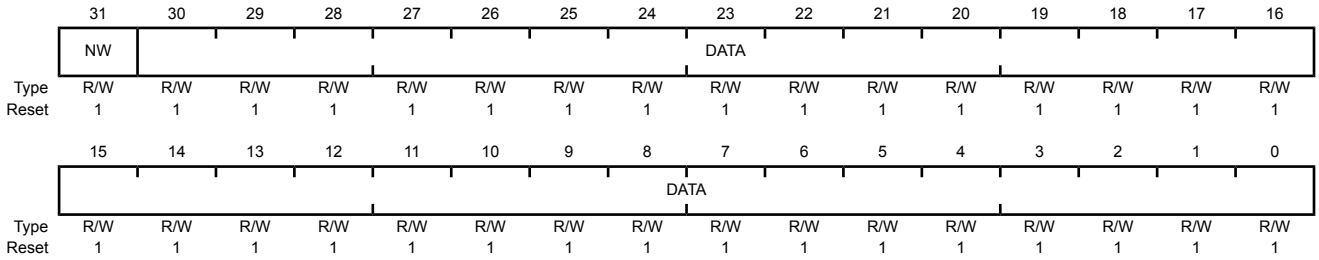
## Register 17: User Register 2 (USER_REG2), offset 0x1E8

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

User Register 2 (USER_REG2)

Base 0x400F.E000
Offset 0x1E8
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NW | R/W | 1 | Not Written<br>When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again. |
| 30:0 | DATA | R/W | 0x7FFFFFFF | User Data<br>Contains the user data value. This field is initialized to all 1s and can only be committed once. |

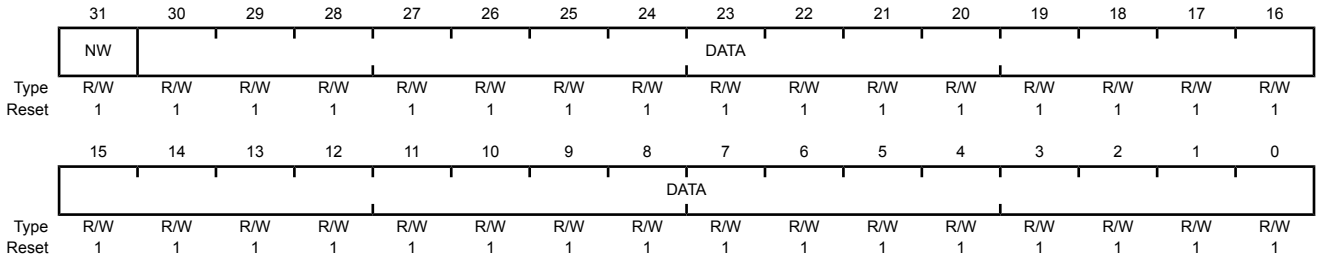
## Register 18: User Register 3 (USER_REG3), offset 0x1EC

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

User Register 3 (USER_REG3)

Base 0x400F.E000
Offset 0x1EC
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | NW | R/W | 1 | Not Written<br>When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again. |
| 30:0 | DATA | R/W | 0x7FFFFFFF | User Data<br>Contains the user data value. This field is initialized to all 1s and can only be committed once. |

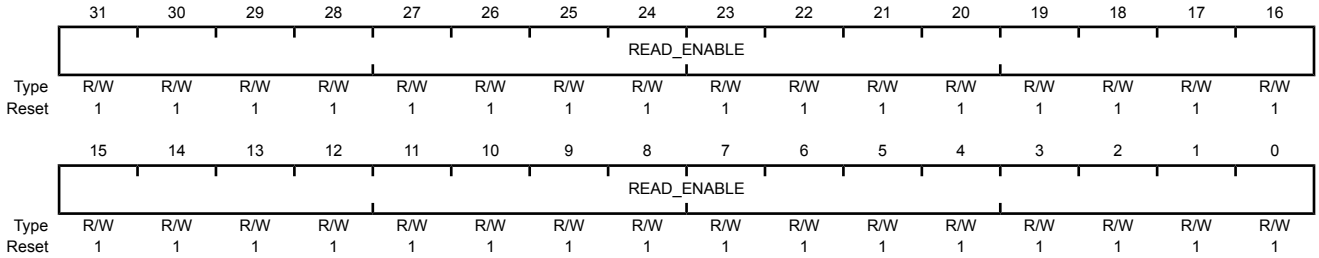
## Register 19: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 1 (FMPRE1)

Base 0x400F.E000
Offset 0x204
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | R/W | 0xFFFFFFFF | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB. |

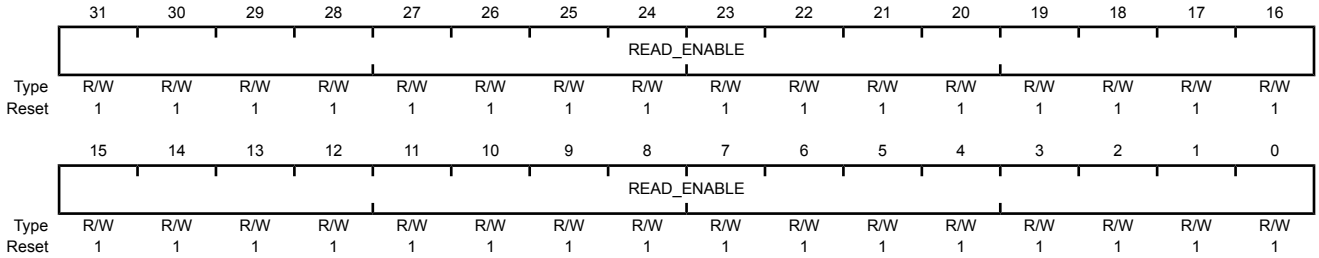
## Register 20: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 2 (FMPRE2)

Base 0x400F.E000
Offset 0x208
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | R/W | 0xFFFFFFFF | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value: 0xFFFFFFFF<br>Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB. |

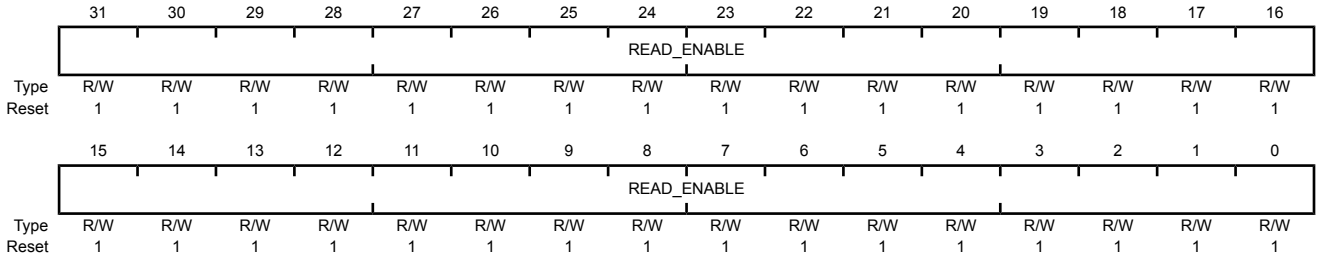
## Register 21: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 3 (FMPRE3)

Base 0x400F.E000
Offset 0x20C
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | R/W | 0xFFFFFFFF | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB. |

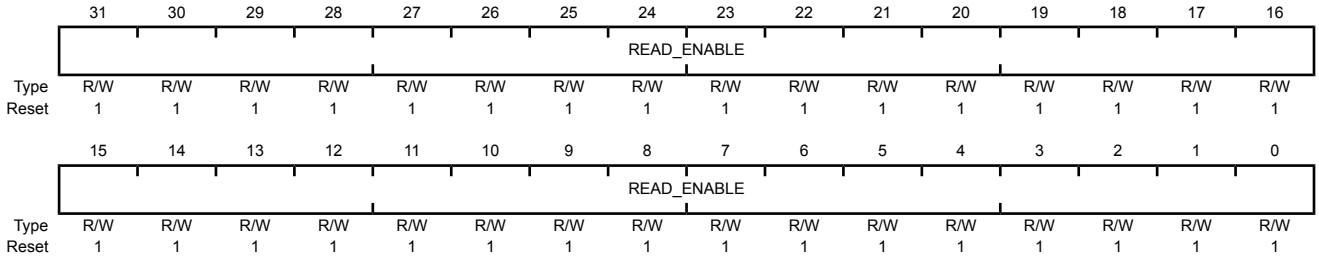
## Register 22: Flash Memory Protection Read Enable 4 (FMPRE4), offset 0x210

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 4 (FMPRE4)

Base 0x400F.E000
Offset 0x210
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | R/W | 0xFFFFFFFF | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value: 0xFFFFFFFF<br>Description: Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 257 to 320 KB. |

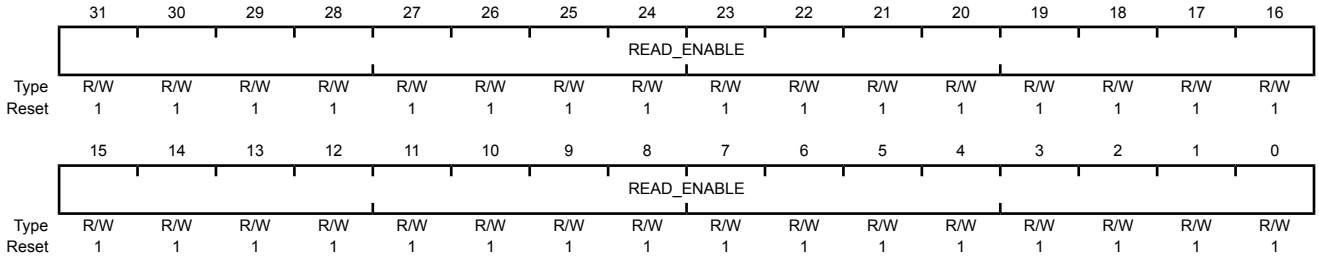
## Register 23: Flash Memory Protection Read Enable 5 (FMPRE5), offset 0x214

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 5 (FMPRE5)

Base 0x400F.E000
Offset 0x214
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | R/W | 0xFFFFFFFF | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 321 to 384 KB. |

## Register 24: Flash Memory Protection Read Enable 6 (FMPRE6), offset 0x218

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 6 (FMPRE6)

Base 0x400F.E000
Offset 0x218
Type R/W, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | READ_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | READ_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | READ_ENABLE | R/W | 0x00000000 | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 385 to 448 KB. |

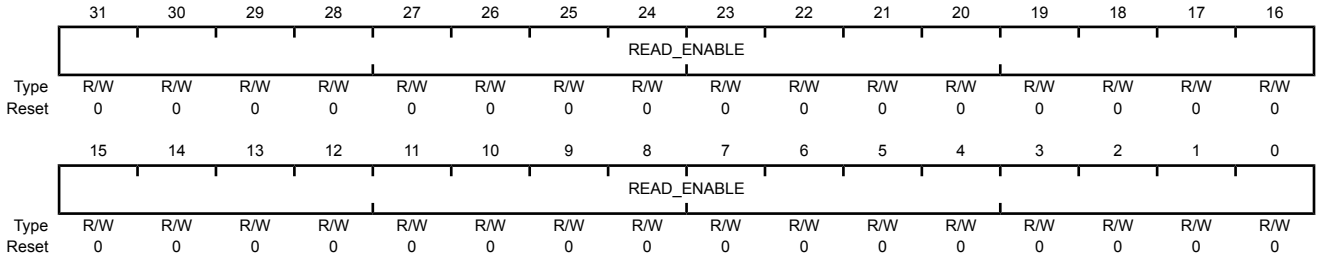
## Register 25: Flash Memory Protection Read Enable 7 (FMPRE7), offset 0x21C

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 7 (FMPRE7)

Base 0x400F.E000
Offset 0x21C
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | READ_ENABLE | R/W | 0x00000000 | Flash Read Enable<br>Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 449 to 512 KB. |

## Register 26: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 1 (FMPPE1)

Base 0x400F.E000
Offset 0x404
Type R/W, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | PROG_ENABLE | R/W | 0xFFFFFFFF | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB. |

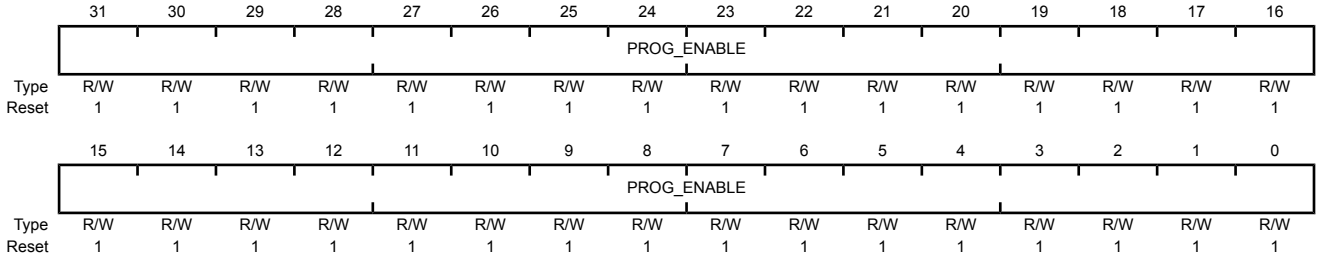
## Register 27: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 2 (FMPPE2)

Base 0x400F.E000
Offset 0x408
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PROG_ENABLE | R/W | 0xFFFFFFFF | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB. |

## Register 28: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 3 (FMPPE3)

Base 0x400F.E000
Offset 0x40C
Type R/W, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | PROG_ENABLE | R/W | 0xFFFFFFFF | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB. |

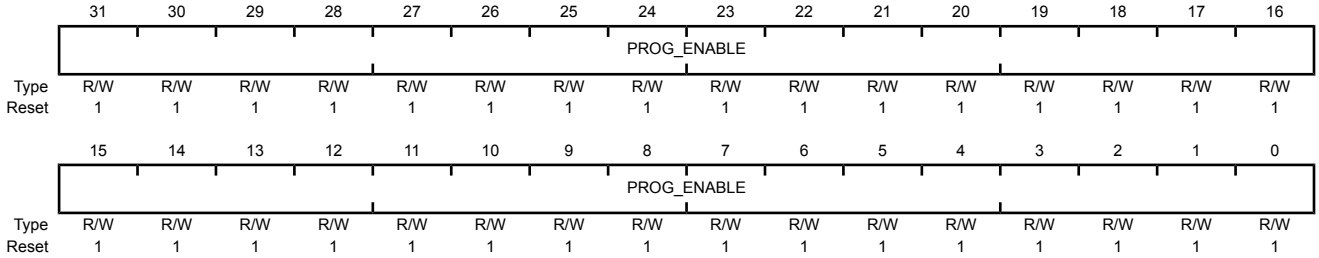
## Register 29: Flash Memory Protection Program Enable 4 (FMPPE4), offset 0x410

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 4 (FMPPE4)

Base 0x400F.E000
Offset 0x410
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PROG_ENABLE | R/W | 0xFFFFFFFF | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 257 to 320 KB. |

## Register 30: Flash Memory Protection Program Enable 5 (FMPPE5), offset 0x414

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 5 (FMPPE5)

Base 0x400F.E000
Offset 0x414
Type R/W, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | PROG_ENABLE | R/W | 0xFFFFFFFF | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 321 to 384 KB. |

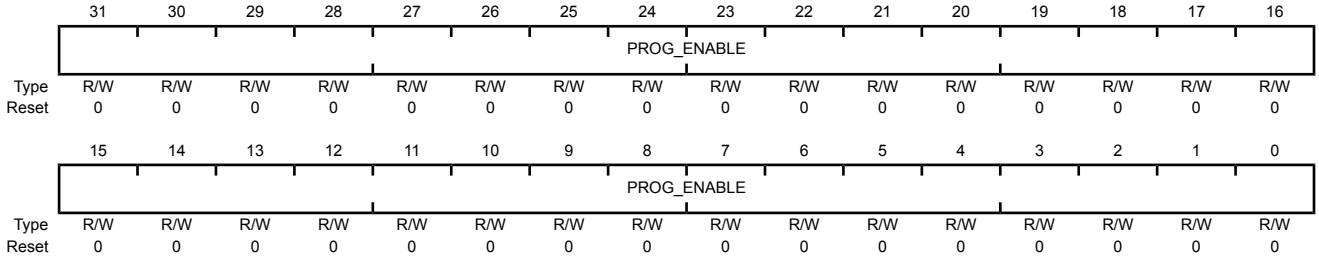
## Register 31: Flash Memory Protection Program Enable 6 (FMPPE6), offset 0x418

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 6 (FMPPE6)

Base 0x400F.E000
Offset 0x418
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | PROG_ENABLE | R/W | 0x00000000 | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 385 to 448 KB. |

## Register 32: Flash Memory Protection Program Enable 7 (FMPPE7), offset 0x41C

**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 execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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 R/W0; 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. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 7 (FMPPE7)

Base 0x400F.E000
Offset 0x41C
Type R/W, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PROG_ENABLE | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | PROG_ENABLE | R/W | 0x00000000 | Flash Programming Enable<br>Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.<br><br>Value Description<br>0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 449 to 512 KB. |

# 8 Micro Direct Memory Access (µDMA)

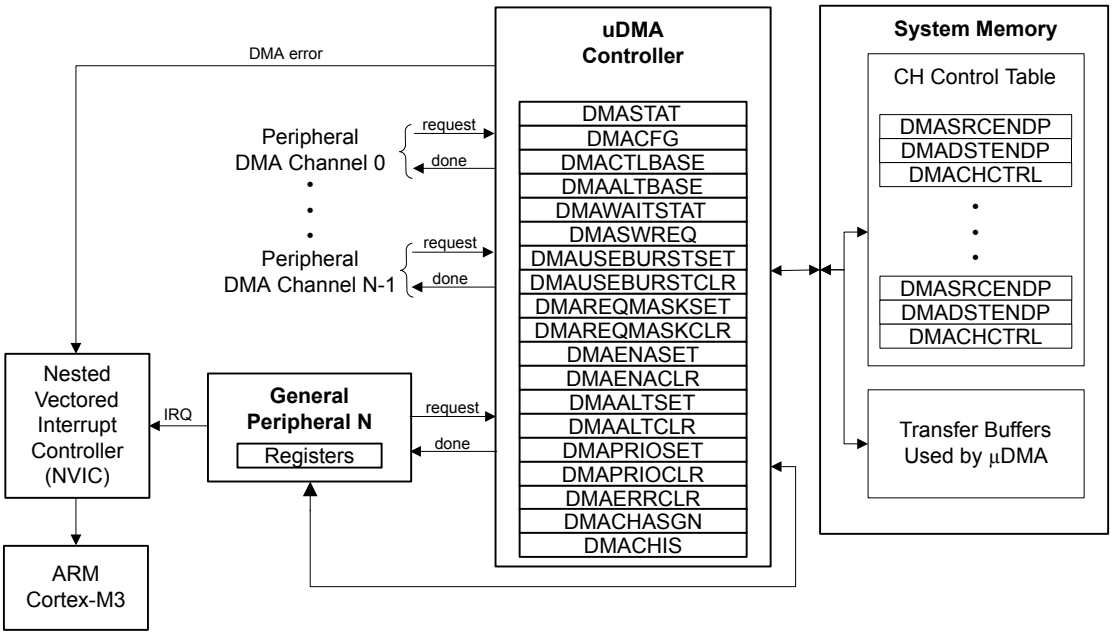
The LM3S1G58 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the Cortex™-M3 processor, allowing for more efficient use of the processor and the available bus bandwidth. The µDMA controller 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 arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules
  - Primary and secondary channel assignments
  - 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

## 8.1 Block Diagram

**Figure 8-1. µDMA Block Diagram**




## 8.2 Functional Description

The µDMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the microcontroller's Cortex-M3 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.

The µDMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the µDMA controller.

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 rearbitrates 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.

### 8.2.1 Channel Assignments

µDMA channels 0-31 are assigned to peripherals according to the following table. The **DMA Channel Assignment (DMACHASGN)** register (see page 384) can be used to specify the primary or secondary assignment. If the primary function is not available on this microcontroller, the secondary function becomes the primary function. If the secondary function is not available, the primary function is the only option.

**Note:** Channels noted in the table as "Available for software" may be assigned to peripherals in the future. However, they are currently available for software use. Channel 30 is dedicated for software use.

Because of the way the µDMA controller interacts with peripherals, the µDMA channel for the peripheral must be enabled in order for the µDMA controller to be able to read and write the peripheral registers, even if a different µDMA channel is used to perform the µDMA transfer. To minimize confusion and chance of software errors, it is best practice to use a peripheral's µDMA channel for performing all µDMA transfers for that peripheral, even if it is processor-triggered and using AUTO mode, which could be considered a software transfer. Note that if the software channel is used, interrupts occur on the dedicated µDMA interrupt vector. If the peripheral channel is used, then the interrupt occurs on the interrupt vector for the peripheral.

**Table 8-1. µDMA Channel Assignments**

| µDMA Channel | Primary Assignment | Secondary Assignment |
|---|---|---|
| 0 | Available for software | UART2 Receive |
| 1 | Available for software | UART2 Transmit |
| 2 | Available for software | General-Purpose Timer 3A |
| 3 | Available for software | General-Purpose Timer 3B |
| 4 | Available for software | General-Purpose Timer 2A |
| 5 | Available for software | General-Purpose Timer 2B |
| 6 | Available for software | General-Purpose Timer 2A |
| 7 | Available for software | General-Purpose Timer 2B |
| 8 | UART0 Receive | UART1 Receive |
| 9 | UART0 Transmit | UART1 Transmit |
| 10 | SSI0 Receive | SSI1 Receive |
| 11 | SSI0 Transmit | SSI1 Transmit |
| 12 | Available for software | UART2 Receive |
| 13 | Available for software | UART2 Transmit |
| 14 | ADC0 Sample Sequencer 0 | General-Purpose Timer 2A |
| 15 | ADC0 Sample Sequencer 1 | General-Purpose Timer 2B |
| 16 | ADC0 Sample Sequencer 2 | Available for software |
| 17 | ADC0 Sample Sequencer 3 | Available for software |
| 18 | General-Purpose Timer 0A | General-Purpose Timer 1A |
| 19 | General-Purpose Timer 0B | General-Purpose Timer 1B |

**Table 8-1. µDMA Channel Assignments *(continued)***

| µDMA Channel | Primary Assignment | Secondary Assignment |
|---|---|---|
| 20 | General-Purpose Timer 1A | Available for software |
| 21 | General-Purpose Timer 1B | Available for software |
| 22 | UART1 Receive | Available for software |
| 23 | UART1 Transmit | Available for software |
| 24 | SSI1 Receive | ADC1 Sample Sequencer 0 |
| 25 | SSI1 Transmit | ADC1 Sample Sequencer 1 |
| 26 | Available for software | ADC1 Sample Sequencer 2 |
| 27 | Available for software | ADC1 Sample Sequencer 3 |
| 28 | Available for software | Available for software |
| 29 | Available for software | Available for software |
| 30 | Dedicated for software use | |
| 31 | Reserved | |

### 8.2.2 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.

### 8.2.3 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.

### 8.2.4 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 8-2 on page 339, which shows how each peripheral supports the two request types.

**Table 8-2. Request Type Support**

| Peripheral | Single Request Signal | Burst Request Signal |
|---|---|---|
| ADC | None | Sequencer IE bit |
| General-Purpose Timer | Raw interrupt pulse | None |
| 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) |

#### 8.2.4.1 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.

#### 8.2.4.2 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.

### 8.2.5 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 8-3 on page 340 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 8-3. 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 8-4 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 8-4. 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 358. 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.

## 8.2.6 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.

### 8.2.6.1 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.

### 8.2.6.2 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.

### 8.2.6.3 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.

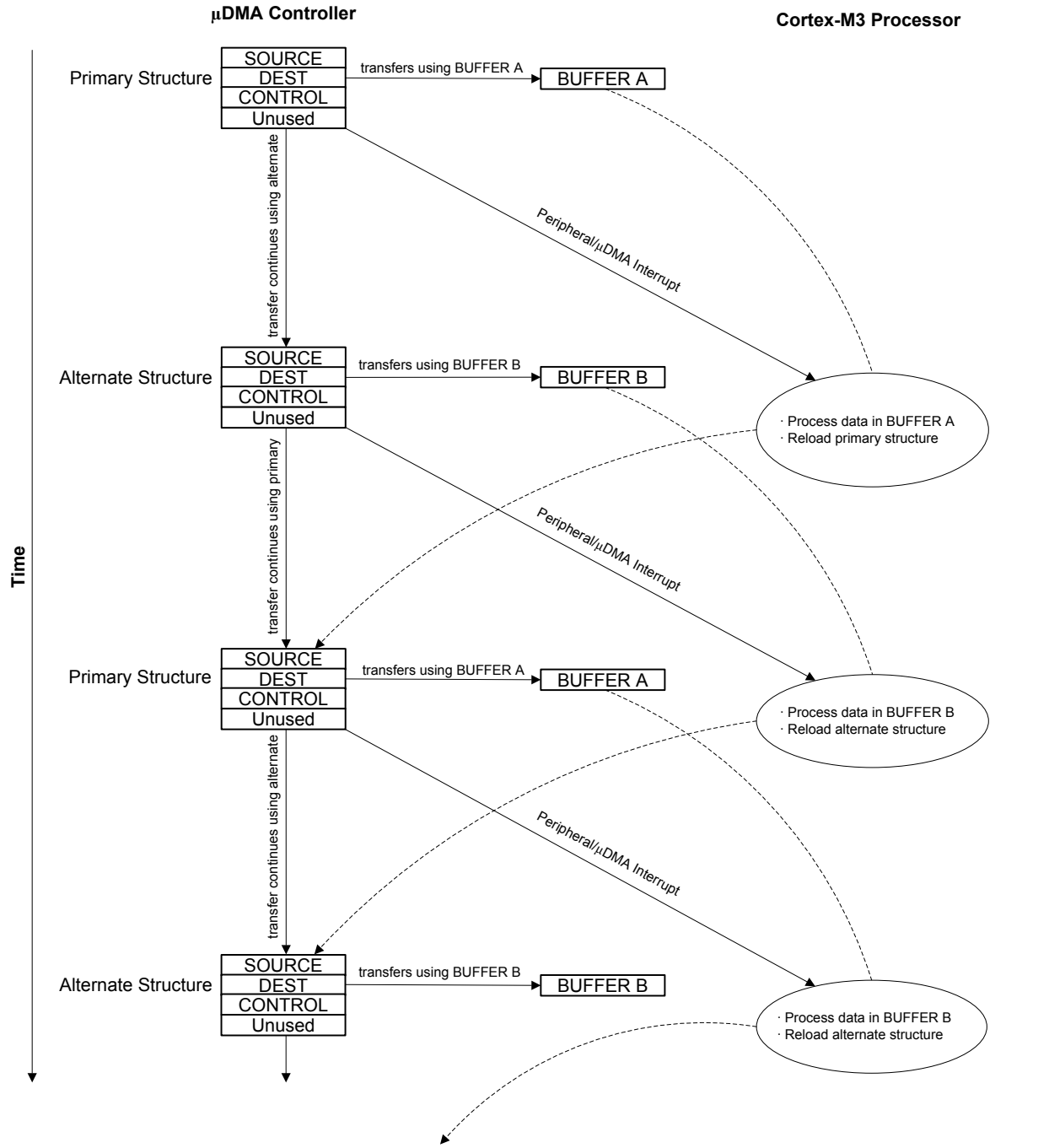
When all the items have been transferred using Auto mode, the µDMA controller sets the mode for that channel to Stop.

### 8.2.6.4 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 8-2 on page 342 for an example showing operation in Ping-Pong mode.

**Figure 8-2. Example of Ping-Pong µDMA Transaction**




### 8.2.6.5 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.

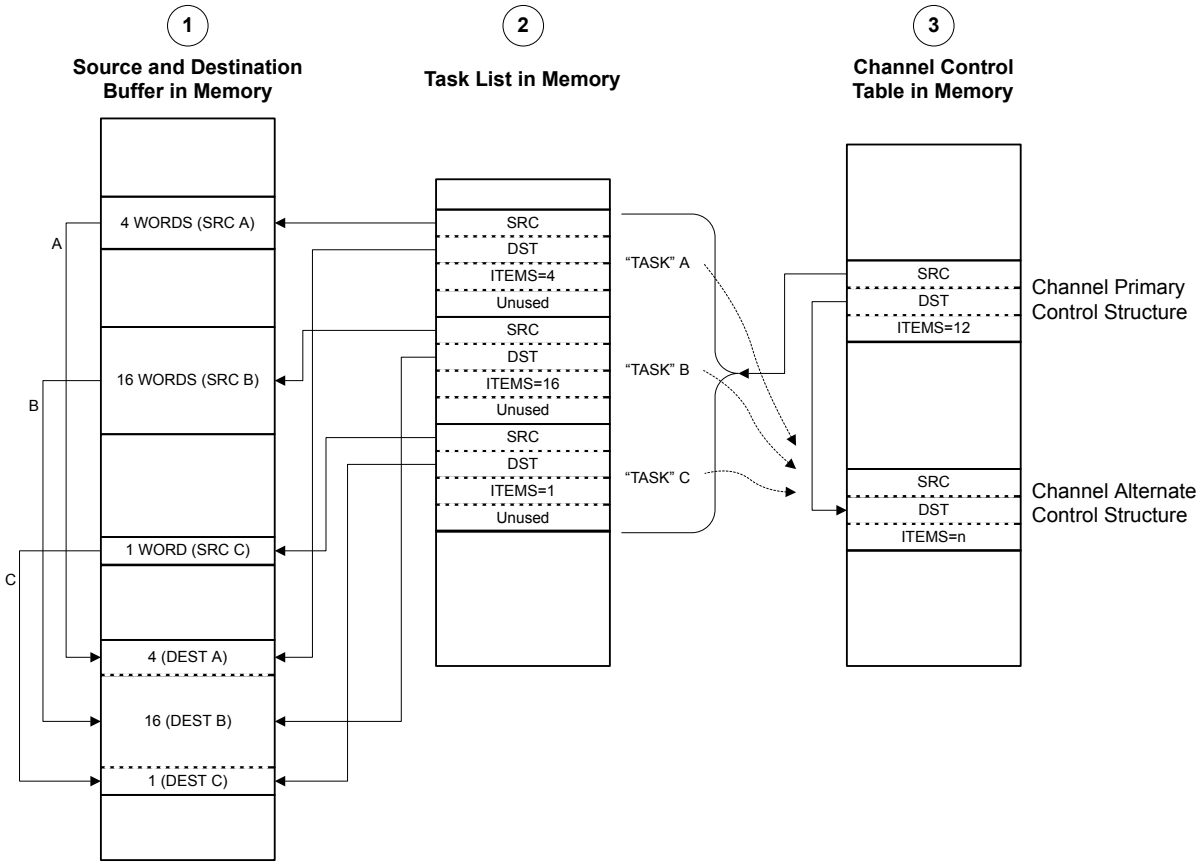
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word 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 arbitrary transfers can be performed based on a single µDMA request.

Refer to Figure 8-3 on page 344 and Figure 8-4 on page 345, 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 8-3 on page 344 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.

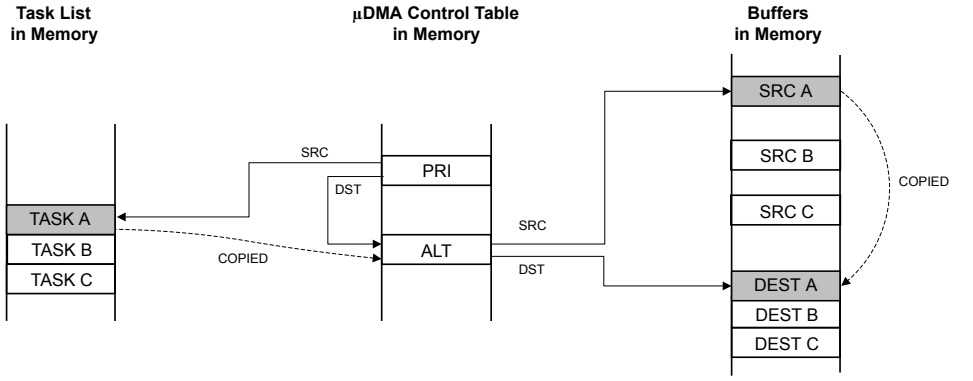
Figure 8-4 on page 345 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from 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 8-3. Memory Scatter-Gather, Setup and Configuration**



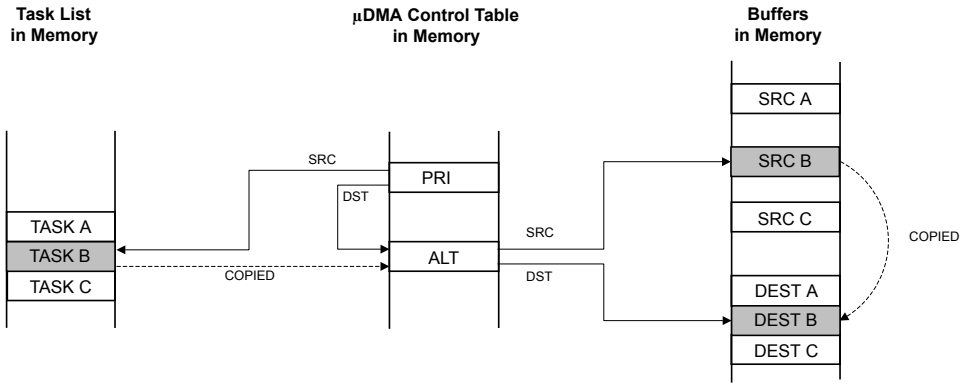

NOTES:
1. Application has a need to copy data items from three separate locations in memory into one combined buffer.
2. Application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the µDMA controller.

## Figure 8-4. Memory Scatter-Gather, µDMA Copy Sequence



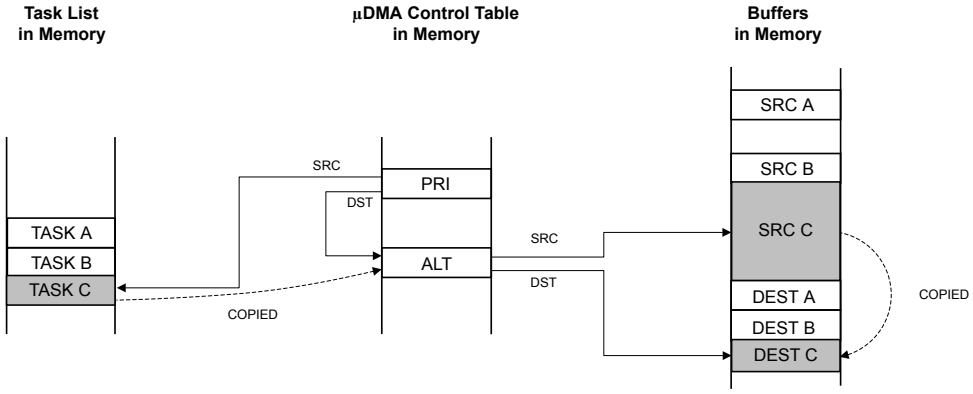

Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

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.

#### 8.2.6.6 Peripheral Scatter-Gather

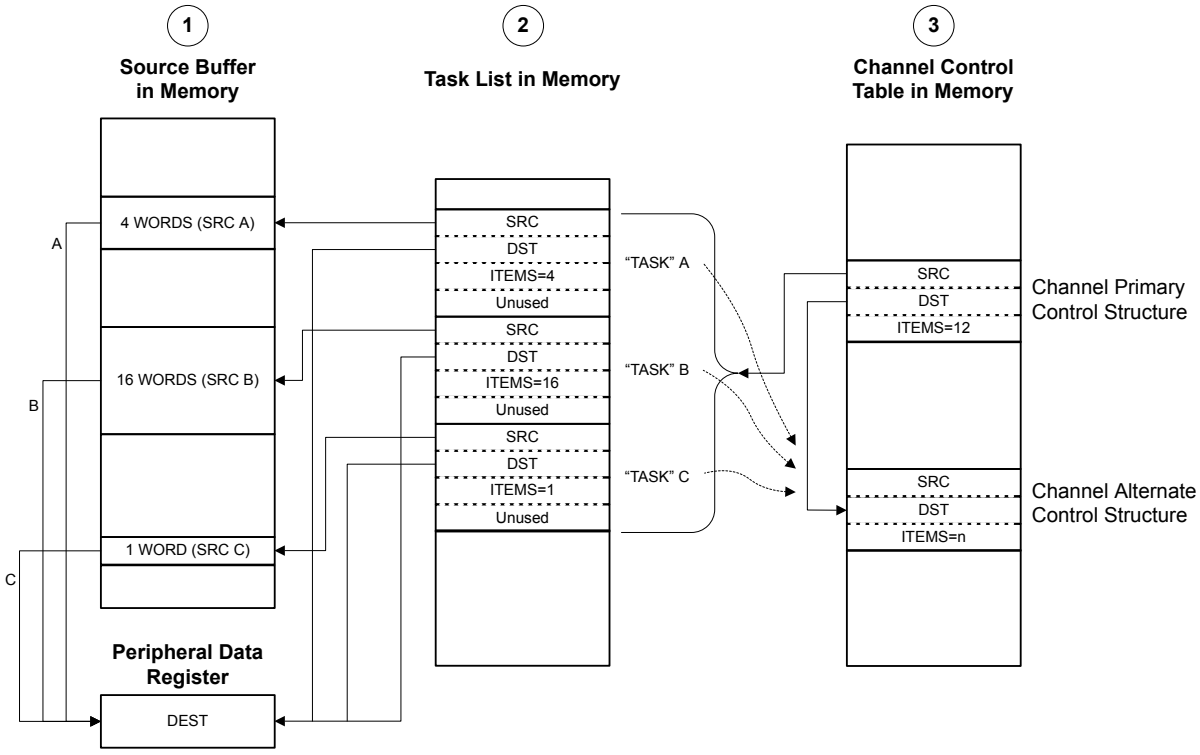
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a µDMA request. Upon detecting a request from the peripheral, the µDMA controller uses the primary control structure to copy one entry from the list to the alternate control structure and then performs the transfer. At the end of this transfer, the next transfer is started only if the peripheral again asserts a µDMA request. The µDMA controller continues to perform transfers from the list only when the peripheral is making a request, until the last transfer is complete. A completion interrupt is generated only after the last transfer.

By using this method, the µDMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Refer to Figure 8-5 on page 347 and Figure 8-6 on page 348, 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 8-5 on page 347 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.

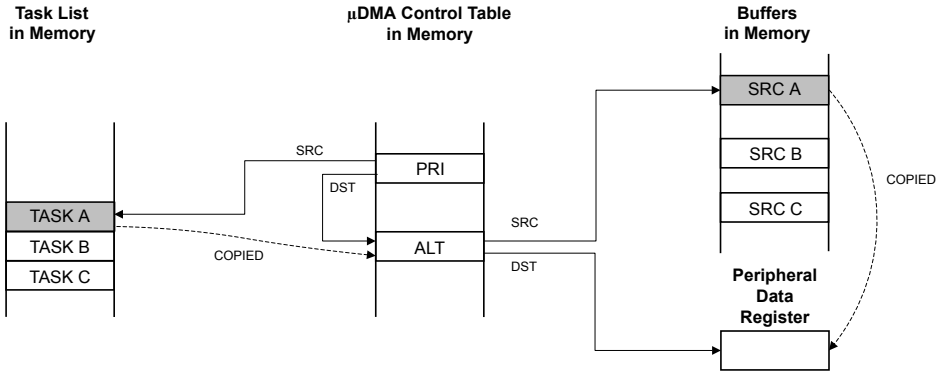
Figure 8-6 on page 348 shows the sequence as the µDMA controller performs the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from 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 8-5. Peripheral Scatter-Gather, Setup and Configuration**



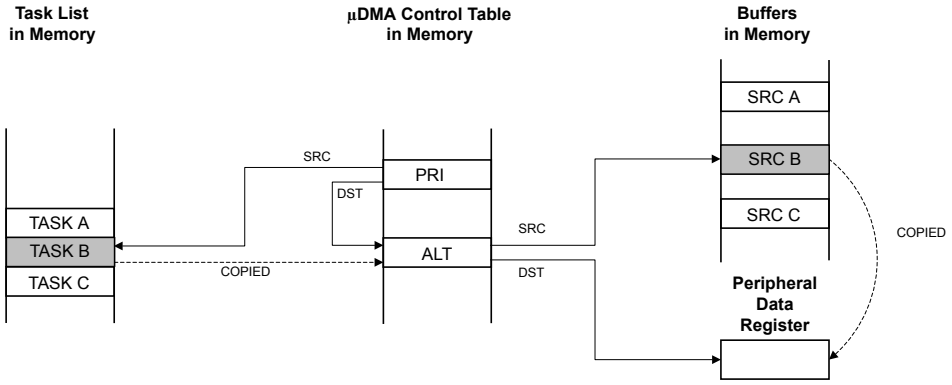

NOTES:
1. Application has a need to copy data items from three separate locations in memory into a peripheral data register.
2. Application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three µDMA copy "tasks."
3. Application sets up the channel primary 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 8-6. Peripheral Scatter-Gather, µDMA Copy Sequence



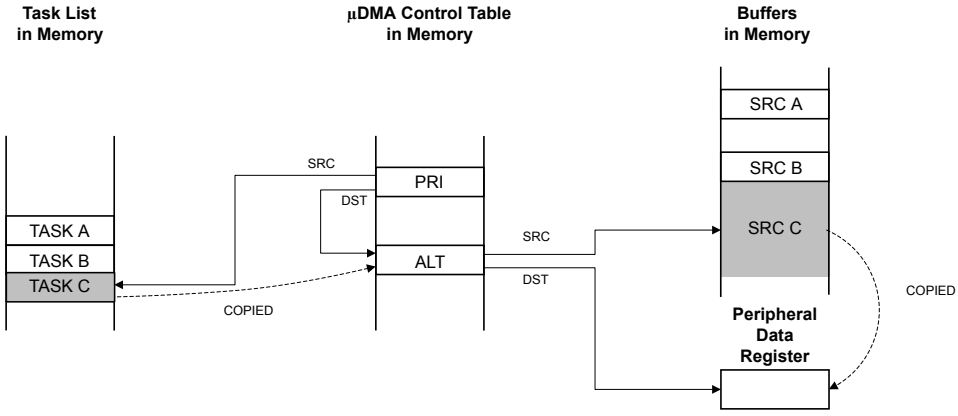

Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

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.

### 8.2.7 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 8-5 shows the configuration to read from a peripheral that supplies 8-bit data.

**Table 8-5. 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 |

### 8.2.8 Peripheral Interface

Each peripheral that supports µDMA has a single request and/or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 8-2 on page 339). The request signal can be disabled or enabled using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The µDMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the µDMA channel is configured correctly and enabled, and the peripheral asserts the request signal, the µDMA controller begins the transfer.

**Note:** When using µDMA to transfer data to and from a peripheral, the peripheral must disable all interrupts to the NVIC.

When a µDMA transfer is complete, the µDMA controller generates an interrupt, see “Interrupts and Errors” on page 350 for more information.

For more information on how a specific peripheral interacts with the µDMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

### 8.2.9 Software Request

One µDMA channel is dedicated to software-initiated transfers. This channel also has a dedicated interrupt to signal completion of a µDMA transfer. 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 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 channel may be used for software requests as long as the corresponding peripheral is not using µDMA for data transfer.

### 8.2.10 Interrupts and Errors

When a µDMA transfer is complete, the µDMA controller generates a completion interrupt on the interrupt vector of the peripheral. Therefore, if µDMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the µDMA transfer completion interrupt. If the transfer uses the software µDMA channel, then the completion interrupt occurs on the dedicated software µDMA interrupt vector (see Table 8-6 on page 350).

When µDMA is enabled for a peripheral, the µDMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (the interrupts are still reported in the peripheral's interrupt registers). Thus, when a large amount of data is transferred using µDMA, instead of receiving multiple interrupts from the peripheral as data flows, the interrupt controller receives only one interrupt when the transfer is complete. Unmasked peripheral error interrupts continue to be sent to the interrupt controller.

When a µDMA channel generates a completion interrupt, the CHIS bit corresponding to the peripheral channel is set in the **DMA Channel Interrupt Status (DMACHIS)** register (see page 385). This register can be used by the peripheral interrupt handler code to determine if the interrupt was caused by the µDMA channel or an error event reported by the peripheral's interrupt registers. The completion interrupt request from the µDMA controller is automatically cleared when the interrupt handler is activated.

If the µDMA controller encounters a bus or memory protection error as it 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 8-6 shows the dedicated interrupt assignments for the µDMA controller.

**Table 8-6. µDMA Interrupt Assignments**

| Interrupt | Assignment |
|---|---|
| 46 | µDMA Software Channel Transfer |
| 47 | µDMA Error |

## 8.3 Initialization and Configuration

### 8.3.1 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. The µDMA peripheral must be enabled in the System Control block. To do this, set the UDMA bit of the System Control **RCGC2** register (see page 251).
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.

### 8.3.2 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.

#### 8.3.2.1 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.

#### 8.3.2.2 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 8-7.

**Table 8-7. 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 8-8.

**Table 8-8. 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 | 23:18 | 0 | Reserved |

**Table 8-8. Channel Control Word Configuration for Memory Transfer Example *(continued)***

| Field in DMACHCTL | Bits | Value | Description |
|---|---|---|---|
| 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 |

#### 8.3.2.3 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.

### 8.3.3 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.

#### 8.3.3.1 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.

#### 8.3.3.2 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 8-9.

**Table 8-9. Channel Control Structure Offsets for Channel 7**

| Offset | Description |
|---|---|
| Control Table Base + 0x070 | Channel 7 Source End Pointer |

**Table 8-9. Channel Control Structure Offsets for Channel 7 *(continued)***

| Offset | Description |
|---|---|
| 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 8-10.

**Table 8-10. 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 | 23:18 | 0 | Reserved |
| 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 |

**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.

#### 8.3.3.3 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 interrupt handler receives an interrupt when the entire transfer is complete.

## 8.3.4 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.

### 8.3.4.1 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.

### 8.3.4.2 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 8-11.

**Table 8-11. 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 8-12.

2. Program the alternate channel control word at offset 0x288 according to Table 8-12.

**Table 8-12. 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 | 23:18 | 0 | Reserved |
| ARBSIZE | 17:14 | 3 | Arbitrates after 8 transfers |
| 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.

#### 8.3.4.3 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.

#### 8.3.4.4 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.

#### 8.3.4.5 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

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 8-12 on page 355.
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 8-12 on page 355.

### 8.3.5 Configuring Channel Assignments

Channel assignments for each µDMA channel can be changed using the **DMACHASGN** register. Each bit represents a µDMA channel. If the bit is set, then the secondary function is used for the channel.

Refer to Table 8-1 on page 337 for channel assignments.

For example, to use SSI1 Receive on channel 8 instead of UART0, set bit 8 of the **DMACHASGN** register.

## 8.4 Register Map

Table 8-13 on page 357 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, that is, the base address is n/a (not applicable). 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 339 and Table 8-3 on page 340 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 251). There must be a delay of 3 system clocks after the µDMA module clock is enabled before any µDMA module registers are accessed.

### Table 8-13. µDMA Register Map

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **µDMA Channel Control Structure (Offset from Channel Control Table Base)** | | | | | |
| 0x000 | DMASRCENDP | R/W | - | DMA Channel Source Address End Pointer | 359 |
| 0x004 | DMADSTENDP | R/W | - | DMA Channel Destination Address End Pointer | 360 |
| 0x008 | DMACHCTL | R/W | - | DMA Channel Control Word | 361 |
| **µDMA Registers (Offset from µDMA Base Address)** | | | | | |
| 0x000 | DMASTAT | RO | 0x001F.0000 | DMA Status | 366 |
| 0x004 | DMACFG | WO | - | DMA Configuration | 368 |
| 0x008 | DMACTLBASE | R/W | 0x0000.0000 | DMA Channel Control Base Pointer | 369 |
| 0x00C | DMAALTBASE | RO | 0x0000.0200 | DMA Alternate Channel Control Base Pointer | 370 |
| 0x010 | DMAWAITSTAT | RO | 0xFFFF.FFC0 | DMA Channel Wait-on-Request Status | 371 |
| 0x014 | DMASWREQ | WO | - | DMA Channel Software Request | 372 |
| 0x018 | DMAUSEBURSTSET | R/W | 0x0000.0000 | DMA Channel Useburst Set | 373 |
| 0x01C | DMAUSEBURSTCLR | WO | - | DMA Channel Useburst Clear | 374 |
| 0x020 | DMAREQMASKSET | R/W | 0x0000.0000 | DMA Channel Request Mask Set | 375 |
| 0x024 | DMAREQMASKCLR | WO | - | DMA Channel Request Mask Clear | 376 |
| 0x028 | DMAENASET | R/W | 0x0000.0000 | DMA Channel Enable Set | 377 |
| 0x02C | DMAENACLR | WO | - | DMA Channel Enable Clear | 378 |
| 0x030 | DMAALTSET | R/W | 0x0000.0000 | DMA Channel Primary Alternate Set | 379 |
| 0x034 | DMAALTCLR | WO | - | DMA Channel Primary Alternate Clear | 380 |
| 0x038 | DMAPRIOSET | R/W | 0x0000.0000 | DMA Channel Priority Set | 381 |
| 0x03C | DMAPRIOCLR | WO | - | DMA Channel Priority Clear | 382 |
| 0x04C | DMAERRCLR | R/W | 0x0000.0000 | DMA Bus Error Clear | 383 |
| 0x500 | DMACHASGN | R/W | 0x0000.0000 | DMA Channel Assignment | 384 |
| 0x504 | DMACHIS | R/W1C | 0x0000.0000 | DMA Channel Interrupt Status | 385 |
| 0xFD0 | DMAPeriphID4 | RO | 0x0000.0004 | DMA Peripheral Identification 4 | 390 |
| 0xFE0 | DMAPeriphID0 | RO | 0x0000.0030 | DMA Peripheral Identification 0 | 386 |
| 0xFE4 | DMAPeriphID1 | RO | 0x0000.00B2 | DMA Peripheral Identification 1 | 387 |
| 0xFE8 | DMAPeriphID2 | RO | 0x0000.000B | DMA Peripheral Identification 2 | 388 |
| 0xFEC | DMAPeriphID3 | RO | 0x0000.0000 | DMA Peripheral Identification 3 | 389 |
| 0xFF0 | DMAPCellID0 | RO | 0x0000.000D | DMA PrimeCell Identification 0 | 391 |
| 0xFF4 | DMAPCellID1 | RO | 0x0000.00F0 | DMA PrimeCell Identification 1 | 392 |
| 0xFF8 | DMAPCellID2 | RO | 0x0000.0005 | DMA PrimeCell Identification 2 | 393 |

**Table 8-13. µDMA Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xFFC | DMAPCellID3 | RO | 0x0000.00B1 | DMA PrimeCell Identification 3 | 394 |

## 8.5 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 339 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

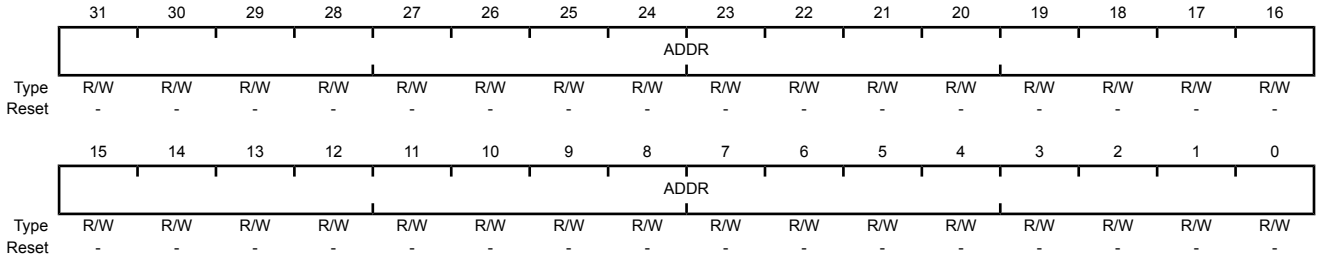
**DMA Channel Source Address End Pointer (DMASRCENDP)** is part of the Channel Control Structure and is used to specify the source address for a µDMA transfer.

The µDMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the µDMA controller.

**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 R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | R/W | - | 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). |

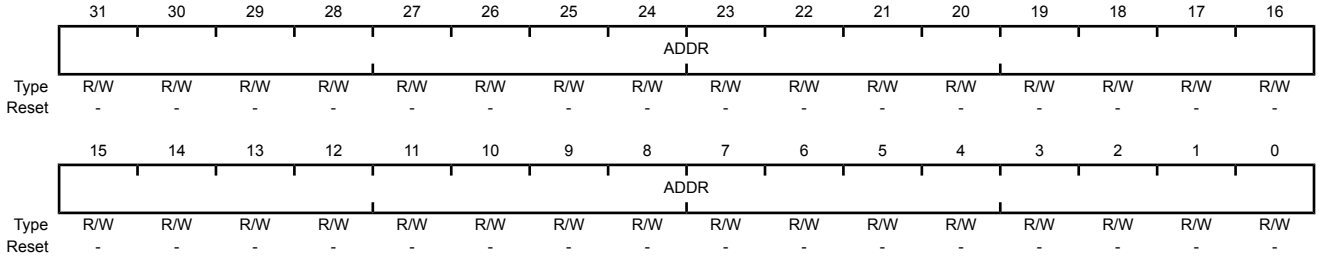
## Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

**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 R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | R/W | - | 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). |

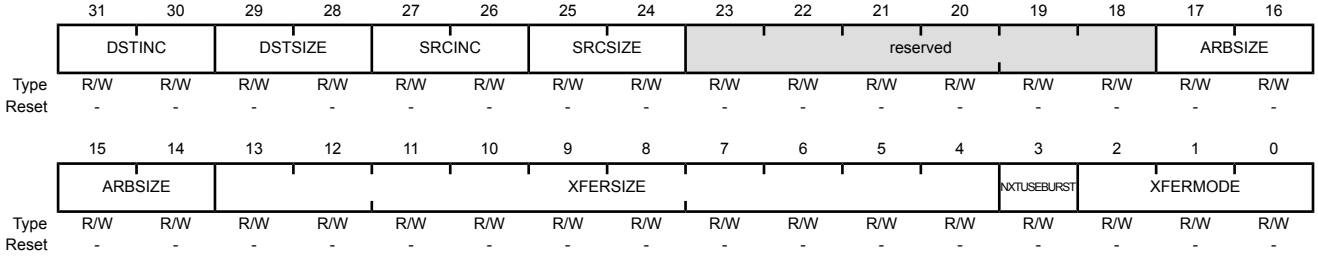
## Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

**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 R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:30 | DSTINC | R/W | - | 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 |
| 29:28 | DSTSIZE | R/W | - | Destination Data Size<br>This field configures the destination item data size.<br>**Note:** DSTSIZE must be the same as SRCSIZE.<br><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 |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 27:26 | SRCINC | R/W | - | 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><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 | R/W | - | Source Data Size<br>This field configures the source item data size.<br>**Note:** DSTSIZE must be the same as SRCSIZE.<br><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 |
| 23:18 | reserved | R/W | - | Software should not rely on the value of 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 |
|---|---|---|---|---|
| 17:14 | ARBSIZE | R/W | - | 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. |
| 13:4 | XFERSIZE | R/W | - | 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 | R/W | - | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 2:0 | XFERMODE | R/W | - | µDMA Transfer Mode<br>This field configures the operating mode of the µDMA cycle. Refer to "Transfer Modes" on page 341 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 341.

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 342.

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 346.

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.

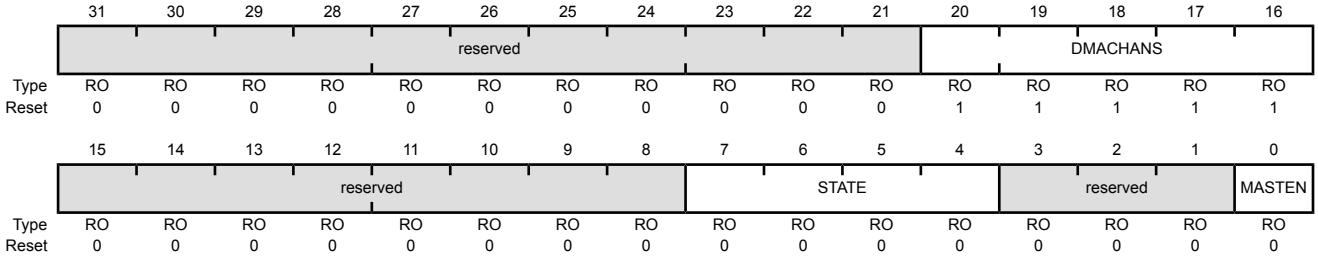
## 8.6 µDMA Register Descriptions

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 |

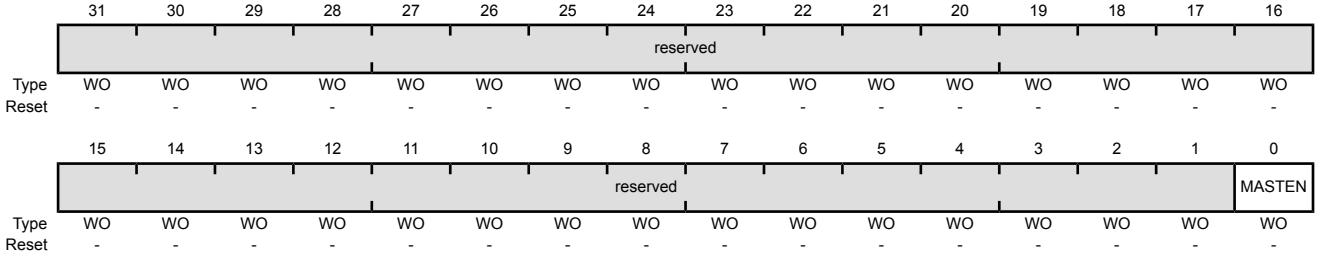
| Value | Description |
|---|---|
| 0 | The µDMA controller is disabled. |
| 1 | The µDMA controller is enabled. |

## 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. |

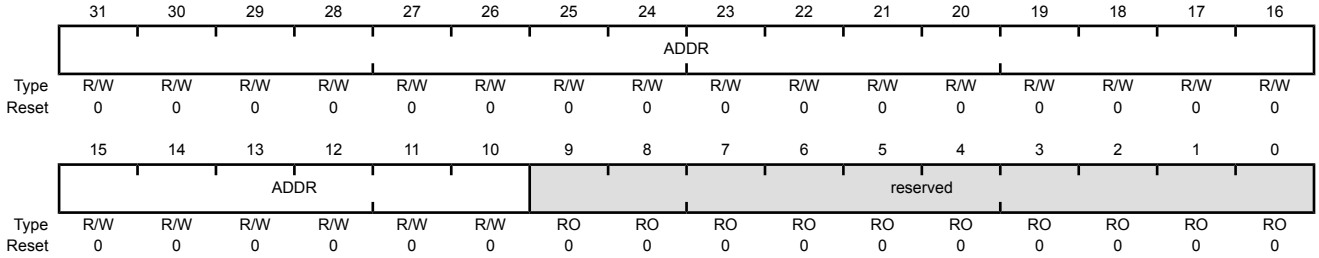
## Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

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 339 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 R/W, reset 0x0000.0000



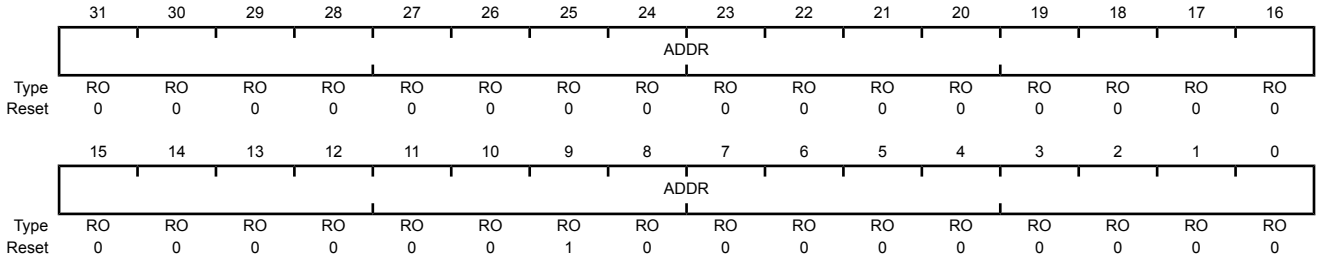

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:10 | ADDR | R/W | 0x0000.00 | Channel Control Base Address<br>This field contains the pointer to the base address of the channel control table. The base address must be 1024-byte aligned. |
| 9:0 | reserved | RO | 0x00 | Software 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



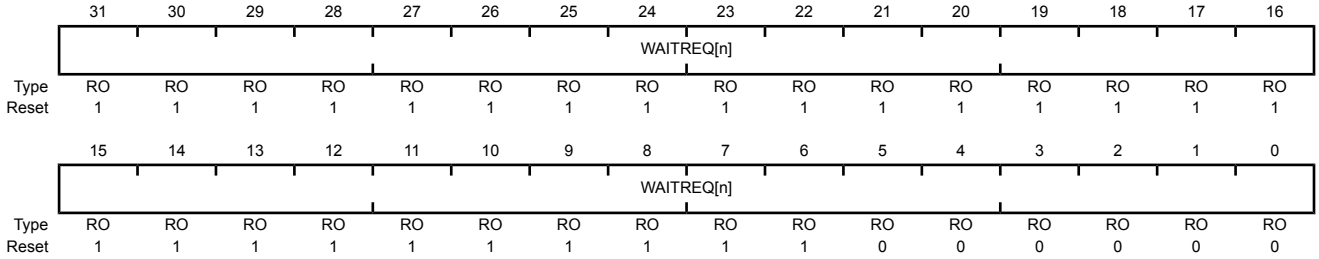

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | ADDR | RO | 0x0000.0200 | Alternate Channel Address Pointer<br>This field provides the base address of the alternate channel control structures. |

## 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 0xFFFF.FFC0



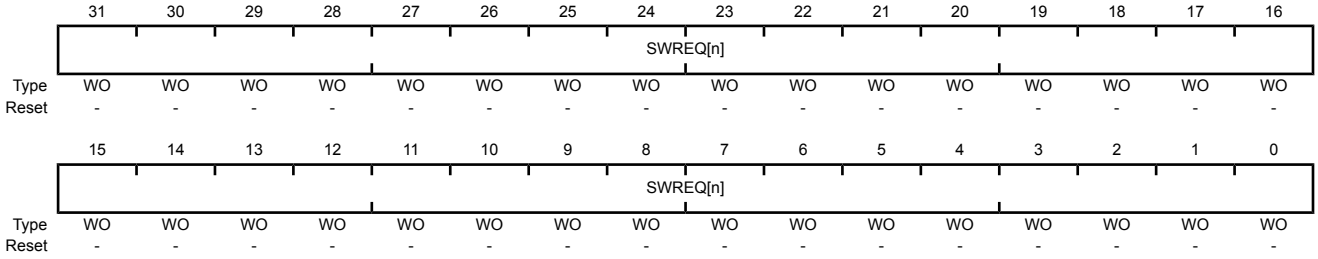

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WAITREQ[n] | RO | 0xFFFF.FFC0 | Channel [n] Wait Status<br>These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0.<br><br>Value Description<br>1 The corresponding channel is waiting on a request.<br>0 The corresponding channel is not 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>1 Generate a software request for the corresponding channel.<br>0 No request generated.<br><br>These bits are automatically cleared when the software request has been completed. |

## Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

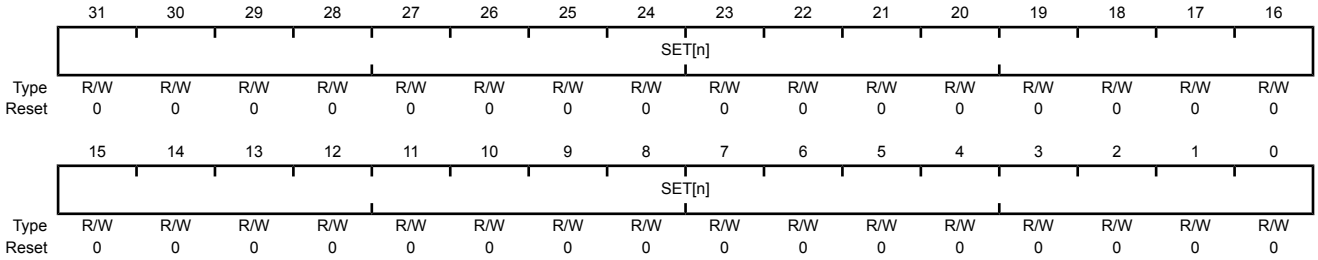
Each bit of the **DMAUSEBURSTSET** register represents the corresponding µDMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

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 338 for more details about request types.

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | R/W | 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. |

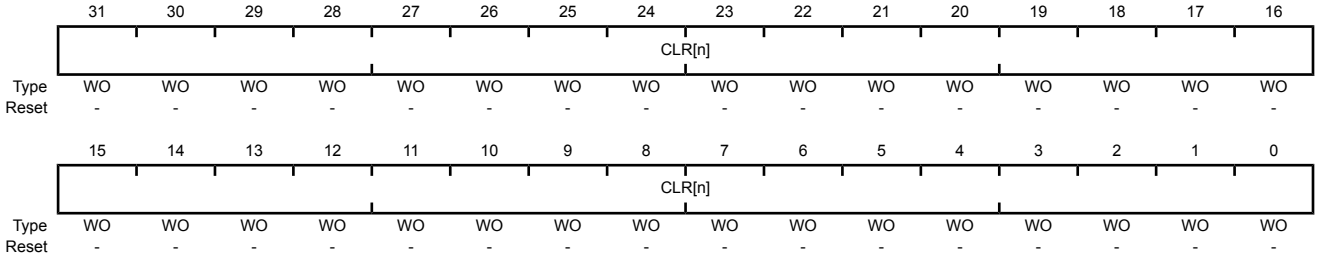
Bit 0 corresponds to channel 0. This bit is automatically cleared as described above. A bit can also be manually cleared by setting the corresponding CLR[n] bit in the **DMAUSEBURSTCLR** register.

## 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 |

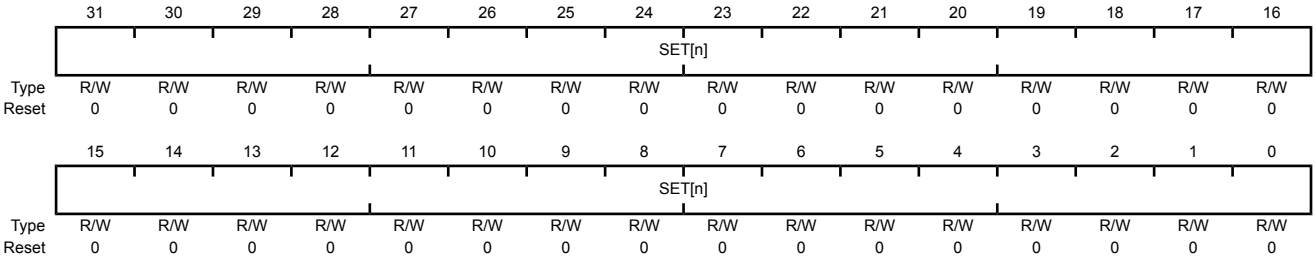
| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAUSEBURSTSET** register meaning that µDMA channel [n] responds to single and burst requests. |

## 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | R/W | 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. |

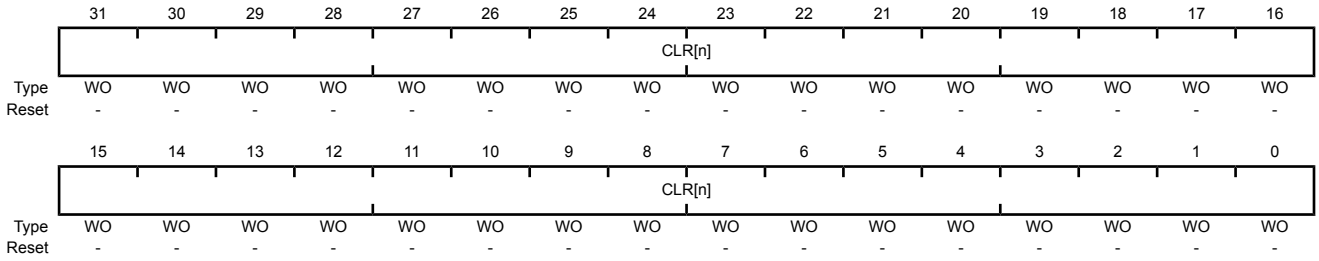
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAREQMASKCLR** register.

## 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 |

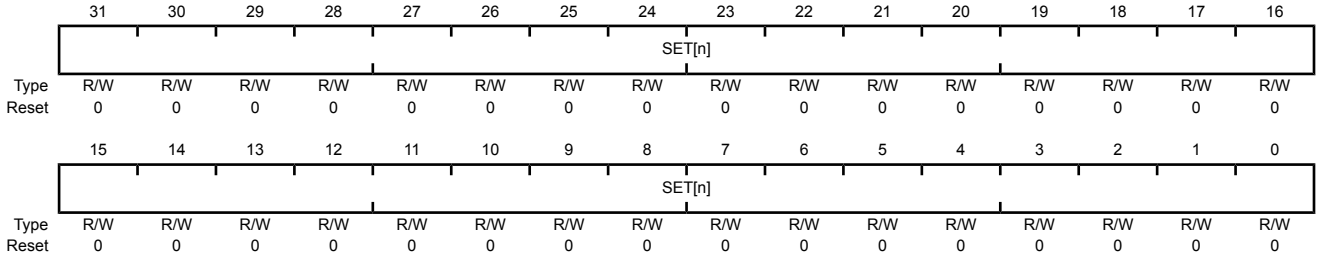
| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAREQMASKSET** register meaning that the peripheral associated with channel [n] is enabled to request µDMA transfers. |

## 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | R/W | 0x0000.0000 | Channel [n] Enable Set |

| Value | Description |
|---|---|
| 0 | µDMA Channel [n] is disabled. |
| 1 | µDMA Channel [n] is enabled. |

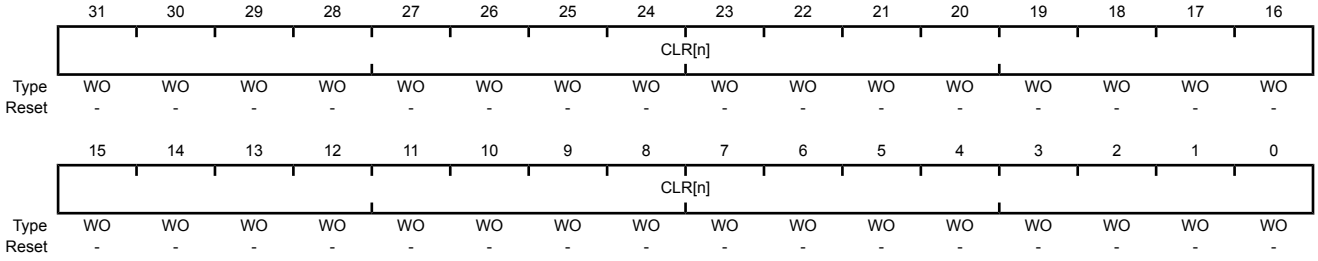
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAENACLR** register.

## 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. |

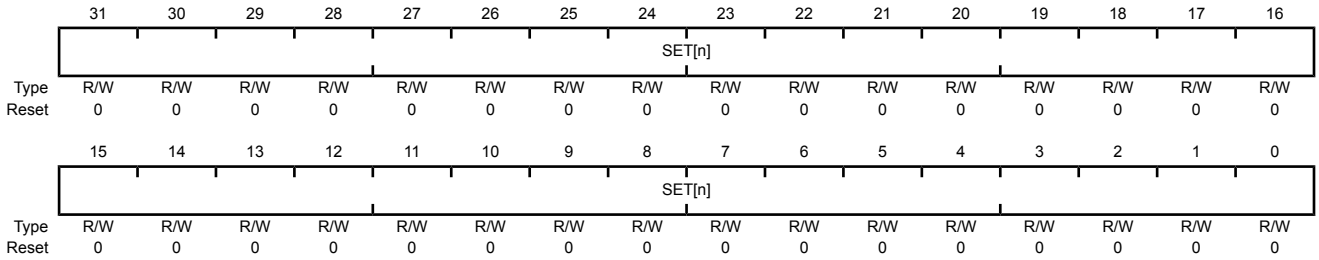
**Note:** The controller disables a channel when it completes the µDMA cycle.

## 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | R/W | 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.

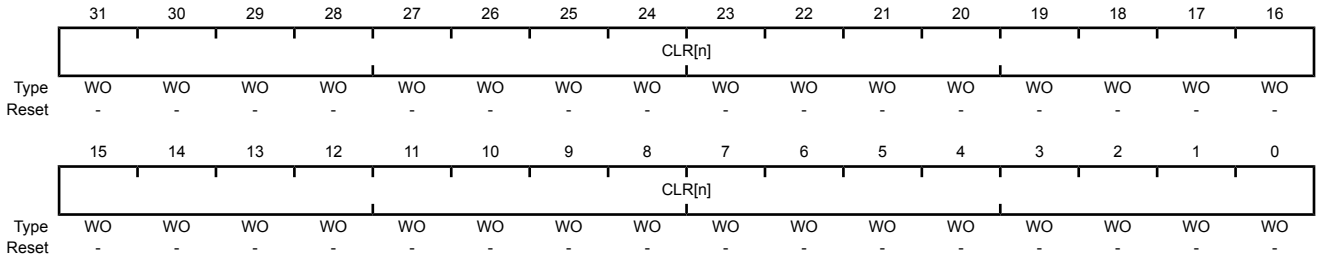
**Note:** For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.

## 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. |

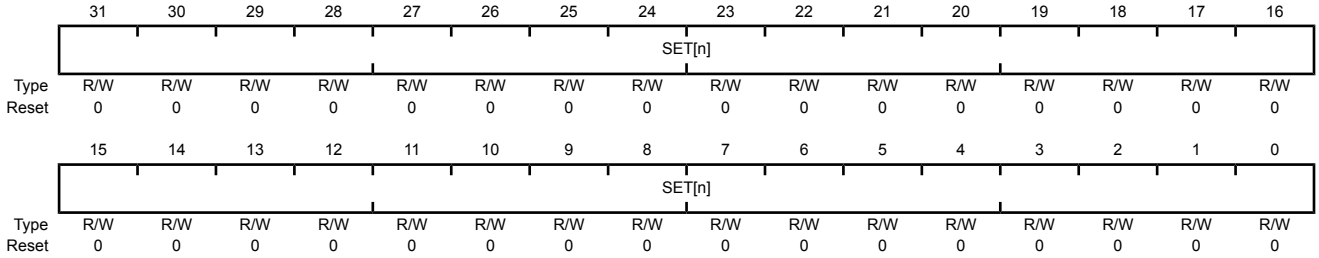
**Note:** For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.

## 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | SET[n] | R/W | 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. |

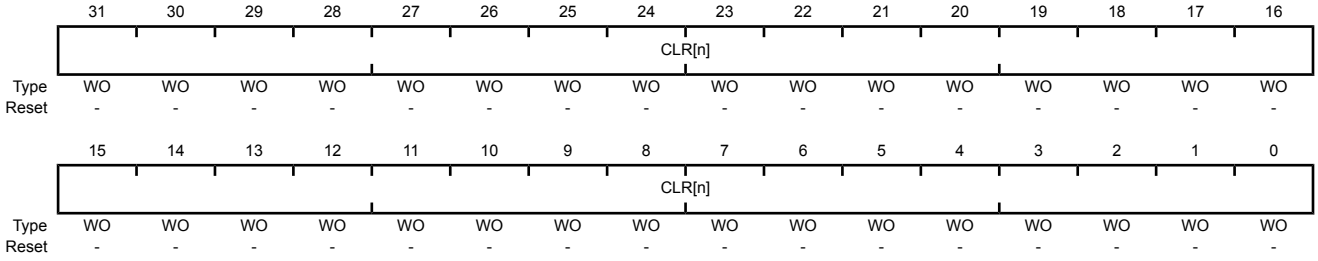
Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAPRIOCLR** register.

## 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 |

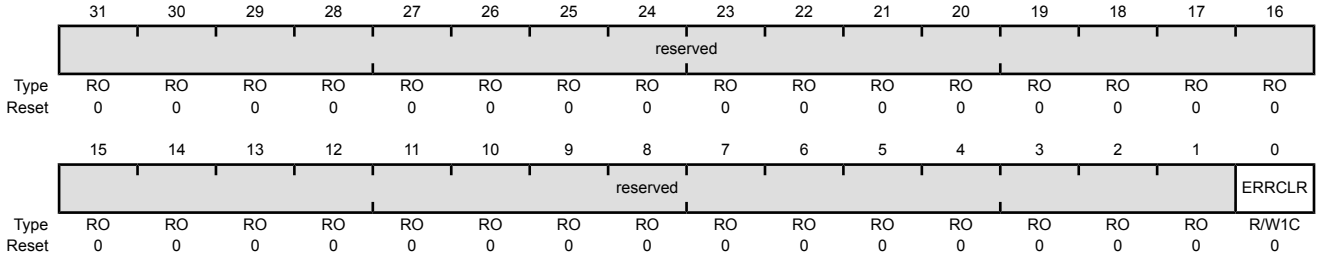
| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the **DMAPRIOSET** register meaning that channel [n] is using the default priority level. |

## 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 R/W, reset 0x0000.0000



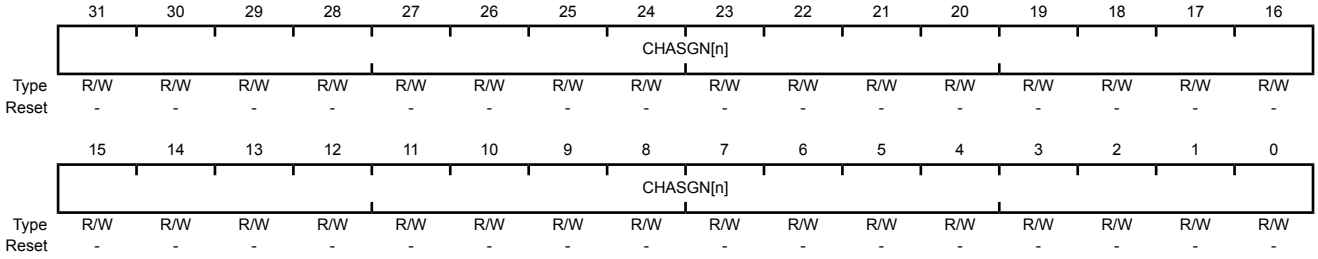

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | ERRCLR | R/W1C | 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 8-1 on page 337.

DMA Channel Assignment (DMACHASGN)

Base 0x400F.F000
Offset 0x500
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CHASGN[n] | R/W | - | Channel [n] Assignment Select |

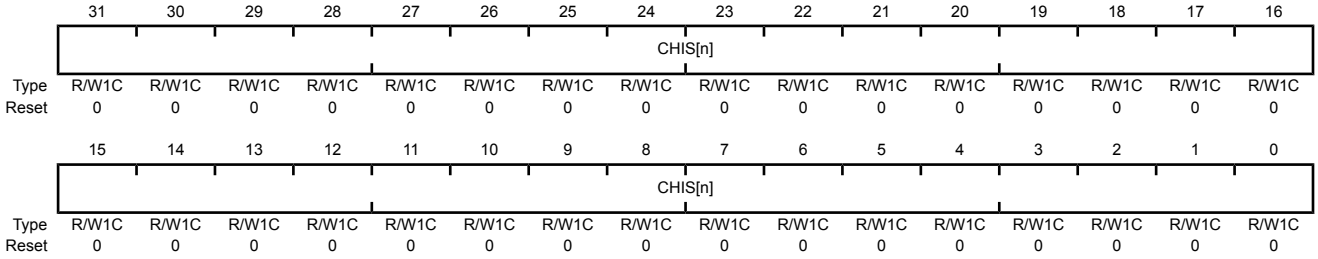
| Value | Description |
|---|---|
| 0 | Use the primary channel assignment. |
| 1 | Use the secondary channel assignment. |

## Register 22: DMA Channel Interrupt Status (DMACHIS), offset 0x504

Each bit of the **DMACHIS** register represents the corresponding µDMA channel. A bit is set when that µDMA channel causes a completion interrupt. The bits are cleared by a writing a 1.

DMA Channel Interrupt Status (DMACHIS)

Base 0x400F.F000
Offset 0x504
Type R/W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | CHIS[n] | R/W1C | 0x0000.0000 | Channel [n] Interrupt Status |

| Value | Description |
|---|---|
| 1 | The corresponding µDMA channel caused an interrupt. |
| 0 | The corresponding µDMA channel has not caused an interrupt. |

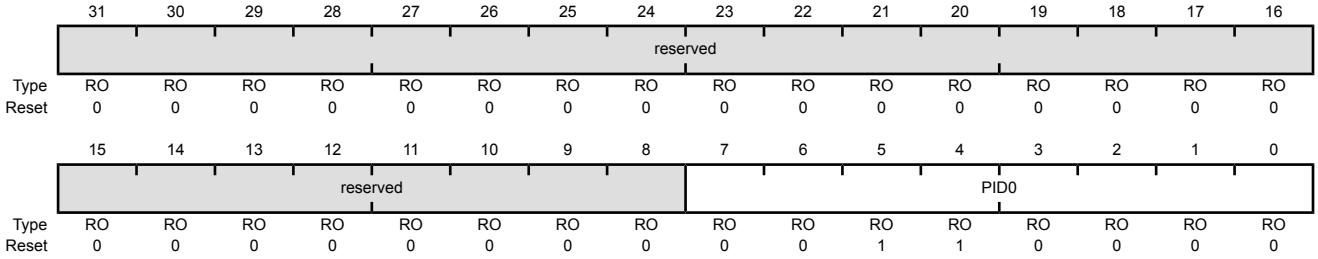
This bit is cleared by writing a 1 to it.

## Register 23: 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



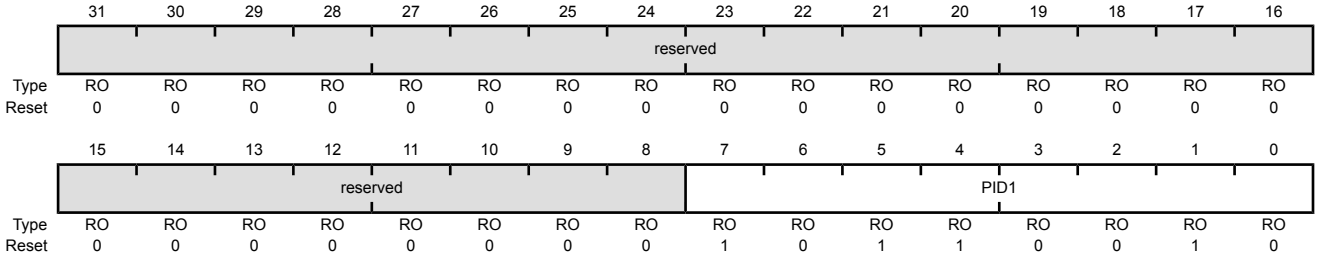

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 24: 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



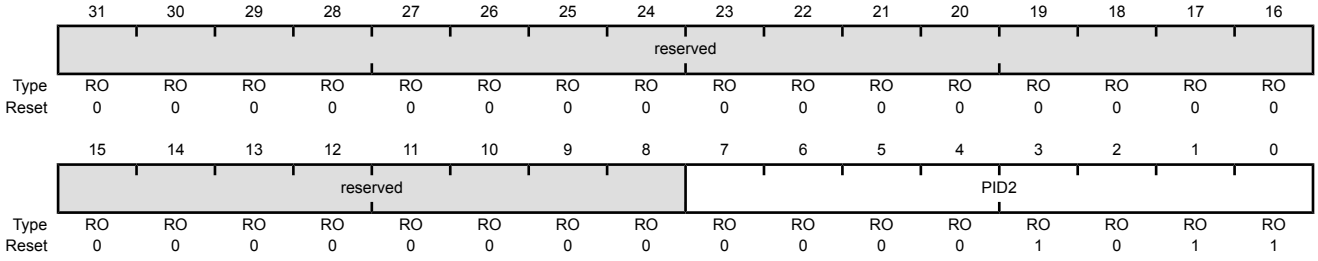

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 25: 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



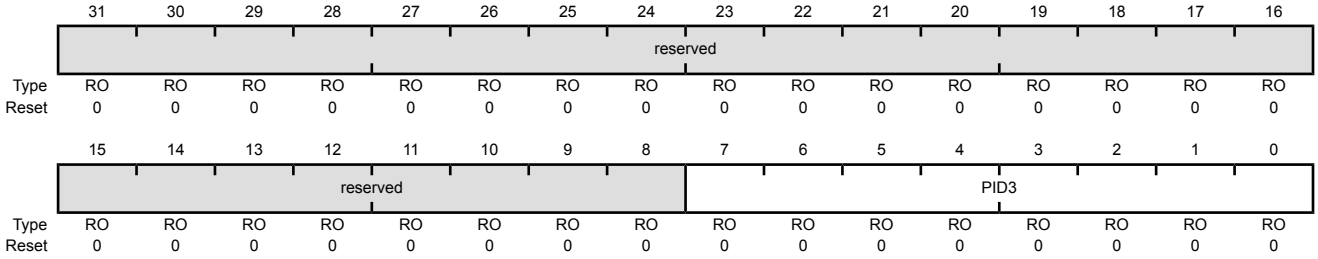

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 26: 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



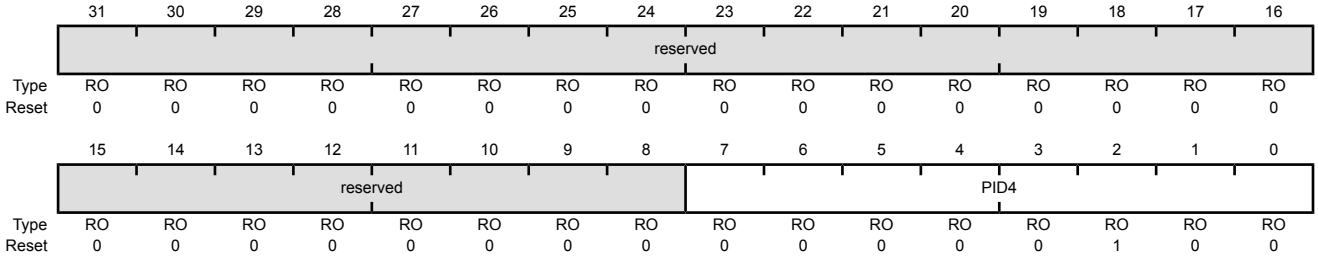

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 27: 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



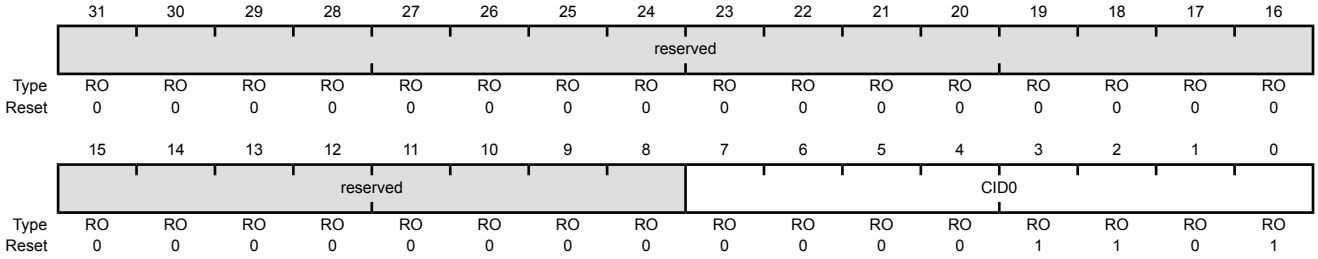

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x04 | µDMA Peripheral ID Register<br>Can be used by software to identify the presence of this peripheral. |

## Register 28: 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



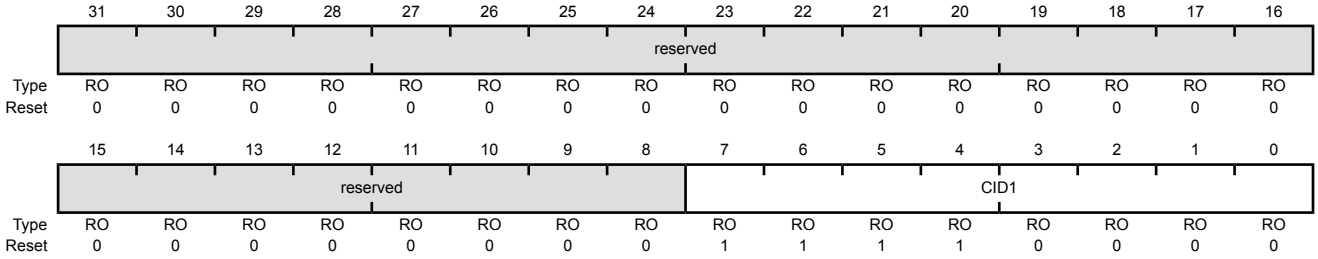

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | µDMA PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 29: 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



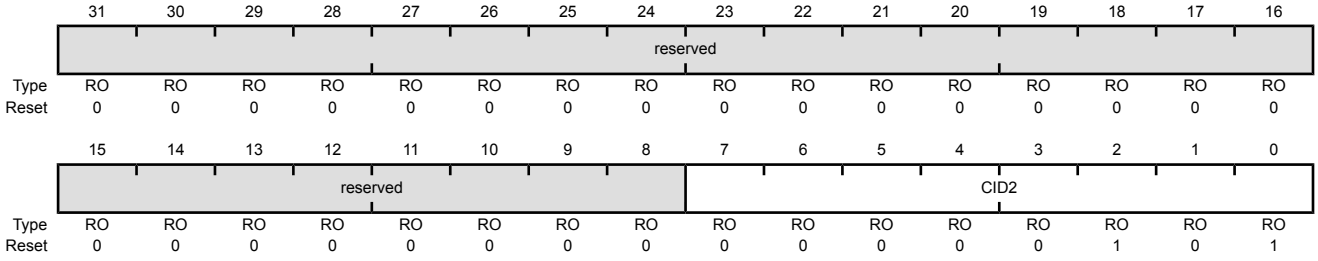

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | µDMA PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 30: 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



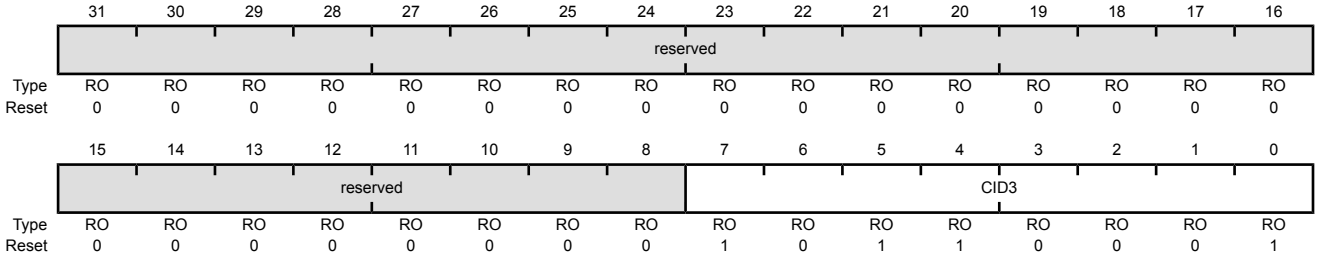

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | µDMA PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 31: 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. |

# 9 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of eight 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). The GPIO module supports up to 60 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 60 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced High-Performance Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- 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, and 8-mA pad drive for digital communication; up to four pads can sink 18-mA for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

## 9.1 Signal Description

GPIO signals have alternate hardware functions. The following table lists the GPIO pins and their analog and digital alternate functions. The AINx and VREFA analog signals are not 5-V tolerant and 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. All GPIO signals are 5-V tolerant when configured as inputs except for PB0 and PB1, which are limited to 3.6 V. 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. 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:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

**Table 9-1. GPIO Pins With Non-Zero Reset Values**

| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL |
|---|---|---|---|---|---|---|
| PA[1:0] | UART0 | 0 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 1 | 0 | 0 | 0x3 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 |

**Table 9-2. GPIO Pins and Alternate Functions (100LQFP)**

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PA0 | 26 | - | U0Rx | - | - | - | - | - | - | I2C1SCL | U1Rx | - | - |
| PA1 | 27 | - | U0Tx | - | - | - | - | - | - | I2C1SDA | U1Tx | - | - |
| PA2 | 28 | - | SSI0Clk | - | - | - | - | - | - | - | - | - | - |
| PA3 | 29 | - | SSI0Fss | - | - | - | - | - | - | - | - | - | - |
| PA4 | 30 | - | SSI0Rx | - | - | - | - | - | - | - | - | - | - |
| PA5 | 31 | - | SSI0Tx | - | - | - | - | - | - | - | - | - | - |
| PA6 | 34 | - | I2C1SCL | CCP1 | - | - | - | - | - | - | - | - | - |
| PA7 | 35 | - | I2C1SDA | CCP4 | - | - | - | - | CCP3 | - | - | - | - |
| PB0 | 66 | - | CCP0 | - | - | - | U1Rx | - | - | - | - | - | - |
| PB1 | 67 | - | CCP2 | - | - | CCP1 | U1Tx | - | - | - | - | - | - |
| PB2 | 70 | - | I2C0SCL | - | - | CCP3 | CCP0 | - | - | - | - | - | - |
| PB3 | 71 | - | I2C0SDA | - | - | - | - | - | - | - | - | - | - |
| PB4 | 92 | AIN10 | - | - | - | U2Rx | - | - | U1Rx | - | - | - | - |
| PB5 | 91 | AIN11 | - | CCP5 | CCP6 | CCP0 | - | CCP2 | U1Tx | - | - | - | - |
| PB6 | 90 | VREFA | CCP1 | CCP7 | - | - | - | CCP5 | - | - | - | - | - |
| PB7 | 89 | - | - | - | - | NMI | - | - | - | - | - | - | - |
| PC0 | 80 | - | - | - | TCK SWCLK | - | - | - | - | - | - | - | - |
| PC1 | 79 | - | - | - | TMS SWDIO | - | - | - | - | - | - | - | - |
| PC2 | 78 | - | - | - | TDI | - | - | - | - | - | - | - | - |
| PC3 | 77 | - | - | - | TDO SWO | - | - | - | - | - | - | - | - |
| PC4 | 25 | - | CCP5 | - | - | - | CCP2 | CCP4 | - | - | CCP1 | - | - |

**Table 9-2. GPIO Pins and Alternate Functions (100LQFP) *(continued)***

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PC5 | 24 | - | CCP1 | - | - | - | CCP3 | - | - | - | - | - | - |
| PC6 | 23 | - | CCP3 | - | - | - | U1Rx | CCP0 | - | - | - | - | - |
| PC7 | 22 | - | CCP4 | - | - | CCP0 | U1Tx | - | - | - | - | - | - |
| PD0 | 10 | AIN15 | - | - | - | U2Rx | U1Rx | CCP6 | - | - | - | - | - |
| PD1 | 11 | AIN14 | - | - | - | U2Tx | U1Tx | CCP7 | - | - | - | CCP2 | - |
| PD2 | 12 | AIN13 | U1Rx | CCP6 | - | CCP5 | - | - | - | - | - | - | - |
| PD3 | 13 | AIN12 | U1Tx | CCP7 | - | CCP0 | - | - | - | - | - | - | - |
| PD4 | 95 | AIN7 | CCP0 | CCP3 | - | - | - | - | - | - | - | - | - |
| PD5 | 96 | AIN6 | CCP2 | CCP4 | - | - | - | - | - | - | U2Rx | - | - |
| PD6 | 99 | AIN5 | - | - | - | - | - | - | - | - | U2Tx | - | - |
| PD7 | 100 | AIN4 | - | - | CCP1 | - | - | - | - | - | - | - | - |
| PE0 | 72 | - | - | SSI1Clk | CCP3 | - | - | - | - | - | - | - | - |
| PE1 | 73 | - | - | SSI1Fss | - | CCP2 | CCP6 | - | - | - | - | - | - |
| PE2 | 74 | AIN9 | CCP4 | SSI1Rx | - | - | CCP2 | - | - | - | - | - | - |
| PE3 | 75 | AIN8 | CCP1 | SSI1Tx | - | - | CCP7 | - | - | - | - | - | - |
| PE4 | 6 | AIN3 | CCP3 | - | - | - | U2Tx | CCP2 | - | - | - | - | - |
| PE5 | 5 | AIN2 | CCP5 | - | - | - | - | - | - | - | - | - | - |
| PE6 | 2 | AIN1 | - | - | - | - | - | - | - | - | - | - | - |
| PE7 | 1 | AIN0 | - | - | - | - | - | - | - | - | - | - | - |
| PF0 | 47 | - | - | - | - | - | - | - | - | - | U1DSR | - | - |
| PF1 | 61 | - | - | - | - | - | - | - | - | - | - | CCP3 | - |
| PF2 | 60 | - | - | - | - | - | - | - | - | - | SSI1Clk | - | - |
| PF3 | 59 | - | - | - | - | - | - | - | - | - | SSI1Fss | - | - |
| PF4 | 58 | - | CCP0 | - | - | - | - | - | - | - | SSI1Rx | - | - |
| PF5 | 46 | - | CCP2 | - | - | - | - | - | - | - | SSI1Tx | - | - |
| PF6 | 43 | - | CCP1 | - | - | - | - | - | - | - | - | - | - |
| PF7 | 42 | - | CCP4 | - | - | - | - | - | - | - | - | - | - |
| PG0 | 19 | - | U2Rx | - | I2C1SCL | - | - | - | - | - | - | - | - |
| PG1 | 18 | - | U2Tx | - | I2C1SDA | - | - | - | - | - | - | - | - |
| PG2 | 17 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG3 | 16 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG4 | 41 | - | CCP3 | - | - | - | - | - | - | - | - | - | - |
| PG5 | 40 | - | CCP5 | - | - | - | - | - | - | - | - | - | - |
| PG6 | 37 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG7 | 36 | - | - | - | - | - | - | - | - | CCP5 | - | - | - |
| PH0 | 86 | - | CCP6 | - | - | - | - | - | - | - | - | - | - |
| PH1 | 85 | - | CCP7 | - | - | - | - | - | - | - | - | - | - |
| PH2 | 84 | - | - | - | - | - | - | - | - | - | - | - | - |
| PH3 | 83 | - | - | - | - | - | - | - | - | - | - | - | - |

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

### Table 9-3. GPIO Pins and Alternate Functions (108BGA)

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PA0 | L3 | - | U0Rx | - | - | - | - | - | - | I2C1SCL | U1Rx | - | - |
| PA1 | M3 | - | U0Tx | - | - | - | - | - | - | I2C1SDA | U1Tx | - | - |
| PA2 | M4 | - | SSI0Clk | - | - | - | - | - | - | - | - | - | - |
| PA3 | L4 | - | SSI0Fss | - | - | - | - | - | - | - | - | - | - |
| PA4 | L5 | - | SSI0Rx | - | - | - | - | - | - | - | - | - | - |
| PA5 | M5 | - | SSI0Tx | - | - | - | - | - | - | - | - | - | - |
| PA6 | L6 | - | I2C1SCL | CCP1 | - | - | - | - | - | - | - | - | - |
| PA7 | M6 | - | I2C1SDA | CCP4 | - | - | - | - | CCP3 | - | - | - | - |
| PB0 | E12 | - | CCP0 | - | - | - | U1Rx | - | - | - | - | - | - |
| PB1 | D12 | - | CCP2 | - | - | CCP1 | U1Tx | - | - | - | - | - | - |
| PB2 | C11 | - | I2C0SCL | - | - | CCP3 | CCP0 | - | - | - | - | - | - |
| PB3 | C12 | - | I2C0SDA | - | - | - | - | - | - | - | - | - | - |
| PB4 | A6 | AIN10 | - | - | - | U2Rx | - | - | U1Rx | - | - | - | - |
| PB5 | B7 | AIN11 | - | CCP5 | CCP6 | CCP0 | - | CCP2 | U1Tx | - | - | - | - |
| PB6 | A7 | VREFA | CCP1 | CCP7 | - | - | - | CCP5 | - | - | - | - | - |
| PB7 | A8 | - | - | - | - | NMI | - | - | - | - | - | - | - |
| PC0 | A9 | - | - | - | TCK SWCLK | - | - | - | - | - | - | - | - |
| PC1 | B9 | - | - | - | TMS SWDIO | - | - | - | - | - | - | - | - |
| PC2 | B8 | - | - | - | TDI | - | - | - | - | - | - | - | - |
| PC3 | A10 | - | - | - | TDO SWO | - | - | - | - | - | - | - | - |
| PC4 | L1 | - | CCP5 | - | - | - | CCP2 | CCP4 | - | - | CCP1 | - | - |
| PC5 | M1 | - | CCP1 | - | - | - | CCP3 | - | - | - | - | - | - |
| PC6 | M2 | - | CCP3 | - | - | - | U1Rx | CCP0 | - | - | - | - | - |
| PC7 | L2 | - | CCP4 | - | - | CCP0 | U1Tx | - | - | - | - | - | - |
| PD0 | G1 | AIN15 | - | - | - | U2Rx | U1Rx | CCP6 | - | - | - | - | - |
| PD1 | G2 | AIN14 | - | - | - | U2Tx | U1Tx | CCP7 | - | - | - | CCP2 | - |
| PD2 | H2 | AIN13 | U1Rx | CCP6 | - | CCP5 | - | - | - | - | - | - | - |
| PD3 | H1 | AIN12 | U1Tx | CCP7 | - | CCP0 | - | - | - | - | - | - | - |
| PD4 | A4 | AIN7 | CCP0 | CCP3 | - | - | - | - | - | - | - | - | - |
| PD5 | B4 | AIN6 | CCP2 | CCP4 | - | - | - | - | - | - | U2Rx | - | - |
| PD6 | A3 | AIN5 | - | - | - | - | - | - | - | - | U2Tx | - | - |
| PD7 | A2 | AIN4 | - | - | CCP1 | - | - | - | - | - | - | - | - |
| PE0 | A11 | - | - | SSI1Clk | CCP3 | - | - | - | - | - | - | - | - |
| PE1 | B12 | - | - | SSI1Fss | - | CCP2 | CCP6 | - | - | - | - | - | - |
| PE2 | B11 | AIN9 | CCP4 | SSI1Rx | - | - | CCP2 | - | - | - | - | - | - |
| PE3 | A12 | AIN8 | CCP1 | SSI1Tx | - | - | CCP7 | - | - | - | - | - | - |
| PE4 | B2 | AIN3 | CCP3 | - | - | - | U2Tx | CCP2 | - | - | - | - | - |
| PE5 | B3 | AIN2 | CCP5 | - | - | - | - | - | - | - | - | - | - |

**Table 9-3. GPIO Pins and Alternate Functions (108BGA) *(continued)***

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PE6 | A1 | AIN1 | - | - | - | - | - | - | - | - | - | - | - |
| PE7 | B1 | AIN0 | - | - | - | - | - | - | - | - | - | - | - |
| PF0 | M9 | - | - | - | - | - | - | - | - | - | U1DSR | - | - |
| PF1 | H12 | - | - | - | - | - | - | - | - | - | - | CCP3 | - |
| PF2 | J11 | - | - | - | - | - | - | - | - | - | SSI1Clk | - | - |
| PF3 | J12 | - | - | - | - | - | - | - | - | - | SSI1Fss | - | - |
| PF4 | L9 | - | CCP0 | - | - | - | - | - | - | - | SSI1Rx | - | - |
| PF5 | L8 | - | CCP2 | - | - | - | - | - | - | - | SSI1Tx | - | - |
| PF6 | M8 | - | CCP1 | - | - | - | - | - | - | - | - | - | - |
| PF7 | K4 | - | CCP4 | - | - | - | - | - | - | - | - | - | - |
| PG0 | K1 | - | U2Rx | - | I2C1SCL | - | - | - | - | - | - | - | - |
| PG1 | K2 | - | U2Tx | - | I2C1SDA | - | - | - | - | - | - | - | - |
| PG2 | J1 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG3 | J2 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG4 | K3 | - | CCP3 | - | - | - | - | - | - | - | - | - | - |
| PG5 | M7 | - | CCP5 | - | - | - | - | - | - | - | - | - | - |
| PG6 | L7 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG7 | C10 | - | - | - | - | - | - | - | - | CCP5 | - | - | - |
| PH0 | C9 | - | CCP6 | - | - | - | - | - | - | - | - | - | - |
| PH1 | C8 | - | CCP7 | - | - | - | - | - | - | - | - | - | - |
| PH2 | D11 | - | - | - | - | - | - | - | - | - | - | - | - |
| PH3 | D10 | - | - | - | - | - | - | - | - | - | - | - | - |

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

## 9.2 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 9-1 on page 400 and Figure 9-2 on page 401). The LM3S1G58 microcontroller contains eightports and thus eight of these physical GPIO blocks. Note that not all pins may be 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 17-5 on page 757.

**Figure 9-1. Digital I/O Pads**

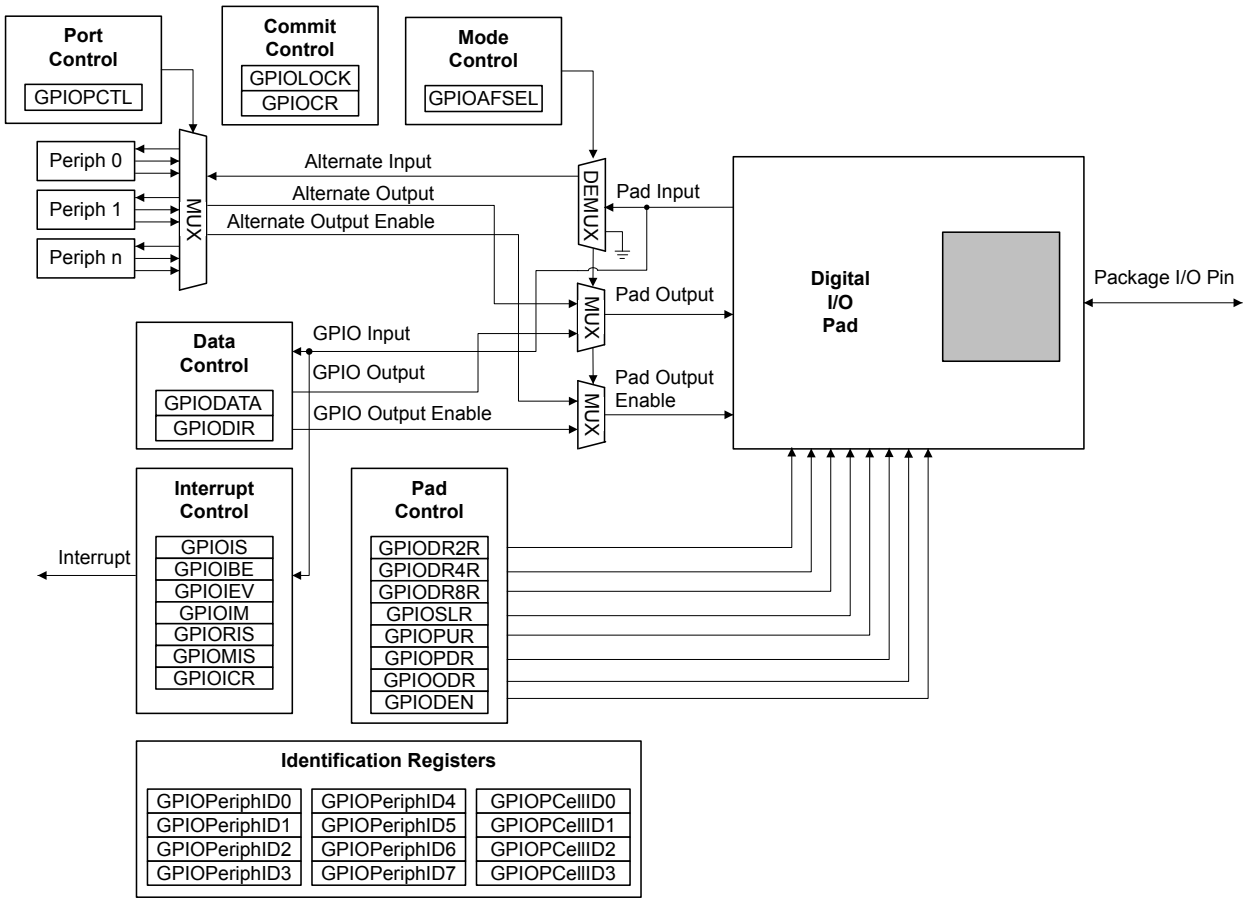
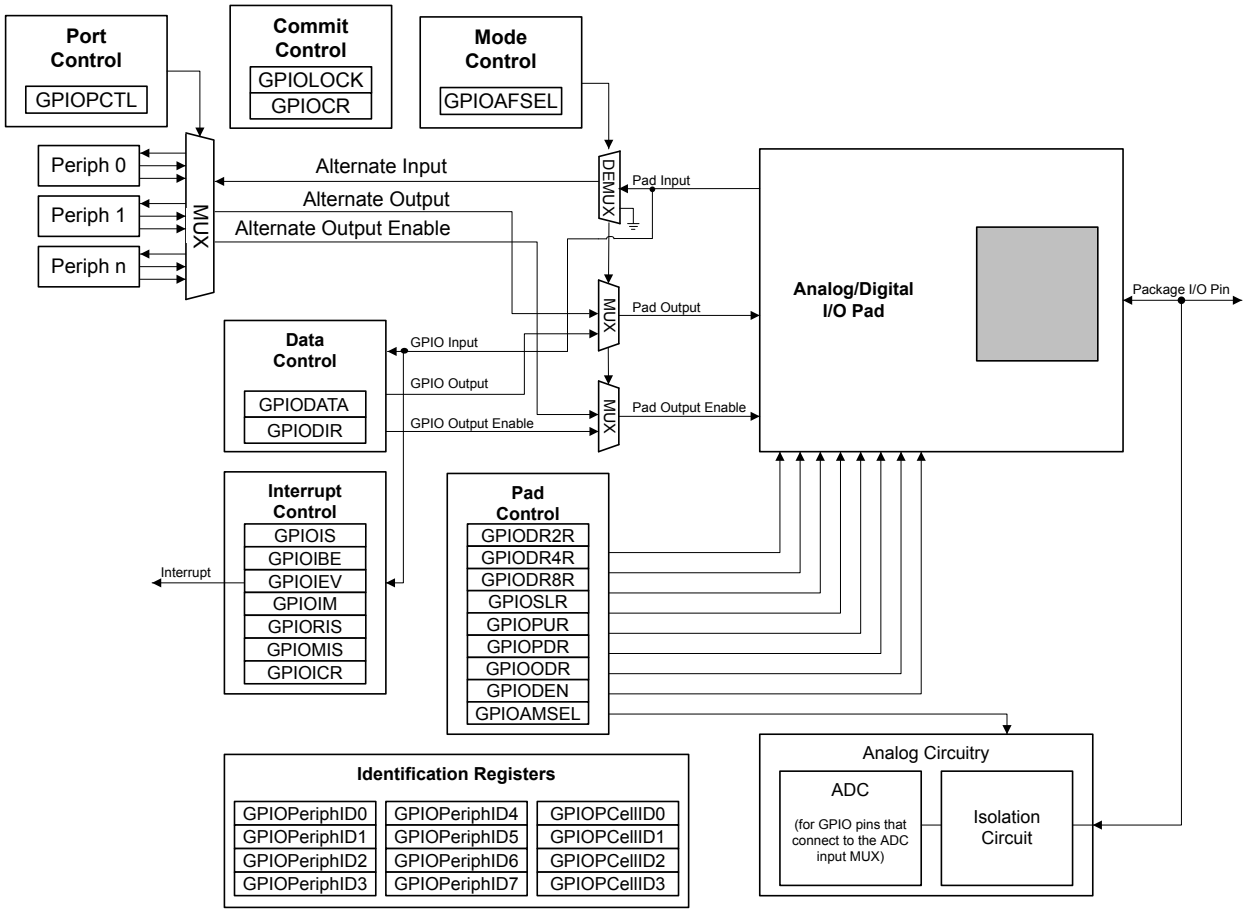


**Figure 9-2. Analog/Digital I/O Pads**




## 9.2.1 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 Stellaris® 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.**

### 9.2.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 409) 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.

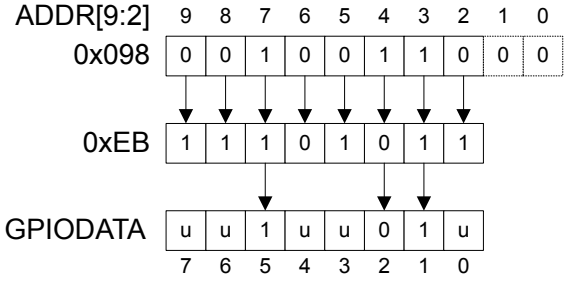
#### 9.2.1.2 Data Register Operation

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 408) 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 9-3, where u indicates that data is unchanged by the write.

**Figure 9-3. 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 9-4.

**Figure 9-4. GPIODATA Read Example**

### 9.2.2 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 410)

- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 411)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 412)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 413).

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 414 and page 415). 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.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 417).

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.

#### 9.2.2.1 ADC Trigger Source

In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set), an interrupt for Port B is generated, and an external 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 552.

If no other Port B pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the **EN0** register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See page 105 for more information.

### 9.2.3 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 418), 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 17-5 on page 757.

**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.

### 9.2.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 418), **GPIO Pull Up Select (GPIOPUR)** register (see page 424), **GPIO Pull-Down Select (GPIOPDR)** register (see page 426), and **GPIO Digital Enable (GPIODEN)** register (see

page 429) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 431) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 432) have been set.

### 9.2.5 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**, **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.

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the $V_{OL}$ value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package or BGA pin group with the total number of high-current GPIO outputs not exceeding four for the entire package.

### 9.2.6 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.

## 9.3 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris parts. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHBCTL** register (see page 203).

To use the pins in a particular GPIO port, the clock for the port must be enabled by setting the appropriate GPIO Port bit field (`GPIOn`) in the **RCGC2** register (see page 251).

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**=0, except for the pins shown in Table 9-1 on page 396. Table 9-4 on page 404 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 9-5 on page 405 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

**Table 9-4. GPIO Pad Configuration Examples**

| Configuration | GPIO Register Bit Value[a] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AFSEL | DIR | ODR | DEN | PUR | PDR | DR2R | DR4R | DR8R | SLR |
| Digital Input (GPIO) | 0 | 0 | 0 | 1 | ? | ? | 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 ($I^2C$) | 1 | X | 1 | 1 | X | X | ? | ? | ? | ? |
| Digital Input (Timer CCP) | 1 | X | 0 | 1 | ? | ? | X | X | X | X |

**Table 9-4. GPIO Pad Configuration Examples *(continued)***

| Configuration | GPIO Register Bit Value[a] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | AFSEL | DIR | ODR | DEN | PUR | PDR | DR2R | DR4R | DR8R | SLR |
| Digital Output (Timer PWM) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? |
| Digital Input/Output (SSI) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? |
| Digital Input/Output (UART) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? |

a. X=Ignored (don't care bit)

?=Can be either 0 or 1, depending on the configuration

**Table 9-5. 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)

## 9.4 Register Map

Table 9-7 on page 406 lists the GPIO registers. Each GPIO port can be accessed through one of two bus apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris parts. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus.

**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.

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000

- GPIO Port E (APB): 0x4002.4000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (APB): 0x4002.5000
- GPIO Port F (AHB): 0x4005.D000
- GPIO Port G (APB): 0x4002.6000
- GPIO Port G (AHB): 0x4005.E000
- GPIO Port H (APB): 0x4002.7000
- GPIO Port H (AHB): 0x4005.F000

Note that each GPIO module clock must be enabled before the registers can be programmed (see page 251). There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

**Table 9-6. GPIO Pins With Non-Zero Reset Values**

| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL |
|---|---|---|---|---|---|---|
| PA[1:0] | UART0 | 0 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 1 | 0 | 0 | 0x3 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 |

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 (PB7 and PC[3:0]). These five pins are the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W.

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as GPIO pins, the PC[3:0] pins default to non-committable. Similarly, to ensure that the NMI pin is not accidentally programmed as a GPIO pin, the PB7 pin defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of **GPIOCR** for Port C is 0x0000.00F0.

**Table 9-7. GPIO Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | GPIODATA | R/W | 0x0000.0000 | GPIO Data | 408 |
| 0x400 | GPIODIR | R/W | 0x0000.0000 | GPIO Direction | 409 |
| 0x404 | GPIOIS | R/W | 0x0000.0000 | GPIO Interrupt Sense | 410 |
| 0x408 | GPIOIBE | R/W | 0x0000.0000 | GPIO Interrupt Both Edges | 411 |
| 0x40C | GPIOIEV | R/W | 0x0000.0000 | GPIO Interrupt Event | 412 |
| 0x410 | GPIOIM | R/W | 0x0000.0000 | GPIO Interrupt Mask | 413 |
| 0x414 | GPIORIS | RO | 0x0000.0000 | GPIO Raw Interrupt Status | 414 |

**Table 9-7. GPIO Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x418 | GPIOMIS | RO | 0x0000.0000 | GPIO Masked Interrupt Status | 415 |
| 0x41C | GPIOICR | W1C | 0x0000.0000 | GPIO Interrupt Clear | 417 |
| 0x420 | GPIOAFSEL | R/W | - | GPIO Alternate Function Select | 418 |
| 0x500 | GPIODR2R | R/W | 0x0000.00FF | GPIO 2-mA Drive Select | 420 |
| 0x504 | GPIODR4R | R/W | 0x0000.0000 | GPIO 4-mA Drive Select | 421 |
| 0x508 | GPIODR8R | R/W | 0x0000.0000 | GPIO 8-mA Drive Select | 422 |
| 0x50C | GPIOODR | R/W | 0x0000.0000 | GPIO Open Drain Select | 423 |
| 0x510 | GPIOPUR | R/W | - | GPIO Pull-Up Select | 424 |
| 0x514 | GPIOPDR | R/W | 0x0000.0000 | GPIO Pull-Down Select | 426 |
| 0x518 | GPIOSLR | R/W | 0x0000.0000 | GPIO Slew Rate Control Select | 428 |
| 0x51C | GPIODEN | R/W | - | GPIO Digital Enable | 429 |
| 0x520 | GPIOLOCK | R/W | 0x0000.0001 | GPIO Lock | 431 |
| 0x524 | GPIOCR | - | - | GPIO Commit | 432 |
| 0x528 | GPIOAMSEL | R/W | 0x0000.0000 | GPIO Analog Mode Select | 434 |
| 0x52C | GPIOPCTL | R/W | - | GPIO Port Control | 435 |
| 0xFD0 | GPIOPeriphID4 | RO | 0x0000.0000 | GPIO Peripheral Identification 4 | 437 |
| 0xFD4 | GPIOPeriphID5 | RO | 0x0000.0000 | GPIO Peripheral Identification 5 | 438 |
| 0xFD8 | GPIOPeriphID6 | RO | 0x0000.0000 | GPIO Peripheral Identification 6 | 439 |
| 0xFDC | GPIOPeriphID7 | RO | 0x0000.0000 | GPIO Peripheral Identification 7 | 440 |
| 0xFE0 | GPIOPeriphID0 | RO | 0x0000.0061 | GPIO Peripheral Identification 0 | 441 |
| 0xFE4 | GPIOPeriphID1 | RO | 0x0000.0000 | GPIO Peripheral Identification 1 | 442 |
| 0xFE8 | GPIOPeriphID2 | RO | 0x0000.0018 | GPIO Peripheral Identification 2 | 443 |
| 0xFEC | GPIOPeriphID3 | RO | 0x0000.0001 | GPIO Peripheral Identification 3 | 444 |
| 0xFF0 | GPIOPCellID0 | RO | 0x0000.000D | GPIO PrimeCell Identification 0 | 445 |
| 0xFF4 | GPIOPCellID1 | RO | 0x0000.00F0 | GPIO PrimeCell Identification 1 | 446 |
| 0xFF8 | GPIOPCellID2 | RO | 0x0000.0005 | GPIO PrimeCell Identification 2 | 447 |
| 0xFFC | GPIOPCellID3 | RO | 0x0000.00B1 | GPIO PrimeCell Identification 3 | 448 |

## 9.5 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 409).

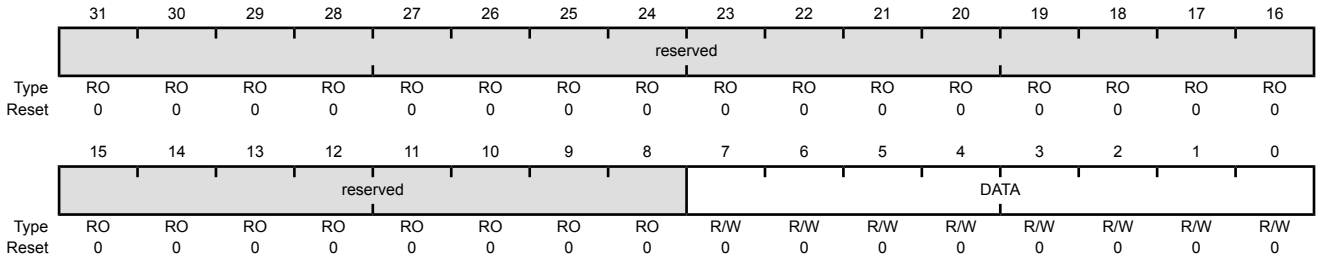
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set. Otherwise, the bit values remain unchanged by the write.

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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x000
Type R/W, 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 | R/W | 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 402 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x400
Type R/W, 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 | | | | | | | | DIR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | DIR | R/W | 0x00 | GPIO Data Direction |

| Value | Description |
|---|---|
| 0 | Corresponding pin is an input. |
| 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.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x404
Type R/W, 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 | | | | | | | | IS | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | IS | R/W | 0x00 | GPIO Interrupt Sense |

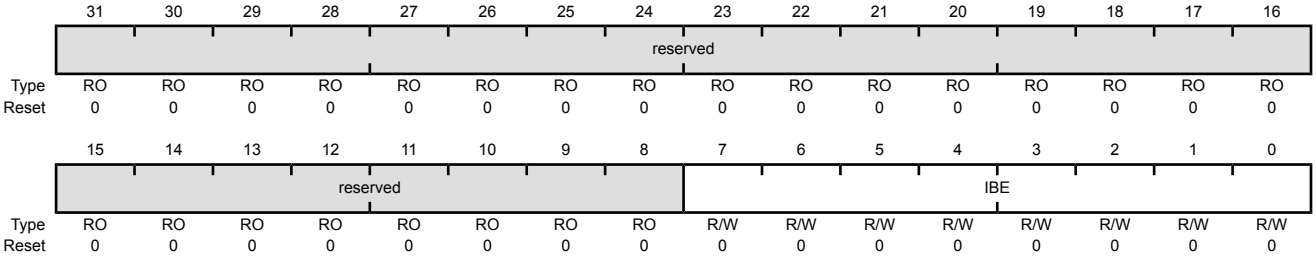
| Value | Description |
|---|---|
| 0 | The edge on the corresponding pin is detected (edge-sensitive). |
| 1 | The level on the corresponding pin is detected (level-sensitive). |

## 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 410) 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 412). Clearing a bit configures the pin to be controlled by the **GPIOIEV** register. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x408
Type R/W, 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 | IBE | R/W | 0x00 | GPIO Interrupt Both Edges |

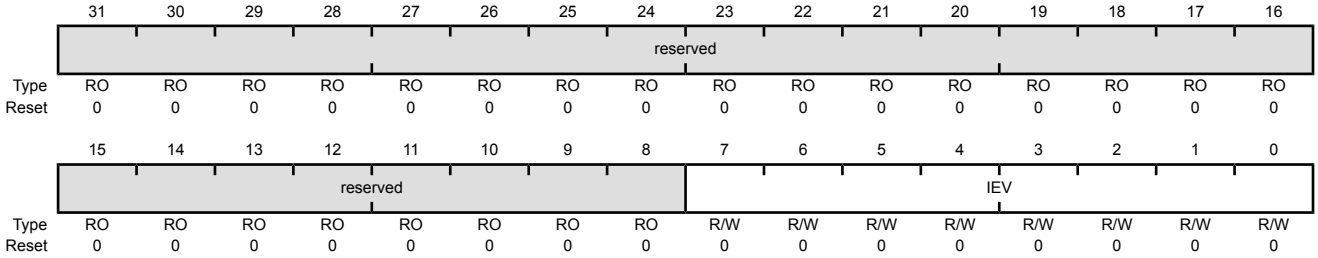
| Value | Description |
|---|---|
| 0 | Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 412). |
| 1 | Both edges on the corresponding pin trigger an interrupt. |

## 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 410). 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x40C
Type R/W, reset 0x0000.0000



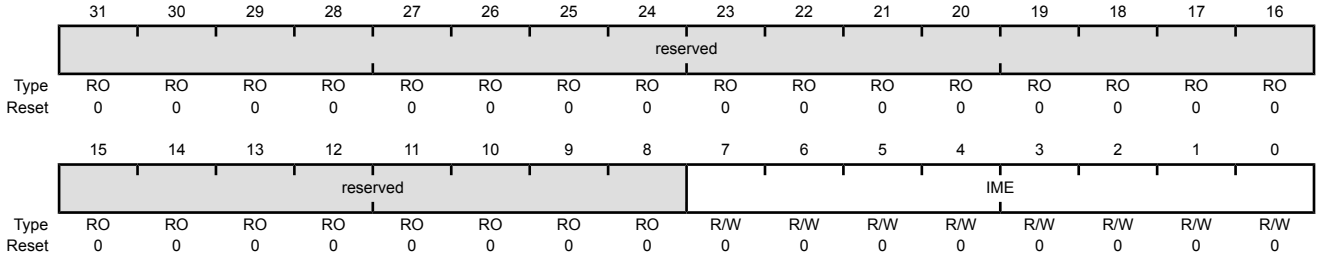

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | IEV | R/W | 0x00 | GPIO Interrupt Event<br>Value Description<br>0 A falling edge or a Low level on the corresponding pin triggers an interrupt.<br>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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x410
Type R/W, 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 | IME | R/W | 0x00 | GPIO Interrupt Mask Enable |

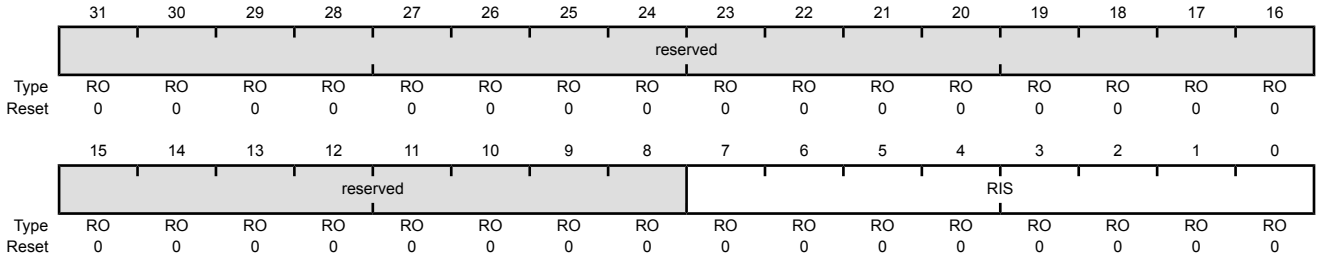
| Value | Description |
|---|---|
| 0 | The interrupt from the corresponding pin is masked. |
| 1 | The interrupt from the corresponding pin is sent to the interrupt controller. |

## 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. If the corresponding bit in the **GPIO Interrupt Mask (GPIOIM)** register (see page 413) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. A bit in this register can be cleared by writing a 1 to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x414
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. |
| 7:0 | RIS | RO | 0x00 | GPIO Interrupt Raw Status<br><br>Value Description<br>1 An interrupt condition has occurred on the corresponding pin.<br>0 An interrupt condition has not occurred on the corresponding pin.<br><br>A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register. |

## 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.

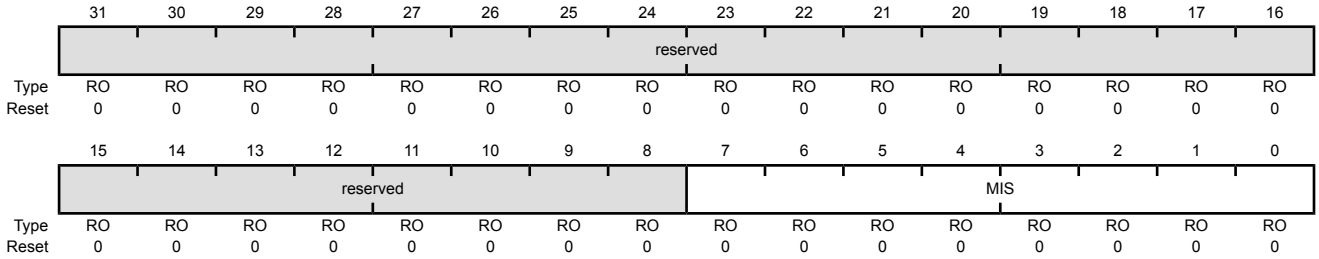
In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set), an interrupt for Port B is generated, and an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 552.

If no other Port B pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the **EN0** register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See page 105 for more information.

**GPIOMIS** is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x418
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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | MIS | RO | 0x00 | GPIO Masked Interrupt Status |

| Value | Description |
|---|---|
| 1 | An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller. |
| 0 | An interrupt condition on the corresponding pin is masked or has not occurred. |

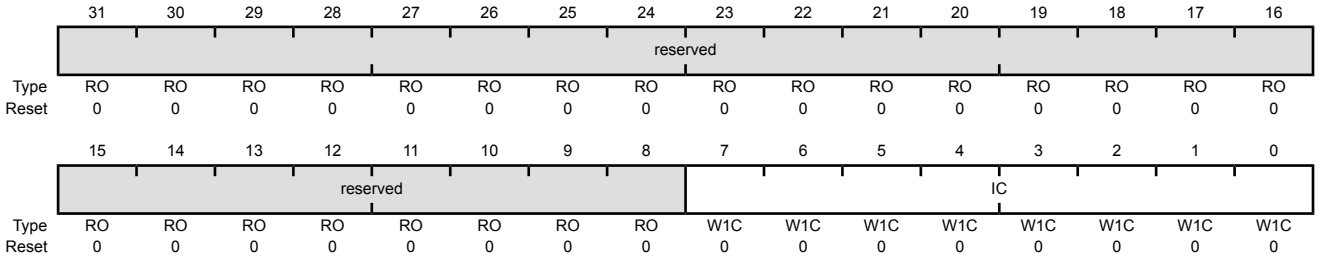
A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

## Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C

The **GPIOICR** register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt bit in the **GPIORIS** and **GPIOMIS** registers. Writing a 0 has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x41C
Type W1C, 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 | IC | W1C | 0x00 | GPIO Interrupt Clear<br>Value Description<br>1 The corresponding interrupt is cleared.<br>0 The corresponding interrupt is unaffected. |

## Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420

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 17-5 on page 757 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:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

**Table 9-8. GPIO Pins With Non-Zero Reset Values**

| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL |
|---|---|---|---|---|---|---|
| PA[1:0] | UART0 | 0 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 1 | 0 | 0 | 0x3 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 |

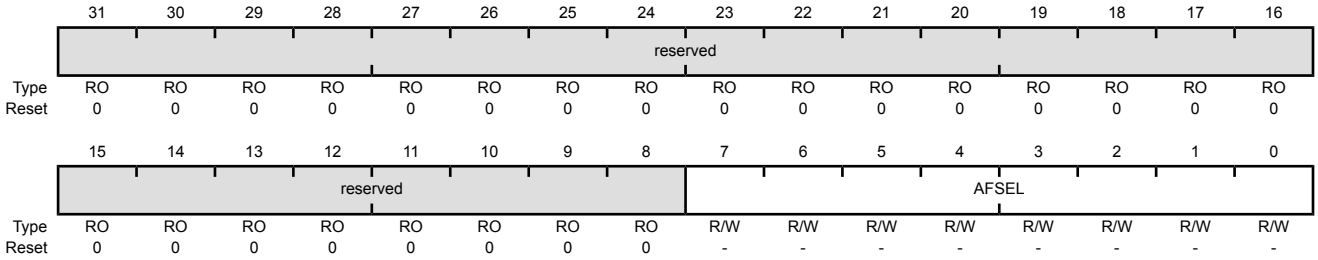
**Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller 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.**

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 418), **GPIO Pull Up Select (GPIOPUR)** register (see page 424), **GPIO Pull-Down Select (GPIOPDR)** register (see page 426), and **GPIO Digital Enable (GPIODEN)** register (see page 429) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 431) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 432) 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 404).

## GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x420
Type R/W, 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 | R/W | - | 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 9-1 on page 396.

## 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.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x500
Type R/W, 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 | | | | | | | | DRV2 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | 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 | R/W | 0xFF | Output Pad 2-mA Drive Enable |

| Value | Description |
|---|---|
| 1 | The corresponding GPIO pin has 2-mA drive. |
| 0 | The drive for the corresponding GPIO pin is controlled by the **GPIODR4R** or **GPIODR8R** register. |

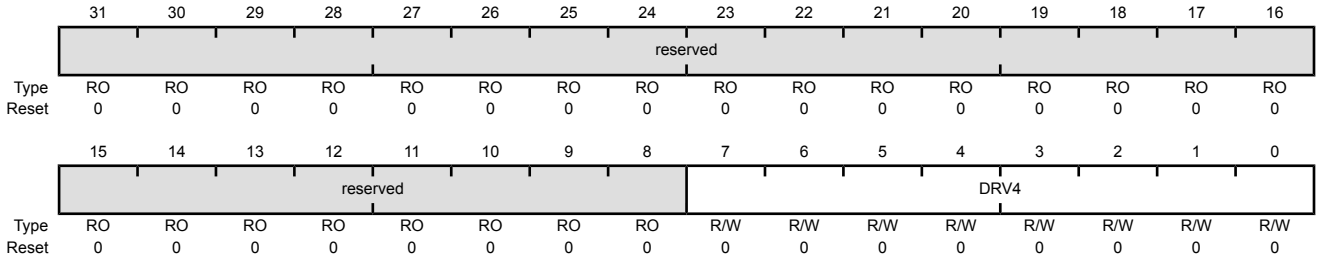
Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

## 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.

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x504
Type R/W, 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 | R/W | 0x00 | Output Pad 4-mA Drive Enable<br><br>Value Description<br>1 The corresponding GPIO pin has 4-mA drive.<br>0 The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR8R** register.<br><br>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 second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle. |

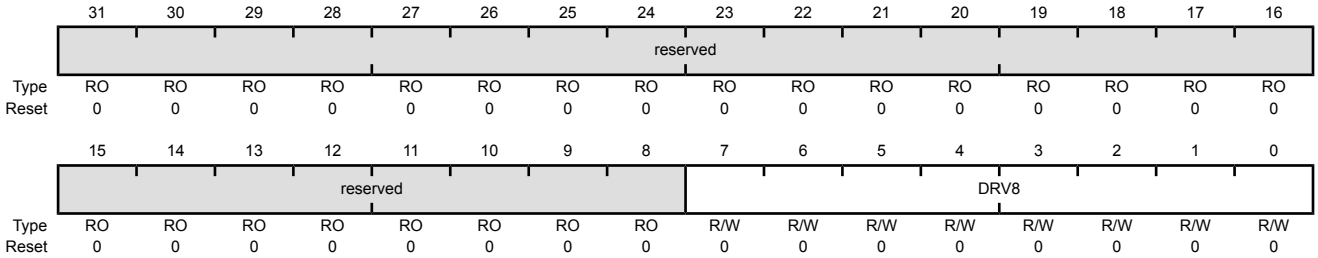
## Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

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 785 for further information.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x508
Type R/W, 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 | R/W | 0x00 | Output Pad 8-mA Drive Enable<br><br>Value Description<br>1 The corresponding GPIO pin has 8-mA drive.<br>0 The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR4R** register.<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 second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle. |

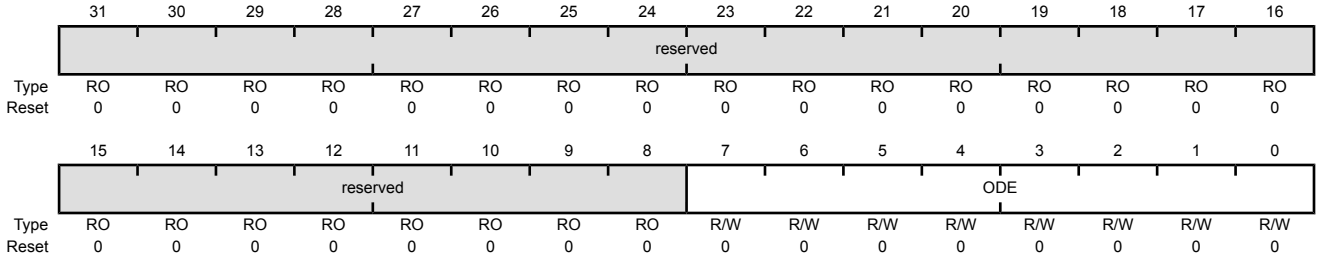
## Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C

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 429). Corresponding bits in the drive strength and slew rate control registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and 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 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 404).

GPIO Open Drain Select (GPIOODR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x50C
Type R/W, 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 | R/W | 0x00 | Output Pad Open Drain Enable<br>Value Description<br>1 The corresponding pin is configured as open drain.<br>0 The corresponding pin is not 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 426). 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:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

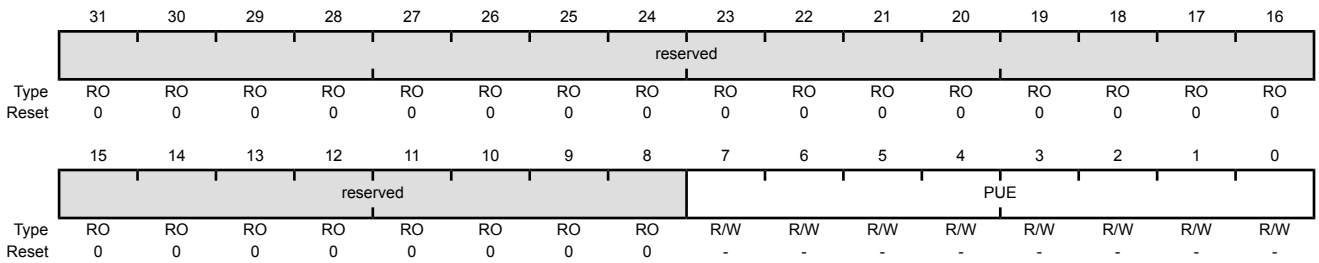
**Table 9-9. GPIO Pins With Non-Zero Reset Values**

| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL |
|---|---|---|---|---|---|---|
| PA[1:0] | UART0 | 0 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 1 | 0 | 0 | 0x3 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 |

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 418), **GPIO Pull Up Select (GPIOPUR)** register (see page 424), **GPIO Pull-Down Select (GPIOPDR)** register (see page 426), and **GPIO Digital Enable (GPIODEN)** register (see page 429) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 431) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 432) have been set.

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x510
Type R/W, 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 | R/W | - | Pad Weak Pull-Up Enable<br><br>Value Description<br>1 The corresponding pin has a weak pull-up resistor.<br>0 The corresponding pin is not affected.<br><br>Setting a bit in the **GPIOPDR** register clears the corresponding bit in the **GPIOPUR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.<br><br>The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 396. |

## 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 424).

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

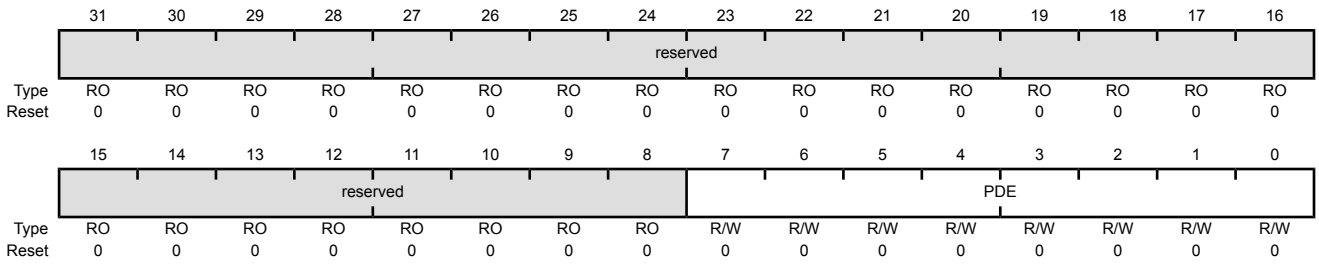
**Table 9-10. GPIO Pins With Non-Zero Reset Values**

| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL |
|---|---|---|---|---|---|---|
| PA[1:0] | UART0 | 0 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 1 | 0 | 0 | 0x3 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 |

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 418), **GPIO Pull Up Select (GPIOPUR)** register (see page 424), **GPIO Pull-Down Select (GPIOPDR)** register (see page 426), and **GPIO Digital Enable (GPIODEN)** register (see page 429) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 431) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 432) have been set.

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x514
Type R/W, reset 0x0000.0000



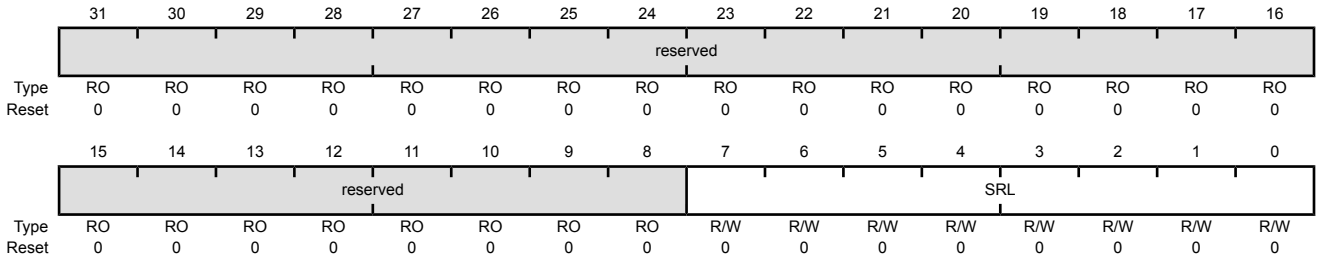
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PDE | R/W | 0x00 | Pad Weak Pull-Down Enable<br><br>Value Description<br>1 The corresponding pin has a weak pull-down resistor.<br>0 The corresponding pin is not affected.<br><br>Setting a bit in the **GPIOPUR** register clears the corresponding bit in the **GPIOPDR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, 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 drive strength option via the **GPIO 8-mA Drive Select (GPIODR8R)** register (see page 422).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x518
Type R/W, 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 | R/W | 0x00 | Slew Rate Limit Enable (8-mA drive only)<br>Value Description<br>1 Slew rate control is enabled for the corresponding pin.<br>0 Slew rate control is disabled 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:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

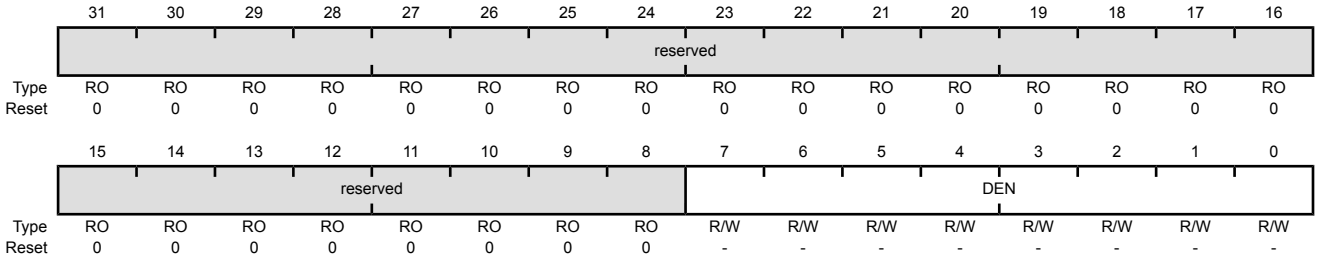
**Table 9-11. GPIO Pins With Non-Zero Reset Values**

| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL |
|---|---|---|---|---|---|---|
| PA[1:0] | UART0 | 0 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 1 | 0 | 0 | 0x3 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 |

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 418), **GPIO Pull Up Select (GPIOPUR)** register (see page 424), **GPIO Pull-Down Select (GPIOPDR)** register (see page 426), and **GPIO Digital Enable (GPIODEN)** register (see page 429) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 431) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 432) have been set.

## GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x51C
Type R/W, 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 | R/W | - | Digital Enable |

| Value | Description |
|---|---|
| 0 | The digital functions for the corresponding pin are disabled. |
| 1 | The digital functions for the corresponding pin are enabled. |

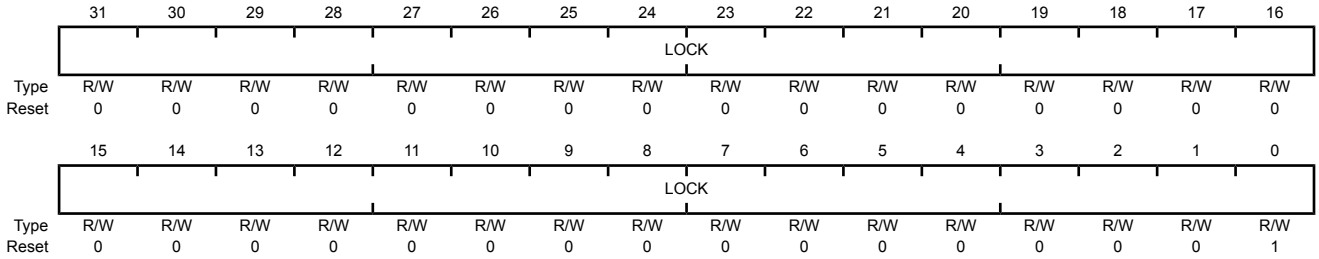
The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 396.

## Register 19: GPIO Lock (GPIOLOCK), offset 0x520

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 432). 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x520
Type R/W, reset 0x0000.0001




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | LOCK | R/W | 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.

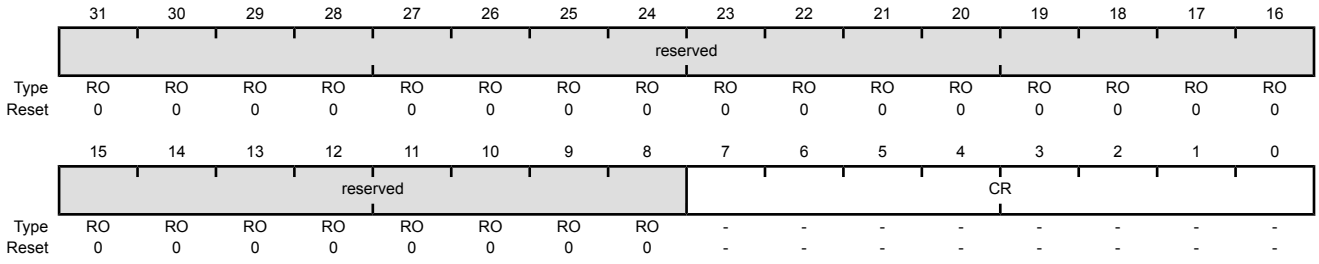
The contents of the **GPIOCR** register can only be modified if the status in the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the status in the **GPIOLOCK** register is locked.

**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 PB7 and PC[3:0], 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 on PB7 and PC[3:0], 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x524
Type -, 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:0 | CR | - | - | GPIO Commit |

| Value | Description |
|---|---|
| 1 | The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits can be written. |
| 0 | The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits cannot be written. |

**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 (PB7 and PC[3:0]). These five pins are the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W.

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as GPIO pins, the PC[3:0] pins default to non-committable. Similarly, to ensure that the NMI pin is not accidentally programmed as a GPIO pin, the PB7 pin defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of **GPIOCR** for Port C is 0x0000.00F0.

## Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

**Important:** This register is only valid for ports D and E; the corresponding base addresses for the remaining ports are not valid.

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 5-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 17-5 on page 757.

GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x528
Type R/W, 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 | | | | | | | | GPIOAMSEL | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | GPIOAMSEL | R/W | 0x00 | GPIO Analog Mode Select |

| Value | Description |
|---|---|
| 1 | The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions. |
| 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. |

**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 17-5 on page 757. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{POR}$) or asserting $\overline{RST}$ puts the pins back to their default state.

**Table 9-12. GPIO Pins With Non-Zero Reset Values**

| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOPCTL |
|---|---|---|---|---|---|---|
| PA[1:0] | UART0 | 0 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 1 | 0 | 0 | 0x3 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x1 |

GPIO Port Control (GPIOPCTL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0x52C
Type R/W, reset -

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PMC7 | | | | PMC6 | | | | PMC5 | | | | PMC4 | | | |
| Type R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PMC3 | | | | PMC2 | | | | PMC1 | | | | PMC0 | | | |
| Type R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | PMC7 | R/W | - | Port Mux Control 7<br>This field controls the configuration for GPIO pin 7. |

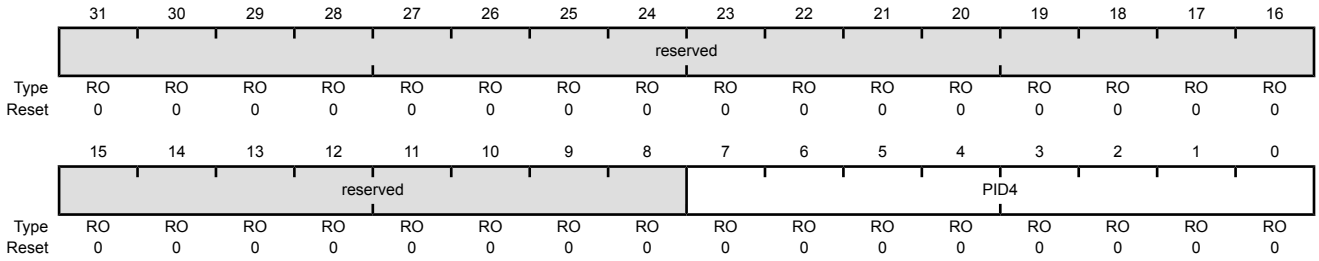
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 27:24 | PMC6 | R/W | - | Port Mux Control 6<br>This field controls the configuration for GPIO pin 6. |
| 23:20 | PMC5 | R/W | - | Port Mux Control 5<br>This field controls the configuration for GPIO pin 5. |
| 19:16 | PMC4 | R/W | - | Port Mux Control 4<br>This field controls the configuration for GPIO pin 4. |
| 15:12 | PMC3 | R/W | - | Port Mux Control 3<br>This field controls the configuration for GPIO pin 3. |
| 11:8 | PMC2 | R/W | - | Port Mux Control 2<br>This field controls the configuration for GPIO pin 2. |
| 7:4 | PMC1 | R/W | - | Port Mux Control 1<br>This field controls the configuration for GPIO pin 1. |
| 3:0 | PMC0 | R/W | - | Port Mux Control 0<br>This field controls the configuration for GPIO pin 0. |

## Register 23: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFD0
Type RO, reset 0x0000.0000



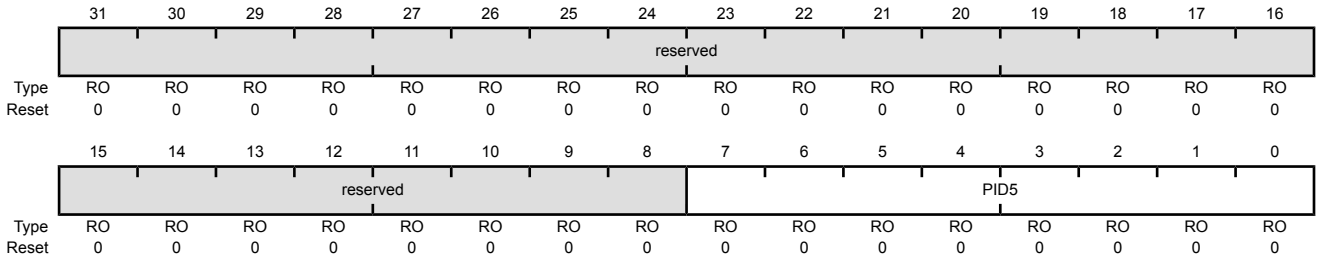

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | GPIO Peripheral ID Register [7:0] |

## Register 24: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFD4
Type RO, reset 0x0000.0000



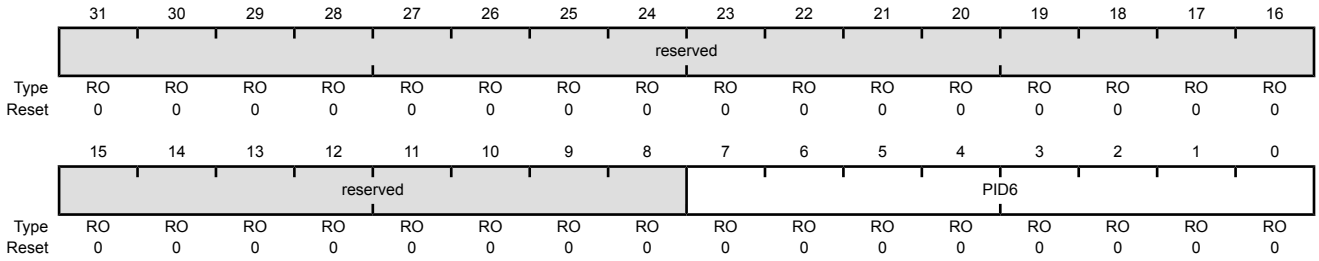

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | GPIO Peripheral ID Register [15:8] |

## Register 25: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFD8
Type RO, reset 0x0000.0000



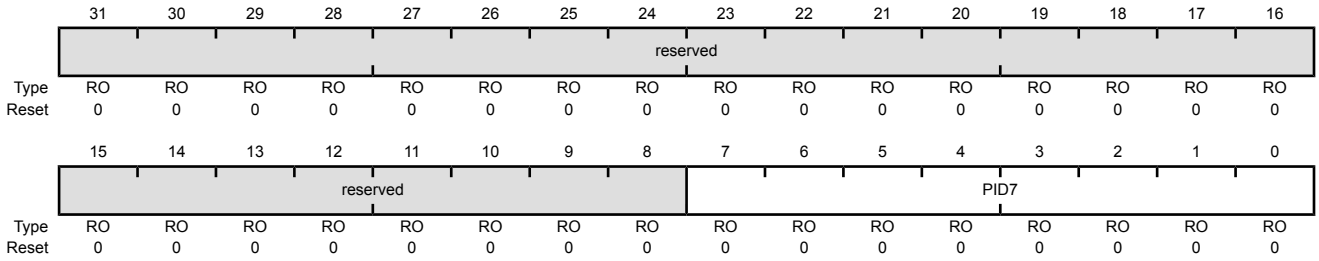

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | GPIO Peripheral ID Register [23:16] |

## Register 26: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFDC
Type RO, reset 0x0000.0000



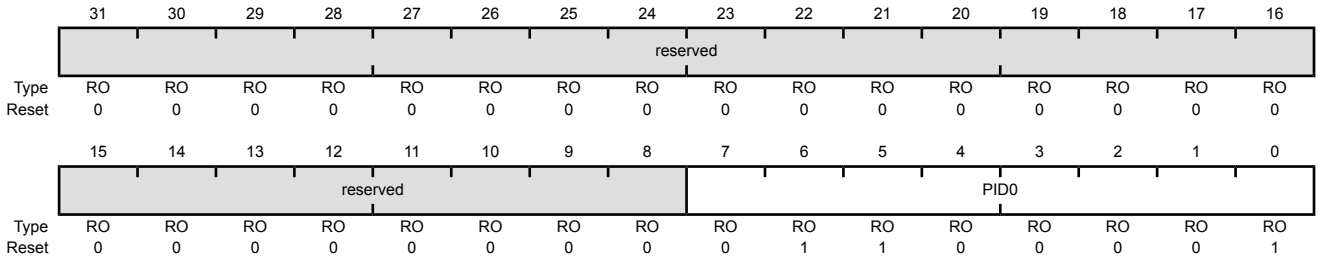

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | GPIO Peripheral ID Register [31:24] |

## Register 27: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFE0
Type RO, reset 0x0000.0061



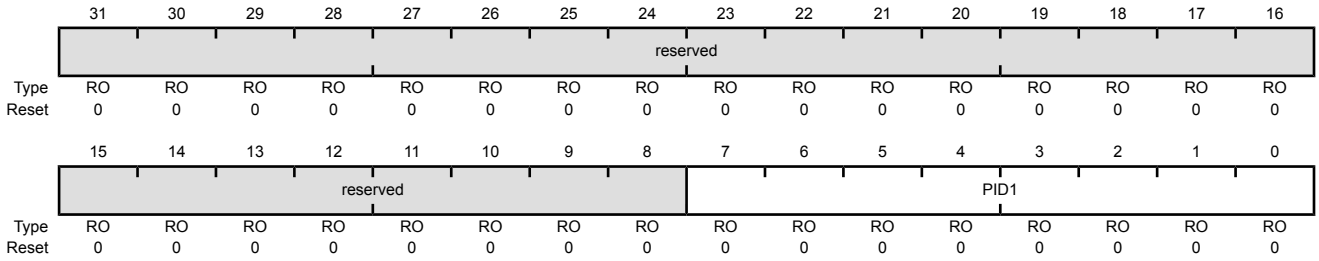

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 28: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFE4
Type RO, reset 0x0000.0000



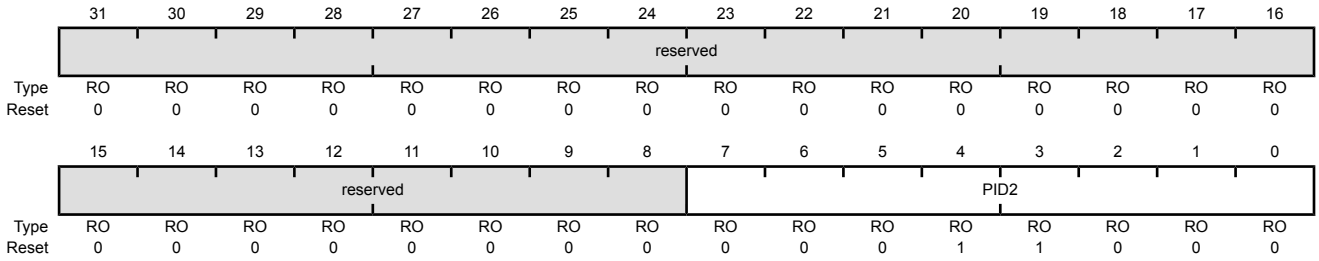

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 29: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFE8
Type RO, reset 0x0000.0018



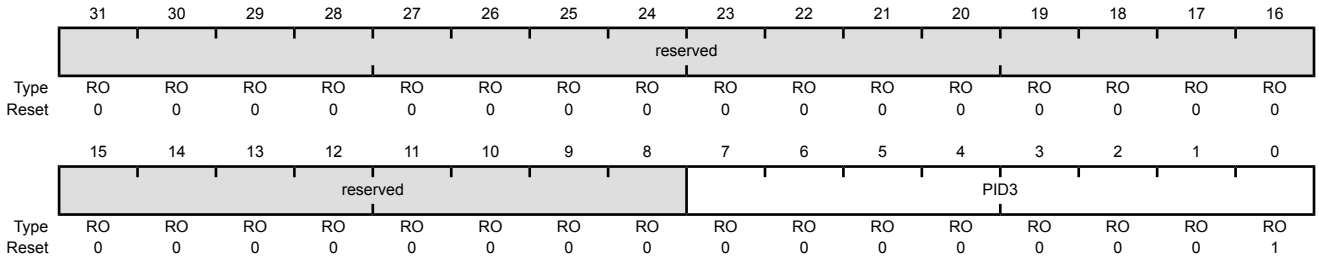

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 30: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFEC
Type RO, reset 0x0000.0001



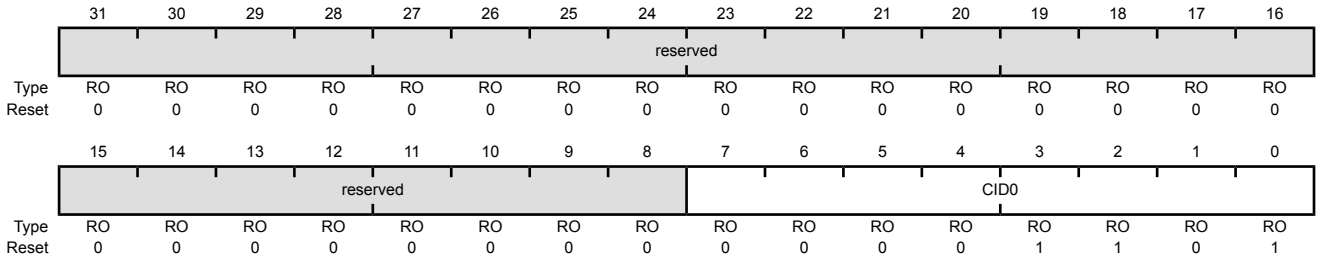

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 31: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.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 | GPIO PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 32: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFF4
Type RO, reset 0x0000.00F0

| | 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 | | | | | | | | CID1 | | | | | | | |
| 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 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

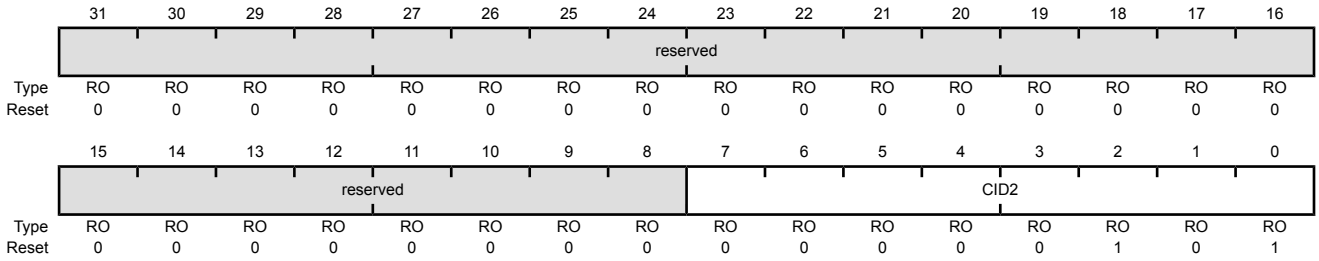
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | GPIO PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 33: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
Offset 0xFF8
Type RO, reset 0x0000.0005



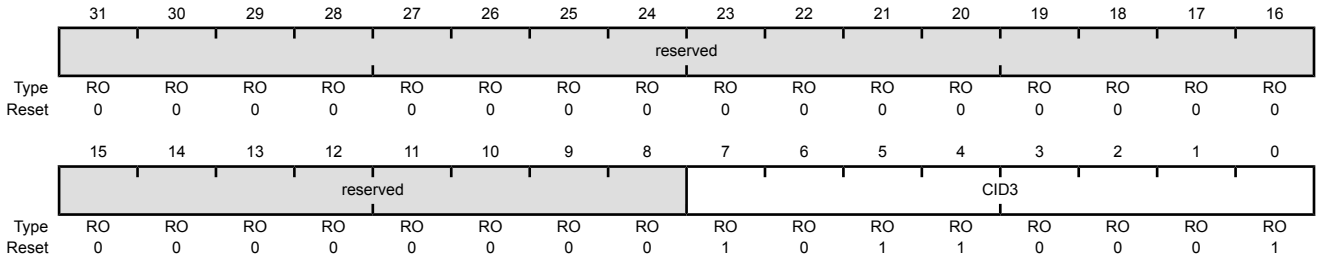

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | GPIO PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 34: 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 (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000
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. |

# 10 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The Stellaris® General-Purpose Timer Module (GPTM) contains four GPTM blocks. Each 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 Stellaris microcontrollers. Other timer resources include the System Timer (SysTick) (see 90).

The General-Purpose Timer Module (GPTM) contains four 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
  - 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Count up or down
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- 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

## 10.1 Block Diagram

In the block diagram, the specific Capture Compare PWM (CCP) pins available depend on the Stellaris device. See Table 10-1 on page 450 for the available CCP pins and their timer assignments.

**Figure 10-1. GPTM Module Block Diagram**

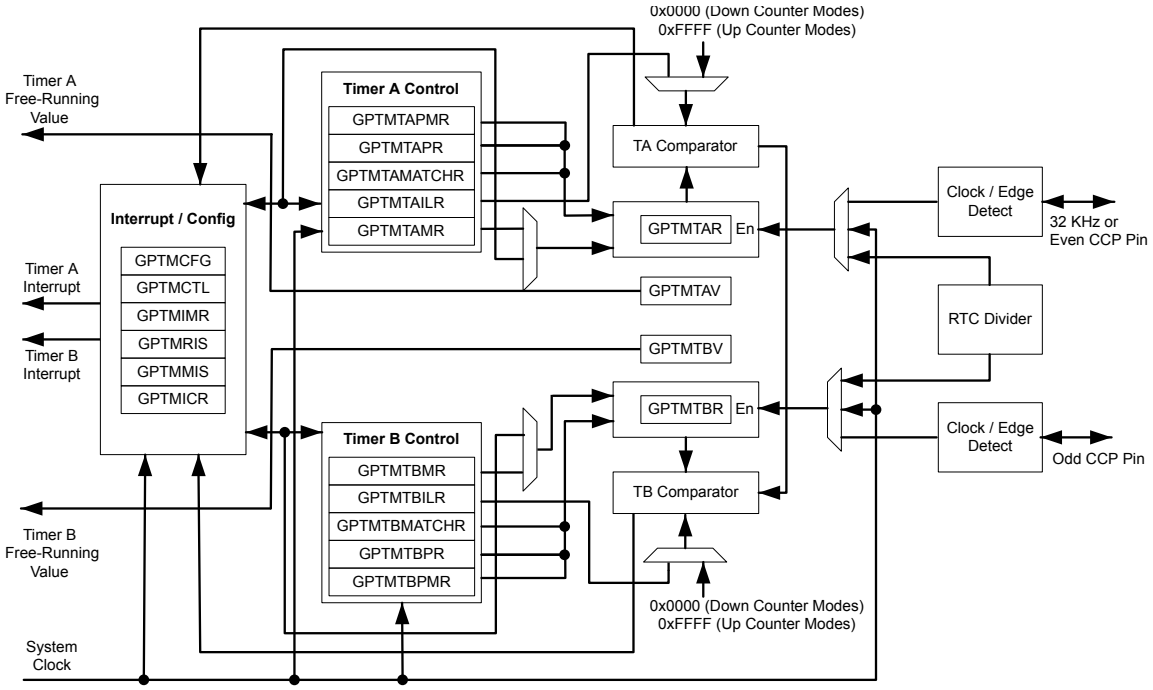



**Table 10-1. Available CCP Pins**

| Timer | 16-Bit Up/Down Counter | Even CCP Pin | Odd CCP Pin |
|---|---|---|---|
| Timer 0 | TimerA | CCP0 | - |
| | TimerB | - | CCP1 |
| Timer 1 | TimerA | CCP2 | - |
| | TimerB | - | CCP3 |
| Timer 2 | TimerA | CCP4 | - |
| | TimerB | - | CCP5 |
| Timer 3 | TimerA | CCP6 | - |
| | TimerB | - | CCP7 |

## 10.2 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 418) 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 435) 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 395.

**Table 10-2. Signals for General-Purpose Timers (100LQFP)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CCP0 | 13<br>22<br>23<br>58<br>66<br>70<br>91<br>95 | PD3 (4)<br>PC7 (4)<br>PC6 (6)<br>PF4 (1)<br>PB0 (1)<br>PB2 (5)<br>PB5 (4)<br>PD4 (1) | I/O | TTL | Capture/Compare/PWM 0. |
| CCP1 | 24<br>25<br>34<br>43<br>67<br>75<br>90<br>100 | PC5 (1)<br>PC4 (9)<br>PA6 (2)<br>PF6 (1)<br>PB1 (4)<br>PE3 (1)<br>PB6 (1)<br>PD7 (3) | I/O | TTL | Capture/Compare/PWM 1. |
| CCP2 | 6<br>11<br>25<br>46<br>67<br>73<br>74<br>91<br>96 | PE4 (6)<br>PD1 (10)<br>PC4 (5)<br>PF5 (1)<br>PB1 (1)<br>PE1 (4)<br>PE2 (5)<br>PB5 (6)<br>PD5 (1) | I/O | TTL | Capture/Compare/PWM 2. |
| CCP3 | 6<br>23<br>24<br>35<br>41<br>61<br>70<br>72<br>95 | PE4 (1)<br>PC6 (1)<br>PC5 (5)<br>PA7 (7)<br>PG4 (1)<br>PF1 (10)<br>PB2 (4)<br>PE0 (3)<br>PD4 (2) | I/O | TTL | Capture/Compare/PWM 3. |
| CCP4 | 22<br>25<br>35<br>42<br>74<br>96 | PC7 (1)<br>PC4 (6)<br>PA7 (2)<br>PF7 (1)<br>PE2 (1)<br>PD5 (2) | I/O | TTL | Capture/Compare/PWM 4. |
| CCP5 | 5<br>12<br>25<br>36<br>40<br>90<br>91 | PE5 (1)<br>PD2 (4)<br>PC4 (1)<br>PG7 (8)<br>PG5 (1)<br>PB6 (6)<br>PB5 (2) | I/O | TTL | Capture/Compare/PWM 5. |
| CCP6 | 10<br>12<br>73<br>86<br>91 | PD0 (6)<br>PD2 (2)<br>PE1 (5)<br>PH0 (1)<br>PB5 (3) | I/O | TTL | Capture/Compare/PWM 6. |

**Table 10-2. Signals for General-Purpose Timers (100LQFP) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CCP7 | 11<br>13<br>75<br>85<br>90 | PD1 (6)<br>PD3 (2)<br>PE3 (5)<br>PH1 (1)<br>PB6 (2) | I/O | TTL | Capture/Compare/PWM 7. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 10-3. Signals for General-Purpose Timers (108BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CCP0 | H1<br>L2<br>M2<br>L9<br>E12<br>C11<br>B7<br>A4 | PD3 (4)<br>PC7 (4)<br>PC6 (6)<br>PF4 (1)<br>PB0 (1)<br>PB2 (5)<br>PB5 (4)<br>PD4 (1) | I/O | TTL | Capture/Compare/PWM 0. |
| CCP1 | M1<br>L1<br>L6<br>M8<br>D12<br>A12<br>A7<br>A2 | PC5 (1)<br>PC4 (9)<br>PA6 (2)<br>PF6 (1)<br>PB1 (4)<br>PE3 (1)<br>PB6 (1)<br>PD7 (3) | I/O | TTL | Capture/Compare/PWM 1. |
| CCP2 | B2<br>G2<br>L1<br>L8<br>D12<br>B12<br>B11<br>B7<br>B4 | PE4 (6)<br>PD1 (10)<br>PC4 (5)<br>PF5 (1)<br>PB1 (1)<br>PE1 (4)<br>PE2 (5)<br>PB5 (6)<br>PD5 (1) | I/O | TTL | Capture/Compare/PWM 2. |
| CCP3 | B2<br>M2<br>M1<br>M6<br>K3<br>H12<br>C11<br>A11<br>A4 | PE4 (1)<br>PC6 (1)<br>PC5 (5)<br>PA7 (7)<br>PG4 (1)<br>PF1 (10)<br>PB2 (4)<br>PE0 (3)<br>PD4 (2) | I/O | TTL | Capture/Compare/PWM 3. |
| CCP4 | L2<br>L1<br>M6<br>K4<br>B11<br>B4 | PC7 (1)<br>PC4 (6)<br>PA7 (2)<br>PF7 (1)<br>PE2 (1)<br>PD5 (2) | I/O | TTL | Capture/Compare/PWM 4. |

**Table 10-3. Signals for General-Purpose Timers (108BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CCP5 | B3<br>H2<br>L1<br>C10<br>M7<br>A7<br>B7 | PE5 (1)<br>PD2 (4)<br>PC4 (1)<br>PG7 (8)<br>PG5 (1)<br>PB6 (6)<br>PB5 (2) | I/O | TTL | Capture/Compare/PWM 5. |
| CCP6 | G1<br>H2<br>B12<br>C9<br>B7 | PD0 (6)<br>PD2 (2)<br>PE1 (5)<br>PH0 (1)<br>PB5 (3) | I/O | TTL | Capture/Compare/PWM 6. |
| CCP7 | G2<br>H1<br>A12<br>C8<br>A7 | PD1 (6)<br>PD3 (2)<br>PE3 (5)<br>PH1 (1)<br>PB6 (2) | I/O | TTL | Capture/Compare/PWM 7. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 10.3 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 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. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range. 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 10-4 on page 453. Note that when counting down, the prescaler acts as a true prescaler and contains the least-significant bits of the count. When counting up, the prescaler acts as a timer extension and holds the most-significant bits of the count.

**Table 10-4. 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 | Down | 16-bit | 8-bit |
| Edge Time | Individual | Down | 16-bit | - |
| PWM | Individual | Down | 16-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 465), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 466), and the **GPTM Timer B Mode (GPTMTBMR)** register (see page 468). When in one of the concatentated 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.

### 10.3.1 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 load registers: the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 483) and the **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 484) and shadow registers: the **GPTM Timer A Value (GPTMTAV)** register (see page 493) and the **GPTM Timer B Value (GPTMTBV)** register (see page 494). The prescale counters are initialized to 0x00: the **GPTM Timer A Prescale (GPTMTAPR)** register (see page 487) and the **GPTM Timer B Prescale (GPTMTBPR)** register (see page 488).

### 10.3.2 Timer Modes

This section describes the operation of the various timer modes. When using Timer A and Timer B in concatenated mode, only the Timer A control and status bits must be used; there is no need to use Timer B control and status bits. The GPTM is placed into individual mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 465). 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. The prescaler is only available in the 16-bit one-shot, periodic, and input edge count timer mode. Note that when counting down, the prescaler acts as a true prescaler and contains the least-significant bits of the count. When counting up, the prescaler acts as a timer extension and holds the most-significant bits of the count. 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 Match (GPTMTnMATCH)** and the optional **GPTM Timer n Prescale Match (GPTMTnPMR)** registers.

#### 10.3.2.1 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 466). 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 470), 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 the section called “Wait-for-Trigger Mode” on page 455).

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 **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 (TnSNAPS bit in the **GPTMTnMR** register is set), the actual free-running value of the timer at the time-out event is loaded into the **GPTMTnR** register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the GPTM generates interrupts 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 475), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 481). If the timeout interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register (see page 473), the GPTM also sets the TnTOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 478). By setting the TnMIE bit in the **GPTMTAMR** register, an interrupt

can also be generated when the Timer value equals the value loaded into the **GPTM Timer n Match (GPTMTnMATCH)** and **GPTM Timer n Prescale Match (GPTMTnPMR)** registers. This interrupt has the same status, masking, and clearing functions as the timeout interrupt. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL**. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

If software updates the **GPTMTnILR** register while the counter is counting down, the counter loads the new value on the next clock cycle and continues counting down from the new value. If software updates the **GPTMTnILR** 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** register while the counter is counting, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the TnSTALL bit in the **GPTMCTL** register is set, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution.

The following table shows a variety of configurations for a 16-bit free-running timer while using the prescaler. All values assume an 80-MHz clock with Tc=12.5 ns (clock period).

**Table 10-5. 16-Bit Timer With Prescaler Configurations**

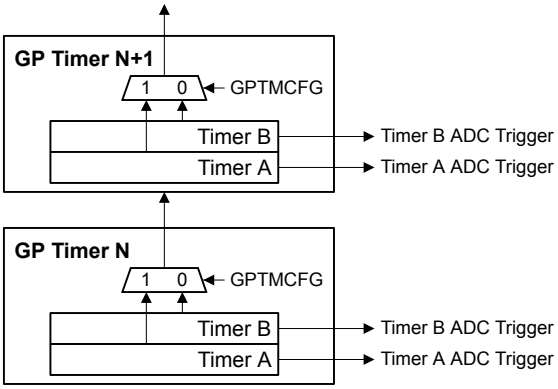
| Prescale | #Clock (Tc)[a] | Max Time | Units |
|---|---|---|---|
| 00000000 | 1 | 0.8192 | ms |
| 00000001 | 2 | 1.6384 | ms |
| 00000010 | 3 | 2.4576 | ms |
| ---------- | -- | -- | -- |
| 11111101 | 254 | 208.0768 | ms |
| 11111110 | 255 | 208.896 | ms |
| 11111111 | 256 | 209.7152 | ms |

a. Tc is the clock period.

### *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 mulitple 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 in 32-bit mode (controlled by the GPTMCFG bit in the **GPTMCFG** register), it triggers Timer A in the next module. If Timer A is in 16-bit mode, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Care must be taken that the TAWOT bit is never set in GPTM0. Figure 10-2 on page 456 shows how the GPTMCFG bit affects the daisy chain. This function is valid for both one-shot and periodic modes.

**Figure 10-2. Timer Daisy Chain**




### 10.3.2.2 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 A Interval Load (GPTMTAILR)** register (see page 483).

The input clock on a CCP 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 **GPTMTAMATCHR** register, 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 addition to generating interrupts, a µDMA trigger can be generated. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

If the TASTALL and/or TBSTALL bits in the **GPTMCTL** register are set, the timer does not freeze if the RTCEN bit is set in **GPTMCTL**.

### 10.3.2.3 Input Edge-Count Mode

**Note:** For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Count mode, the timer is configured as a 24-bit down-counter including the optional prescaler with the upper count value stored in the **GPTM Timer n Prescale (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, 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.

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 the counter by 1 until the event count matches **GPTMTnMATCHR** and **GPTMTnPMR**. When the counts match, the GPTM asserts the CnMRIS bit in the **GPTMRIS** register (and the CnMMIS bit, if the interrupt is not masked).

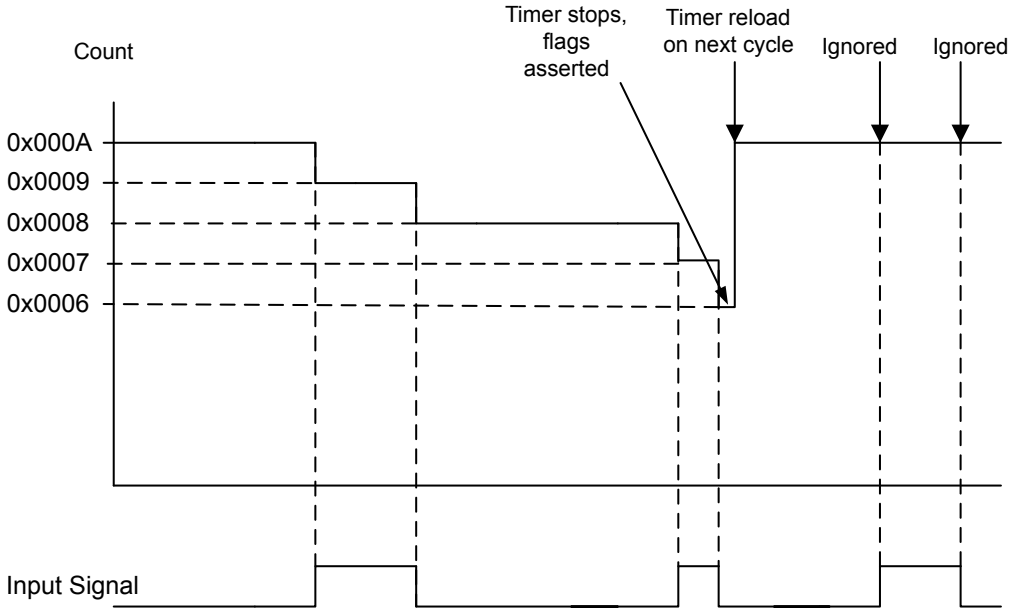
In addition to generating interrupts, an ADC and/or a µDMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL**.The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

After the match value is reached, 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.

Figure 10-3 on page 457 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 10-3. Input Edge-Count Mode Example**




### 10.3.2.4 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.

The prescaler is not available in 16-Bit Input Edge-Time mode.

In Edge-Time mode, the timer is configured as a 16-bit down-counter. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR**register. 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.

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** register and is available to be read by the microcontroller. The GPTM then asserts the CnERIS bit (and the CnEMIS bit, if the interrupt is not masked). The **GPTMTnV** contains the free-running value of the timer and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

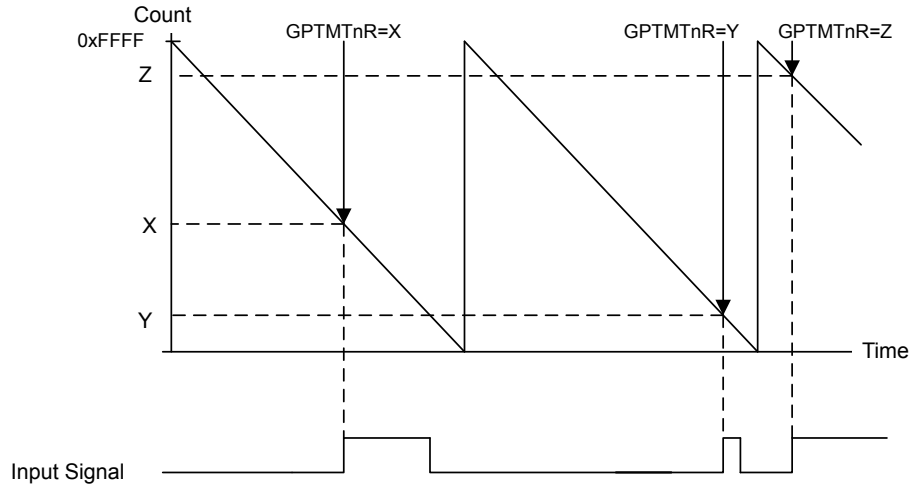
In addition to generating interrupts, an ADC and/or a µDMA trigger can be generated. The ADC trigger is enabled by setting the TnOTE bit in **GPTMCTL**.The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 339.

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 the value from the **GPTMTnILR** register.

Figure 10-4 on page 458 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** register, and is held there until another rising edge is detected (at which point the new count value is loaded into the **GPTMTnR** register).

**Figure 10-4. 16-Bit Input Edge-Time Mode Example**




### 10.3.2.5 PWM Mode

**Note:** The prescaler is not available in 16-Bit PWM mode.

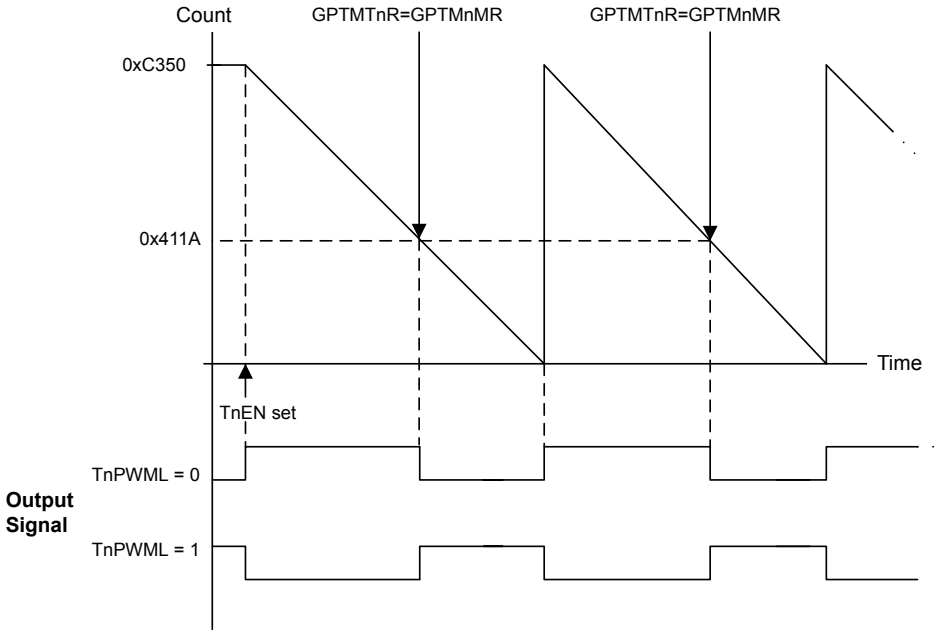
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 16-bit down-counter with a start value (and thus period) defined by the **GPTMTnILR** register. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the **GPTMTnMR** register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x1 or 0x2.

When software writes the TnEN bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0 state. On the next counter cycle in periodic mode, the counter reloads its start value from the **GPTMTnILR** register and continues counting until disabled by software clearing the TnEN bit in the **GPTMCTL** register. No interrupts or status bits are asserted in PWM mode.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** register (its start state), and is deasserted when the counter value equals the value in the **GPTMTnMATCHR** register. Software has the capability of inverting the output PWM signal by setting the TnPWML bit in the **GPTMCTL** register.

Figure 10-5 on page 459 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 10-5. 16-Bit PWM Mode Example**




## 10.3.3 DMA Operation

The timers each have a dedicated µDMA channel and can provide a request signal to the µDMA controller. 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.

No other special steps are needed to enable Timers for µDMA operation. Refer to “Micro Direct Memory Access (µDMA)” on page 335 for more details about programming the µDMA controller.

### 10.3.4 Accessing Concatenated 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 registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 483
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 484
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 491
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 492
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 493
- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 494
- **GPTM Timer A Match (GPTMTAMATCHR)** register [15:0], see page 485
- **GPTM Timer B Match (GPTMTBMATCHR)** register [15:0], see page 486

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]
```

## 10.4 Initialization and Configuration

To use a GPTM, the appropriate TIMERn bit must be set in the **RCGC1** register (see page 243). If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the **RCGC1** register (see page 243). To find out which GPIO port to enable, refer to Table 17-4 on page 753. Configure the PMCn fields in the **GPIOPCTL** register to assign the CCP signals to the appropriate pins (see page 435 and Table 17-5 on page 757).

This section shows module initialization and configuration examples for each of the supported timer modes.

### 10.4.1 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.

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.

### 10.4.2 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. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0001.

3. Write the match value to the **GPTM Timer n Match Register (GPTMTnMATCHR)**.

4. Set/clear the RTCEN bit in the **GPTM Control Register (GPTMCTL)** as needed.

5. If interrupts are required, set the RTCIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.

6. 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.

### 10.4.3 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. 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. Load the event count into the **GPTM Timer n Match (GPTMTnMATCHR)** register.
8. If interrupts are required, set the CnMIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
9. Set the TnEN bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
10. 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 step 4 on page 462 through step 9 on page 462.

### 10.4.4 Input Edge Timing Mode

A timer is configured to Input Edge Timing 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.
4. Configure the type of event that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. If interrupts are required, set the CnEIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
7. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
8. 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. The change takes effect at the next cycle after the write.

### 10.4.5 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. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the match value.
7. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

In PWM Timing 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.

## 10.5 Register Map

Table 10-6 on page 463 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- Timer 0: 0x4003.0000
- Timer 1: 0x4003.1000
- Timer 2: 0x4003.2000
- Timer 3: 0x4003.3000

Note that the GP Timer module clock must be enabled before the registers can be programmed (see page 243). There must be a delay of 3 system clocks after the Timer module clock is enabled before any Timer module registers are accessed.

**Table 10-6. Timers Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | GPTMCFG | R/W | 0x0000.0000 | GPTM Configuration | 465 |
| 0x004 | GPTMTAMR | R/W | 0x0000.0000 | GPTM Timer A Mode | 466 |
| 0x008 | GPTMTBMR | R/W | 0x0000.0000 | GPTM Timer B Mode | 468 |
| 0x00C | GPTMCTL | R/W | 0x0000.0000 | GPTM Control | 470 |
| 0x018 | GPTMIMR | R/W | 0x0000.0000 | GPTM Interrupt Mask | 473 |
| 0x01C | GPTMRIS | RO | 0x0000.0000 | GPTM Raw Interrupt Status | 475 |
| 0x020 | GPTMMIS | RO | 0x0000.0000 | GPTM Masked Interrupt Status | 478 |
| 0x024 | GPTMICR | W1C | 0x0000.0000 | GPTM Interrupt Clear | 481 |
| 0x028 | GPTMTAILR | R/W | 0xFFFF.FFFF | GPTM Timer A Interval Load | 483 |
| 0x02C | GPTMTBILR | R/W | 0x0000.FFFF | GPTM Timer B Interval Load | 484 |
| 0x030 | GPTMTAMATCHR | R/W | 0xFFFF.FFFF | GPTM Timer A Match | 485 |

**Table 10-6. Timers Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x034 | GPTMTBMATCHR | R/W | 0x0000.FFFF | GPTM Timer B Match | 486 |
| 0x038 | GPTMTAPR | R/W | 0x0000.0000 | GPTM Timer A Prescale | 487 |
| 0x03C | GPTMTBPR | R/W | 0x0000.0000 | GPTM Timer B Prescale | 488 |
| 0x040 | GPTMTAPMR | R/W | 0x0000.0000 | GPTM TimerA Prescale Match | 489 |
| 0x044 | GPTMTBPMR | R/W | 0x0000.0000 | GPTM TimerB Prescale Match | 490 |
| 0x048 | GPTMTAR | RO | 0xFFFF.FFFF | GPTM Timer A | 491 |
| 0x04C | GPTMTBR | RO | 0x0000.FFFF | GPTM Timer B | 492 |
| 0x050 | GPTMTAV | RW | 0xFFFF.FFFF | GPTM Timer A Value | 493 |
| 0x054 | GPTMTBV | RW | 0x0000.FFFF | GPTM Timer B Value | 494 |

## 10.6 Register Descriptions

The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

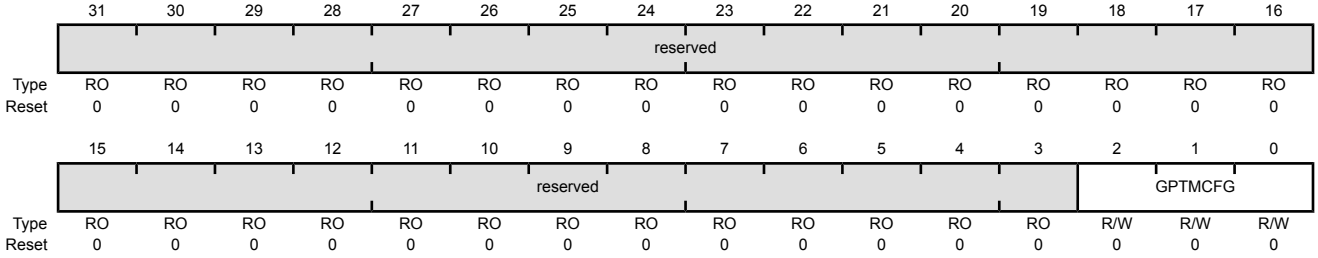
## Register 1: GPTM Configuration (GPTMCFG), offset 0x000

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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x000
Type R/W, 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 | R/W | 0x0 | GPTM Configuration<br>The GPTMCFG values are defined as follows:<br><br>Value Description<br>0x0 32-bit timer configuration.<br>0x1 32-bit real-time clock (RTC) counter configuration.<br>0x2-0x3 Reserved<br>0x4 16-bit timer configuration.<br>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:** Bits in this register should only be changed when the TAEN bit in the **GPTMCTL** register is cleared.

GPTM Timer A Mode (GPTMTAMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x004
Type R/W, 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 | | | | | | | | TASNAPS | TAWOT | TAMIE | TACDIR | TAAMS | TACMR | TAMR | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | TASNAPS | R/W | 0 | GPTM Timer A Snap-Shot Mode<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 value of Timer A is loaded at the time-out event into the **GPTM Timer A (GPTMTAR)** register. |
| 6 | TAWOT | R/W | 0 | GPTM Timer A Wait-on-Trigger<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 10-2 on page 456. This function is valid for both one-shot and periodic modes.<br>This bit must be clear for GP Timer Module 0, Timer A. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TAMIE | R/W | 0 | GPTM Timer A Match Interrupt Enable<br><br>Value Description<br>0 The match interrupt is disabled.<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 | R/W | 0 | GPTM Timer A Count Direction<br><br>Value Description<br>0 The timer counts down.<br>1 When in one-shot or periodic mode, 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 | TAAMS | R/W | 0 | GPTM Timer A Alternate Mode Select<br>The TAAMS values are defined as follows:<br><br>Value Description<br>0 Capture 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 | R/W | 0 | GPTM Timer A Capture Mode<br>The TACMR values are defined as follows:<br><br>Value Description<br>0 Edge-Count mode<br>1 Edge-Time mode |
| 1:0 | TAMR | R/W | 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 **GPTMTBMR** controls the modes for both Timer A and Timer B.

**Important:** Bits in this register should only be changed when the TBEN bit in the **GPTMCTL** register is cleared.

GPTM Timer B Mode (GPTMTBMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x008
Type R/W, 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 | | | | | | | | TBSNAPS | TBWOT | TBMIE | TBCDIR | TBAMS | TBCMR | TBMR | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | TBSNAPS | R/W | 0 | GPTM Timer B Snap-Shot Mode<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. |
| 6 | TBWOT | R/W | 0 | GPTM Timer B Wait-on-Trigger<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 an it receives a trigger from the timer in the previous position in the daisy chain, see Figure 10-2 on page 456. This function is valid for both one-shot and periodic modes. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TBMIE | R/W | 0 | GPTM Timer B Match Interrupt Enable<br><br>Value Description<br>0 The match interrupt is disabled.<br>1 An interrupt is generated when the match value in the **GPTMTBMATCHR** register is reached in the one-shot and periodic modes. |
| 4 | TBCDIR | R/W | 0 | GPTM Timer B Count Direction<br><br>Value Description<br>0 The timer counts down.<br>1 When in one-shot or periodic mode, 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 | R/W | 0 | GPTM Timer B Alternate Mode Select<br>The TBAMS values are defined as follows:<br><br>Value Description<br>0 Capture 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 | R/W | 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 | R/W | 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. |

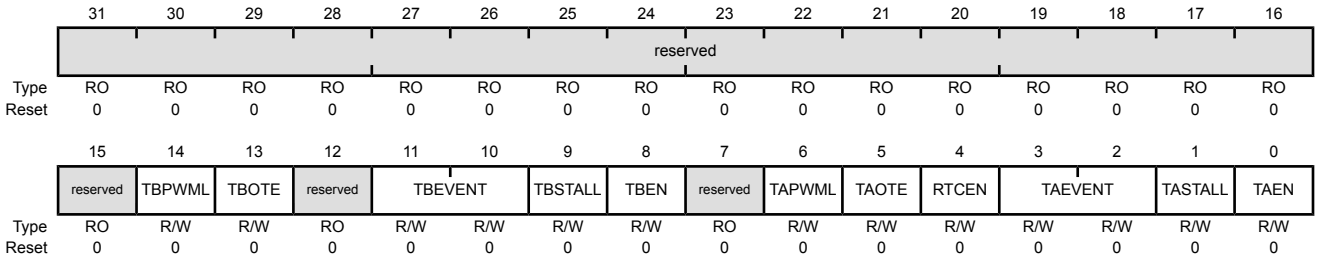
## Register 4: GPTM Control (GPTMCTL), offset 0x00C

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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x00C
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:15 | 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. |
| 14 | TBPWML | R/W | 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 | R/W | 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 552). |
| 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 | R/W | 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 |
| 9 | TBSTALL | R/W | 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 | R/W | 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 | R/W | 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. |
| 5 | TAOTE | R/W | 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 552). |

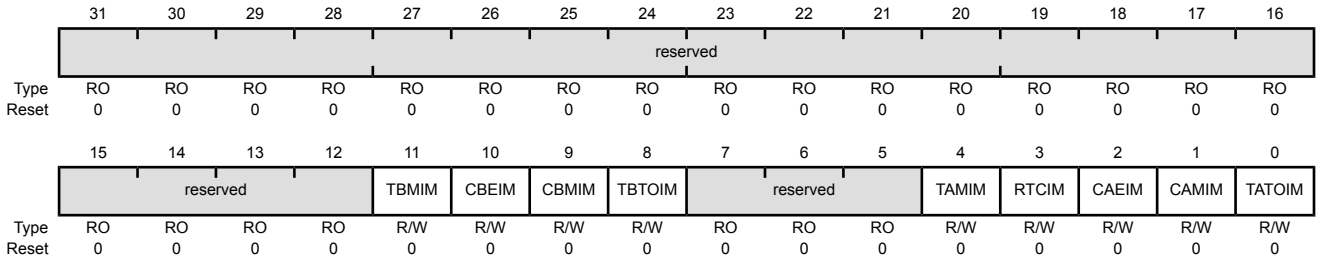
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | RTCEN | R/W | 0 | GPTM RTC Enable<br>The RTCEN values are defined as follows:<br><br>Value Description<br>0 RTC counting is disabled.<br>1 RTC counting is enabled. |
| 3:2 | TAEVENT | R/W | 0x0 | GPTM Timer A Event Mode<br>The 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 |
| 1 | TASTALL | R/W | 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. |
| 0 | TAEN | R/W | 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 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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x018
Type R/W, 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 | TBMIM | R/W | 0 | GPTM Timer B Mode 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 | R/W | 0 | GPTM Capture B 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. |
| 9 | CBMIM | R/W | 0 | GPTM Capture B 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. |

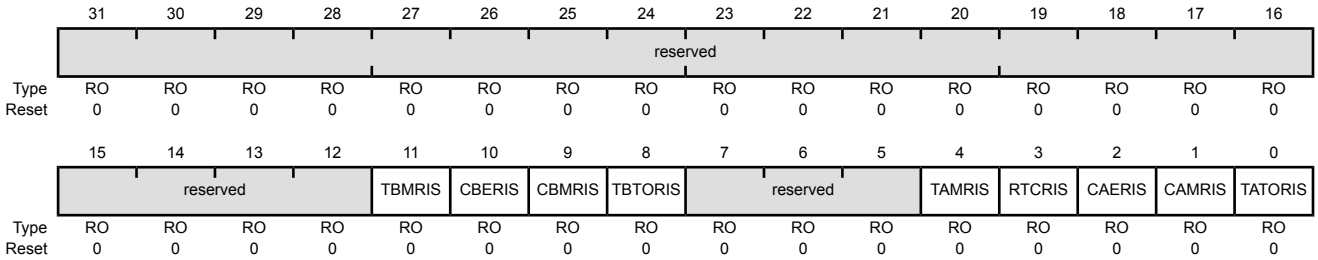
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TBTOIM | R/W | 0 | GPTM Timer B Time-Out Interrupt Mask<br>The TBTOIM values are defined as follows:<br><br>Value Description<br>0 Interrupt is disabled.<br>1 Interrupt is enabled. |
| 7:5 | reserved | RO | 0 | Software should 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 | TAMIM | R/W | 0 | GPTM Timer A Mode 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 | R/W | 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 | R/W | 0 | GPTM Capture A 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. |
| 1 | CAMIM | R/W | 0 | GPTM Capture A 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 | R/W | 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 6: 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**.

GPTM Raw Interrupt Status (GPTMRIS)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x01C
Type RO, 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 | TBMRIS | RO | 0 | GPTM Timer B Mode Match Raw Interrupt<br><br>Value Description<br>1 The TBMIE bit is set in the **GPTMTBMR** register, and the match value in the **GPTMTBMATCHR** register has been reached when in the one-shot and periodic modes.<br>0 The match value has not been reached.<br><br>This bit is cleared by writing a 1 to the TBMCINT bit in the **GPTMICR** register. |
| 10 | CBERIS | RO | 0 | GPTM Capture B Event Raw Interrupt<br><br>Value Description<br>1 The Capture B event has occurred.<br>0 The Capture B event has not occurred.<br><br>This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register. |
| 9 | CBMRIS | RO | 0 | GPTM Capture B Match Raw Interrupt<br><br>Value Description<br>1 The Capture B match has occurred.<br>0 The Capture B match has not occurred.<br><br>This bit is cleared by writing a 1 to the CBMCINT bit in the **GPTMICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TBTORIS | RO | 0 | GPTM Timer B Time-Out Raw Interrupt<br><br>Value Description<br>1 Timer B has timed out.<br>0 Timer B has not timed out.<br><br>This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register. |
| 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 | TAMRIS | RO | 0 | GPTM Timer A Mode Match Raw Interrupt<br><br>Value Description<br>1 The TAMIE bit is set in the **GPTMTAMR** register, and the match value in the **GPTMTAMATCHR** register has been reached when in the one-shot and periodic modes.<br>0 The match value has not been reached.<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>1 The RTC event has occurred.<br>0 The RTC event has not occurred.<br><br>This bit is cleared by writing a 1 to the RTCCINT bit in the **GPTMICR** register. |
| 2 | CAERIS | RO | 0 | GPTM Capture A Event Raw Interrupt<br><br>Value Description<br>1 The Capture A event has occurred.<br>0 The Capture A event has not occurred.<br><br>This bit is cleared by writing a 1 to the CAECINT bit in the **GPTMICR** register. |
| 1 | CAMRIS | RO | 0 | GPTM Capture A Match Raw Interrupt<br><br>Value Description<br>1 The Capture A match has occurred.<br>0 The Capture A match has not occurred.<br><br>This bit is cleared by writing a 1 to the CAMCINT bit in the **GPTMICR** register. |

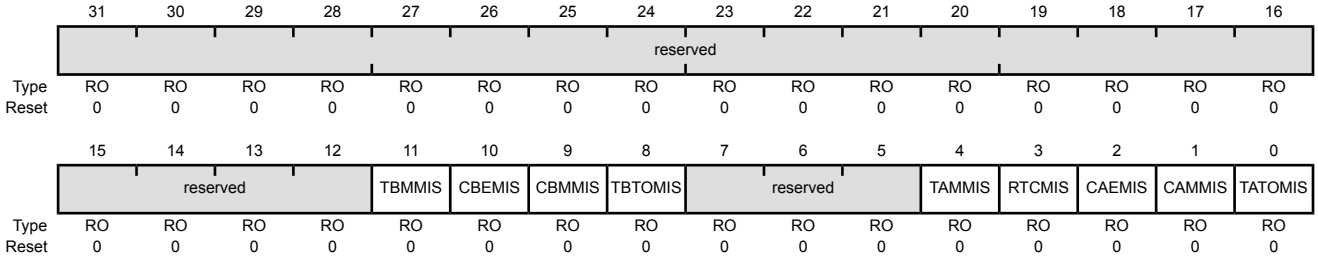
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TATORIS | RO | 0 | GPTM Timer A Time-Out Raw Interrupt<br><br>Value Description<br>1 Timer A has timed out.<br>0 Timer A has not timed out.<br><br>This bit is cleared by writing a 1 to the TATOCINT bit in the **GPTMICR** register. |

## Register 7: 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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x020
Type RO, 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 | TBMMIS | RO | 0 | GPTM Timer B Mode Match Masked Interrupt<br><br>Value Description<br>1 An unmasked Timer B Mode Match interrupt has occurred.<br>0 A Timer B Mode Match interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the TBMCINT bit in the **GPTMICR** register. |
| 10 | CBEMIS | RO | 0 | GPTM Capture B Event Masked Interrupt<br><br>Value Description<br>1 An unmasked Capture B event interrupt has occurred.<br>0 A Capture B event interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the CBECINT bit in the **GPTMICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | CBMMIS | RO | 0 | GPTM Capture B Match Masked Interrupt<br><br>Value Description<br>1 An unmasked Capture B Match interrupt has occurred.<br>0 A Capture B Mode Match interrupt has not occurred or is masked.<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>1 An unmasked Timer B Time-Out interrupt has occurred.<br>0 A Timer B Time-Out interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the TBTOCINT bit in the **GPTMICR** register. |
| 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 | TAMMIS | RO | 0 | GPTM Timer A Mode Match Masked Interrupt<br><br>Value Description<br>1 An unmasked Timer A Mode Match interrupt has occurred.<br>0 A Timer A Mode Match interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the TAMCINT bit in the **GPTMICR** register. |
| 3 | RTCMIS | RO | 0 | GPTM RTC Masked Interrupt<br><br>Value Description<br>1 An unmasked RTC event interrupt has occurred.<br>0 An RTC event interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the RTCCINT bit in the **GPTMICR** register. |
| 2 | CAEMIS | RO | 0 | GPTM Capture A Event Masked Interrupt<br><br>Value Description<br>1 An unmasked Capture A event interrupt has occurred.<br>0 A Capture A event interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the CAECINT bit in the **GPTMICR** register. |

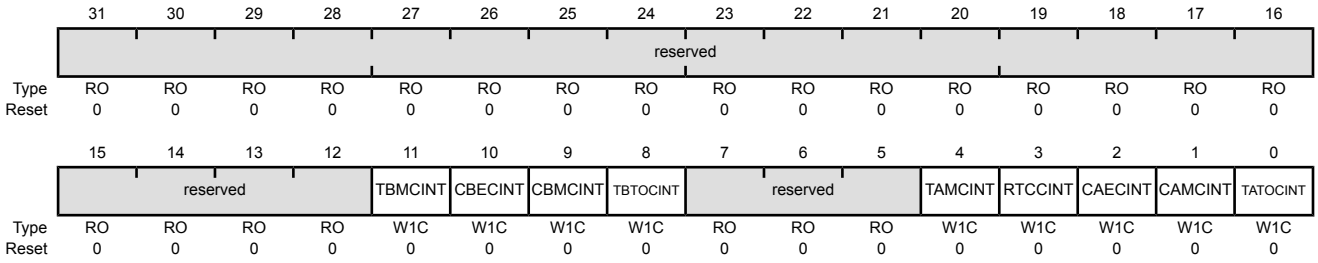
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CAMMIS | RO | 0 | GPTM Capture A Match Masked Interrupt<br><br>Value Description<br>1 An unmasked Capture A Match interrupt has occurred.<br>0 A Capture A Mode Match interrupt has not occurred or is masked.<br><br>This bit 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>1 An unmasked Timer A Time-Out interrupt has occurred.<br>0 A Timer A Time-Out interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the TATOCINT bit in the **GPTMICR** register. |

## Register 8: 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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x024
Type W1C, 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 | TBMCINT | W1C | 0 | GPTM Timer B Mode 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 Capture B 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 Capture B 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. |
| 7:5 | 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 | TAMCINT | W1C | 0 | GPTM Timer A Mode 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 Capture A 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CAMCINT | W1C | 0 | GPTM Capture A 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 9: 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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x028
Type R/W, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAILR | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TAILR | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | TAILR | R/W | 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 10: 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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x02C
Type R/W, reset 0x0000.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBILR | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | 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 GPTM is in 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**. |

## Register 11: 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.

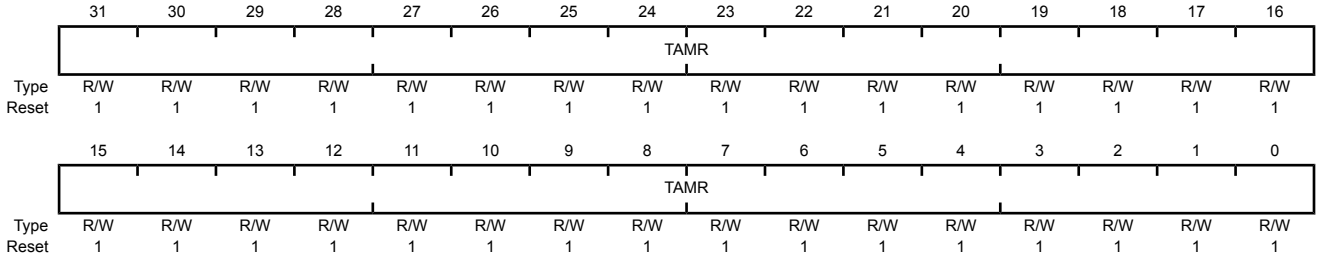
In Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

In PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal.

When a 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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x030
Type R/W, reset 0xFFFF.FFFF




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TAMR | R/W | 0xFFFF.FFFF | GPTM Timer A Match Register<br>This value is compared to the **GPTMTAR** register to determine match events. |

## Register 12: 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.

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)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x034
Type R/W, reset 0x0000.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBMR | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

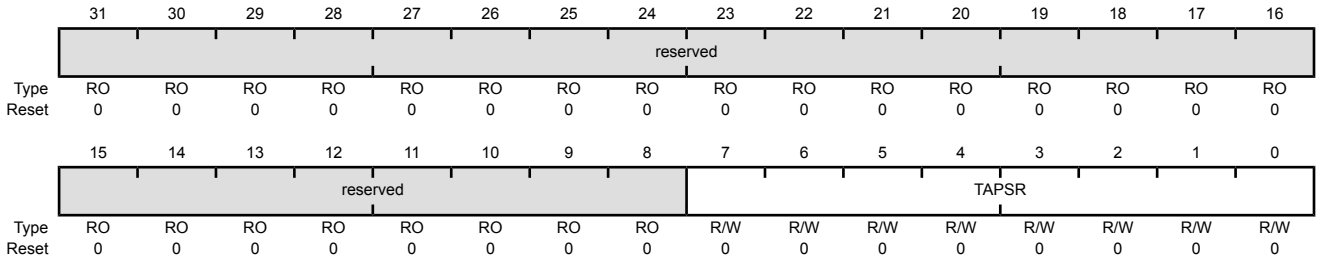
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | TBMR | R/W | 0x0000.FFFF | GPTM Timer B Match Register<br>This value is compared to the **GPTMTBR** register to determine match events. |

## Register 13: GPTM Timer A Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the 16-bit timers in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

GPTM Timer A Prescale (GPTMTAPR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x038
Type R/W, reset 0x0000.0000



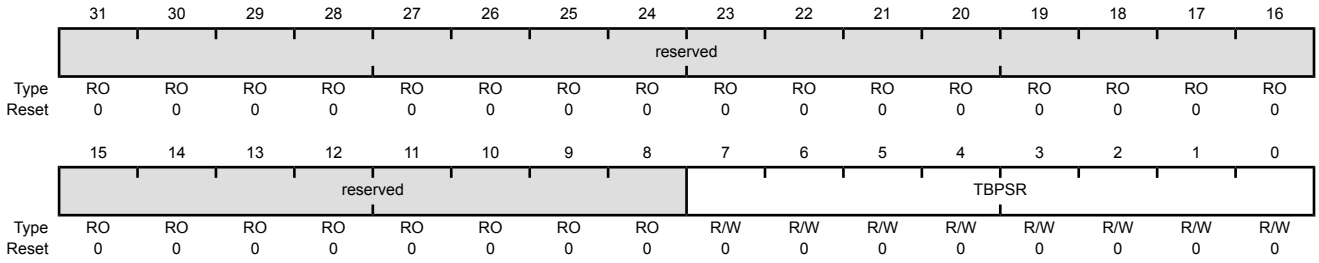

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TAPSR | R/W | 0x00 | GPTM Timer A Prescale<br>The register loads this value on a write. A read returns the current value of the register.<br>Refer to Table 10-5 on page 455 for more details and an example. |

## Register 14: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C

This register allows software to extend the range of the 16-bit timers in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

GPTM Timer B Prescale (GPTMTBPR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x03C
Type R/W, reset 0x0000.0000



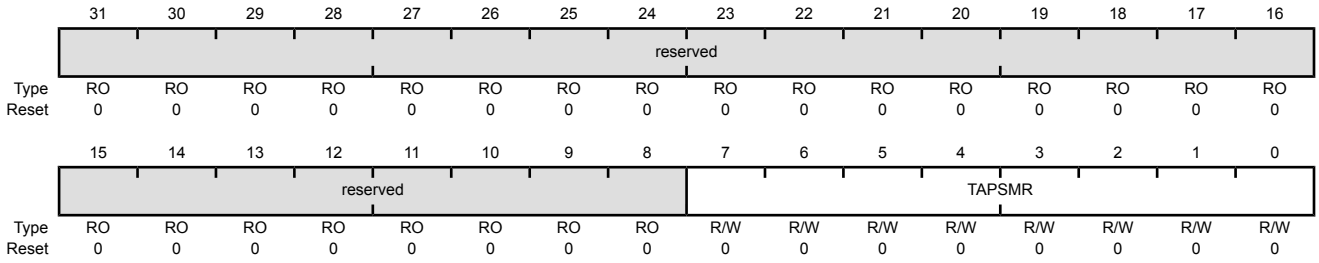

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TBPSR | R/W | 0x00 | GPTM Timer B Prescale<br>The register loads this value on a write. A read returns the current value of this register.<br>Refer to Table 10-5 on page 455 for more details and an example. |

## Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040

This register effectively extends the range of **GPTMTAMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerA Prescale Match (GPTMTAPMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x040
Type R/W, reset 0x0000.0000



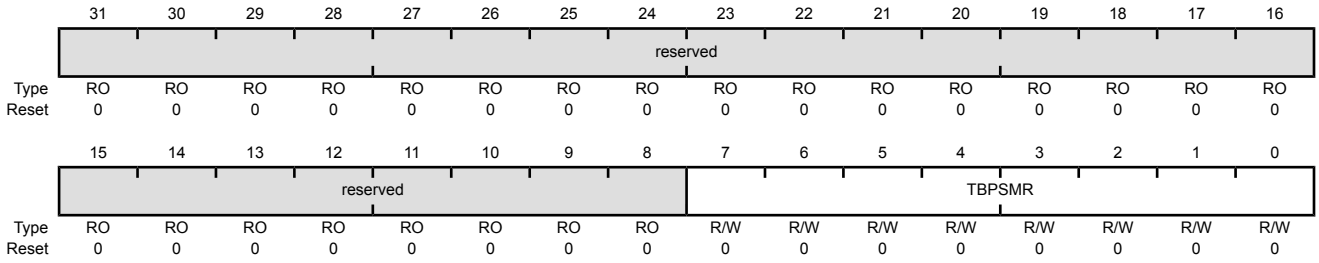

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TAPSMR | R/W | 0x00 | GPTM TimerA Prescale Match<br>This value is used alongside **GPTMTAMATCHR** to detect timer match events while using a prescaler. |

## Register 16: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044

This register effectively extends the range of **GPTMTBMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerB Prescale Match (GPTMTBPMR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x044
Type R/W, 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 | R/W | 0x00 | GPTM TimerB Prescale Match<br>This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler. |

## Register 17: 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. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

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.

GPTM Timer A (GPTMTAR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
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 18: 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. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

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 are reserved in both cases.

GPTM Timer B (GPTMTBR)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
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 19: GPTM Timer A Value (GPTMTAV), offset 0x050

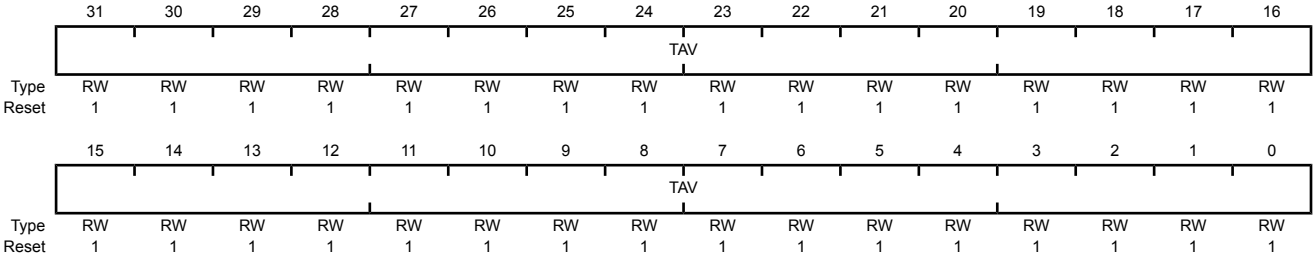
When read, this register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a 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. Bits 31:24 always read as 0.

**Note:** The **GPTMTAV** register cannot be written in Edge-Count mode.

GPTM Timer A Value (GPTMTAV)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
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. |

## Register 20: 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. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

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 **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 current, free-running 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. Bits 31:24 are reserved in both cases.

GPTM Timer B Value (GPTMTBV)

Timer 0 base: 0x4003.0000
Timer 1 base: 0x4003.1000
Timer 2 base: 0x4003.2000
Timer 3 base: 0x4003.3000
Offset 0x054
Type RW, reset 0x0000.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBV | | | | | | | | | | | | | | | |
| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBV | | | | | | | | | | | | | | | |
| 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 | 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. |

# 11 Watchdog Timers

A watchdog timer can generate an 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 LM3S1G58 microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other is clocked by the PIOSC (Watchdog Timer 1). 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.

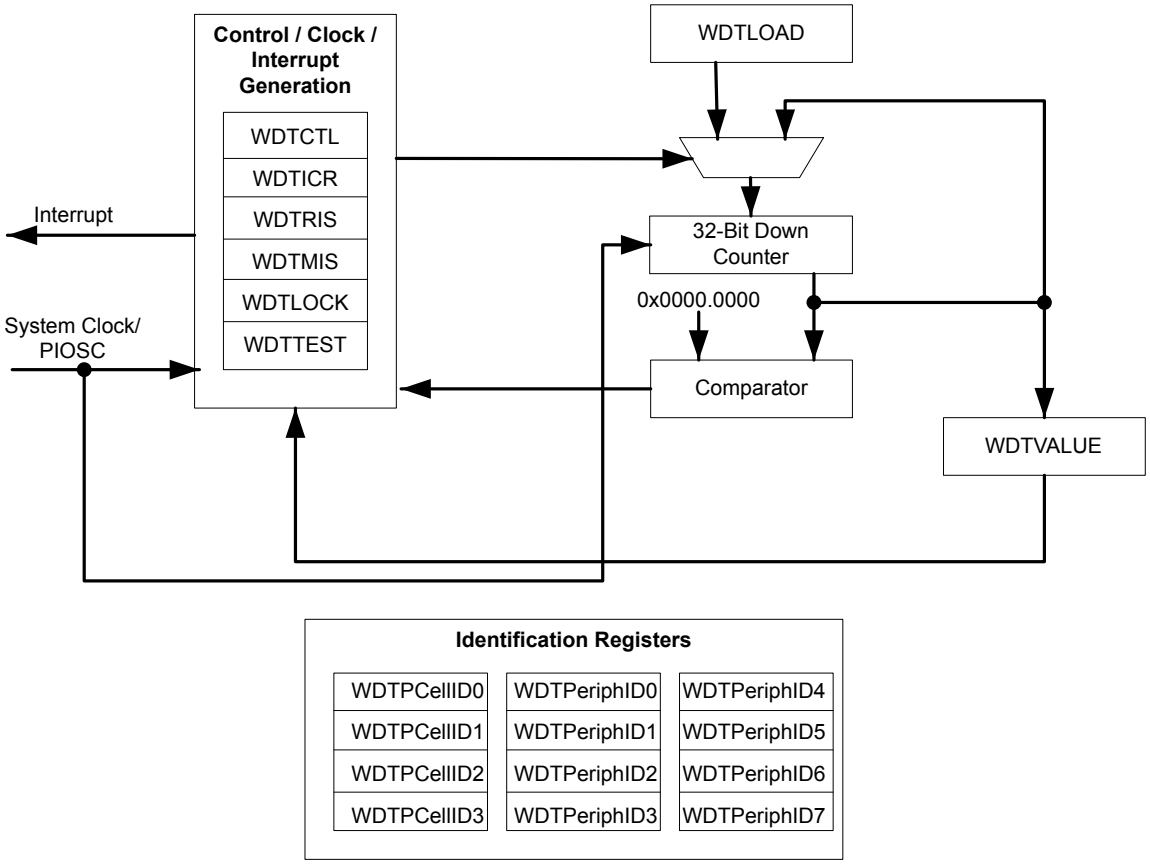
The Stellaris® LM3S1G58 controller has two Watchdog Timer modules with the following features:

- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking
- 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.

## 11.1 Block Diagram

**Figure 11-1. WDT Module Block Diagram**




## 11.2 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. 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.

### 11.2.1 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.

## 11.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the WDT bit in the **RCGC0** register, see page 237.

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. If the Watchdog is configured to trigger system resets, set the RESEN bit in the **WDTCTL** register.
4. If WDT1, wait for the WRC bit in the **WDTCTL** register to be set.
5. Set the INTEN bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

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.

## 11.4 Register Map

Table 11-1 on page 498 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 237).

**Table 11-1. Watchdog Timers Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | WDTLOAD | R/W | 0xFFFF.FFFF | Watchdog Load | 499 |
| 0x004 | WDTVALUE | RO | 0xFFFF.FFFF | Watchdog Value | 500 |
| 0x008 | WDTCTL | R/W | 0x0000.0000 (WDT0) 0x8000.0000 (WDT1) | Watchdog Control | 501 |
| 0x00C | WDTICR | WO | - | Watchdog Interrupt Clear | 503 |
| 0x010 | WDTRIS | RO | 0x0000.0000 | Watchdog Raw Interrupt Status | 504 |
| 0x014 | WDTMIS | RO | 0x0000.0000 | Watchdog Masked Interrupt Status | 505 |
| 0x418 | WDTTEST | R/W | 0x0000.0000 | Watchdog Test | 506 |
| 0xC00 | WDTLOCK | R/W | 0x0000.0000 | Watchdog Lock | 507 |
| 0xFD0 | WDTPeriphID4 | RO | 0x0000.0000 | Watchdog Peripheral Identification 4 | 508 |
| 0xFD4 | WDTPeriphID5 | RO | 0x0000.0000 | Watchdog Peripheral Identification 5 | 509 |
| 0xFD8 | WDTPeriphID6 | RO | 0x0000.0000 | Watchdog Peripheral Identification 6 | 510 |
| 0xFDC | WDTPeriphID7 | RO | 0x0000.0000 | Watchdog Peripheral Identification 7 | 511 |
| 0xFE0 | WDTPeriphID0 | RO | 0x0000.0005 | Watchdog Peripheral Identification 0 | 512 |
| 0xFE4 | WDTPeriphID1 | RO | 0x0000.0018 | Watchdog Peripheral Identification 1 | 513 |
| 0xFE8 | WDTPeriphID2 | RO | 0x0000.0018 | Watchdog Peripheral Identification 2 | 514 |
| 0xFEC | WDTPeriphID3 | RO | 0x0000.0001 | Watchdog Peripheral Identification 3 | 515 |
| 0xFF0 | WDTPCellID0 | RO | 0x0000.000D | Watchdog PrimeCell Identification 0 | 516 |
| 0xFF4 | WDTPCellID1 | RO | 0x0000.00F0 | Watchdog PrimeCell Identification 1 | 517 |
| 0xFF8 | WDTPCellID2 | RO | 0x0000.0006 | Watchdog PrimeCell Identification 2 | 518 |
| 0xFFC | WDTPCellID3 | RO | 0x0000.00B1 | Watchdog PrimeCell Identification 3 | 519 |

## 11.5 Register Descriptions

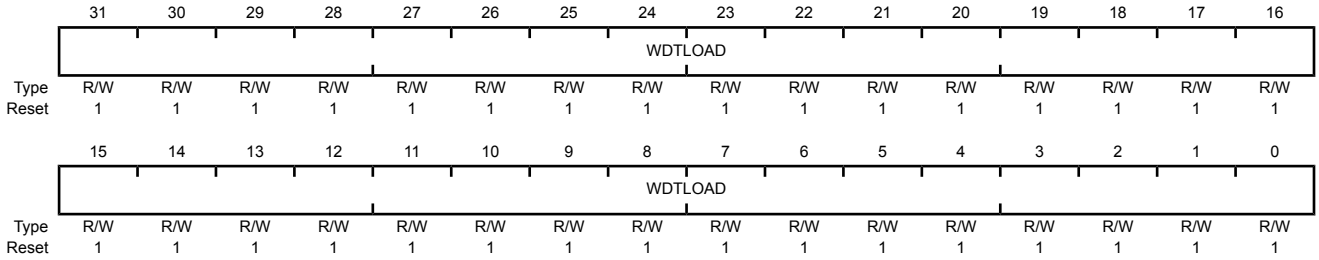
The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

## 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 R/W, reset 0xFFFF.FFFF



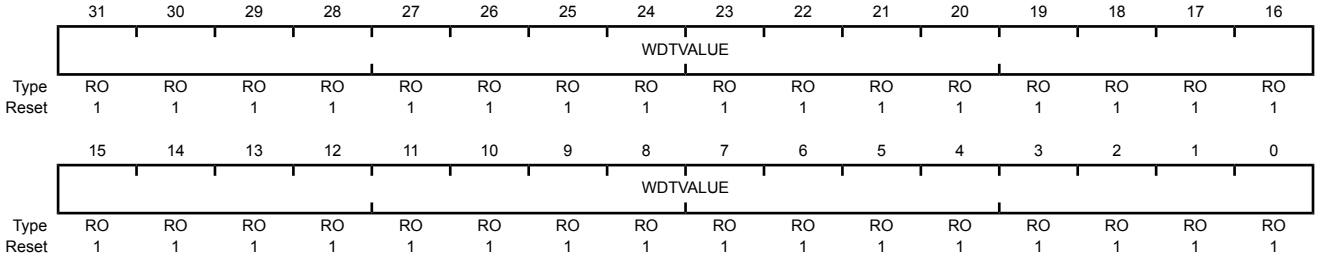

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WDTLOAD | R/W | 0xFFFF.FFFF | Watchdog Load Value |

## 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

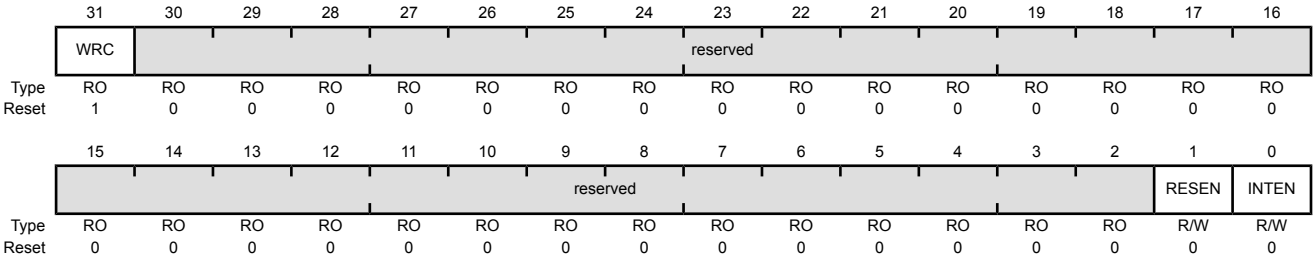
This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled by setting the INTEN bit, all subsequent writes to the INTEN bit are ignored. The only mechanism that can re-enable writes to this bit is a hardware reset.

**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 R/W, 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. |
| 30:2 | 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. |

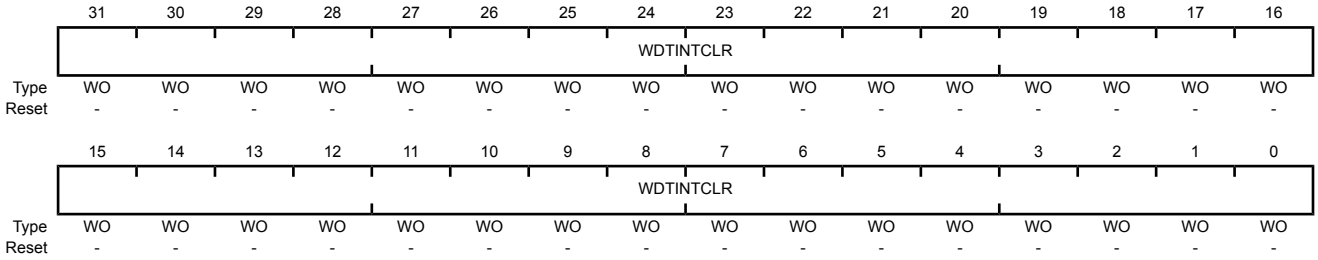
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | RESEN | R/W | 0 | Watchdog Reset Enable<br>The RESEN values are defined as follows:<br><br>Value Description<br>0 Disabled.<br>1 Enable the Watchdog module reset output. |
| 0 | INTEN | R/W | 0 | Watchdog Interrupt Enable<br>The 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).<br>1 Interrupt event enabled. Once enabled, all writes are ignored. |

## 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. Value for a read or reset is indeterminate.

Watchdog Interrupt Clear (WDTICR)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x00C
Type WO, reset -



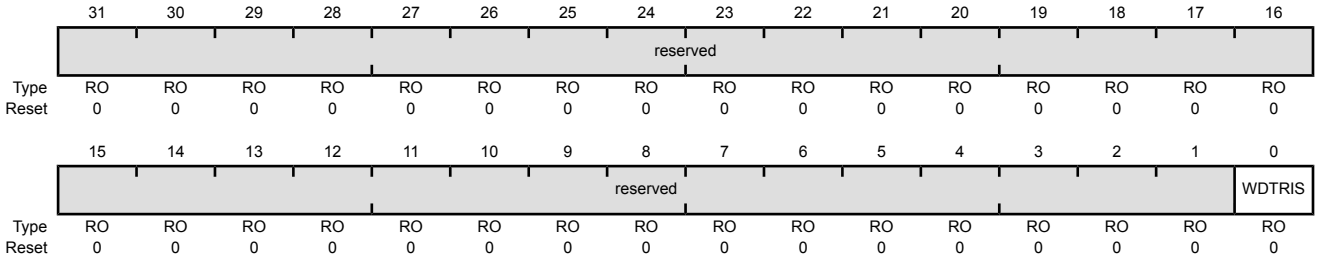

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WDTINTCLR | WO | - | Watchdog Interrupt Clear |

## 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 |

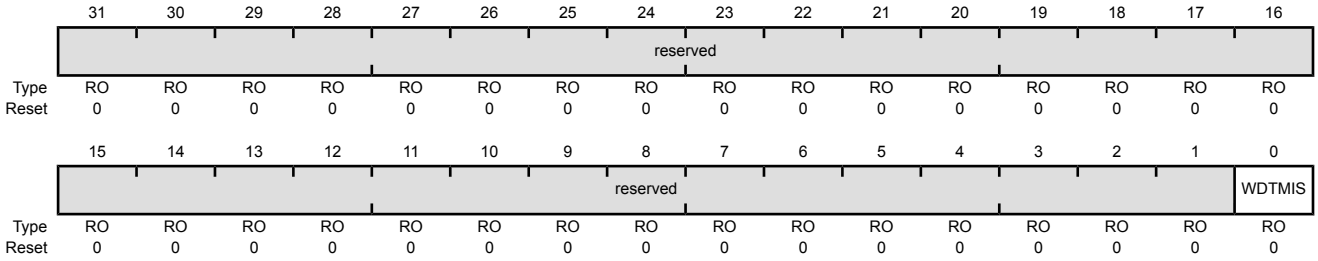
| Value | Description |
|---|---|
| 1 | A watchdog time-out event has occurred. |
| 0 | The watchdog has not timed out. |

## 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 |
|---|---|
| 1 | A watchdog time-out event has been signalled to the interrupt controller. |
| 0 | The watchdog has not timed out or the watchdog timer interrupt is masked. |

## 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 R/W, 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 | R/W | 0 | Watchdog Stall Enable<br>Value Description<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.<br>0 The watchdog timer continues counting if the microcontroller is stopped with a debugger. |
| 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. 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 R/W, reset 0x0000.0000

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

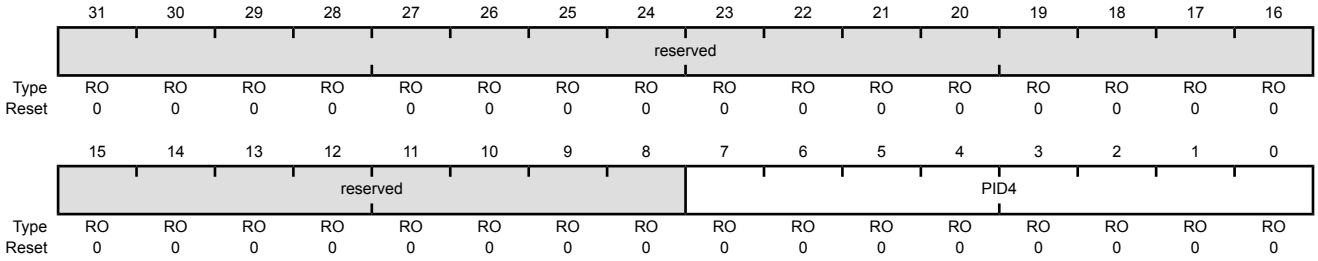
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:0 | WDTLOCK | R/W | 0x0000.0000 | Watchdog Lock<br>A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates.<br>A read of this register returns the following values:<br>Value Description<br>0x0000.0001 Locked<br>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



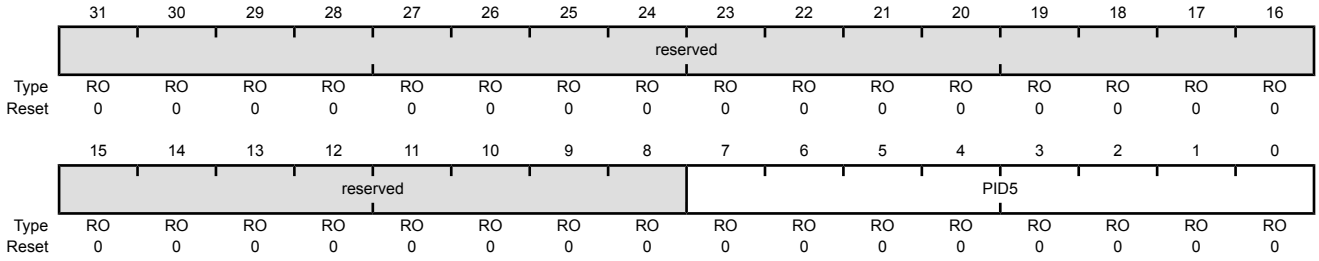

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



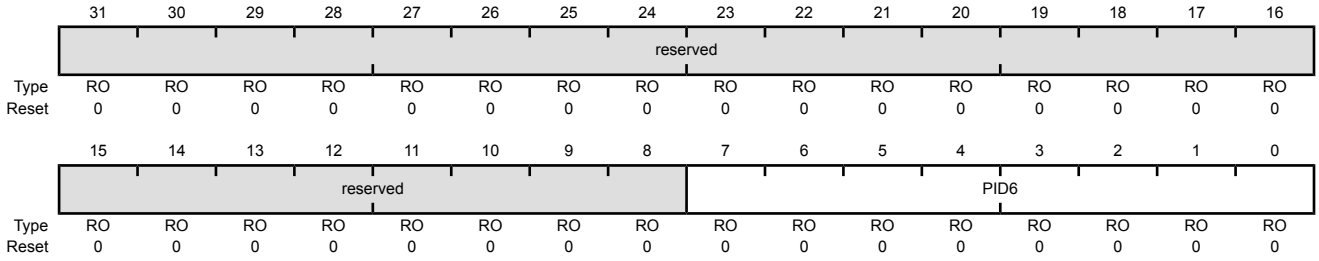

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



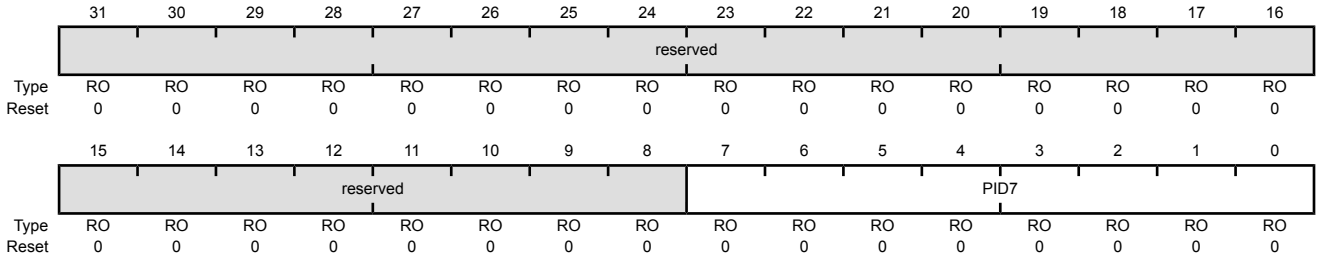

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



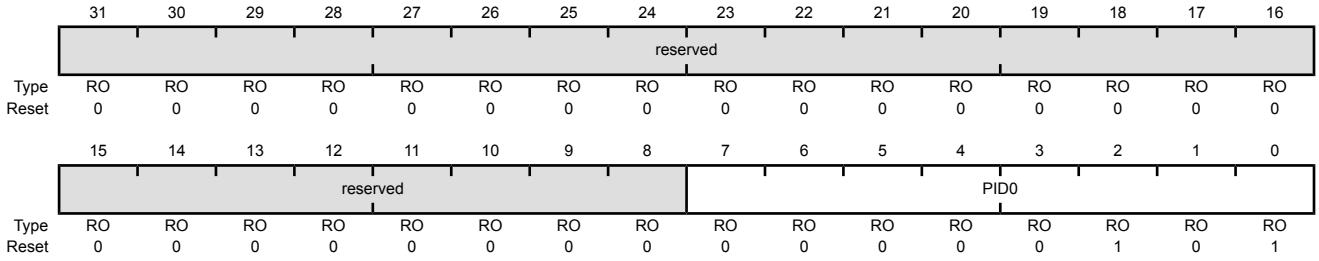

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



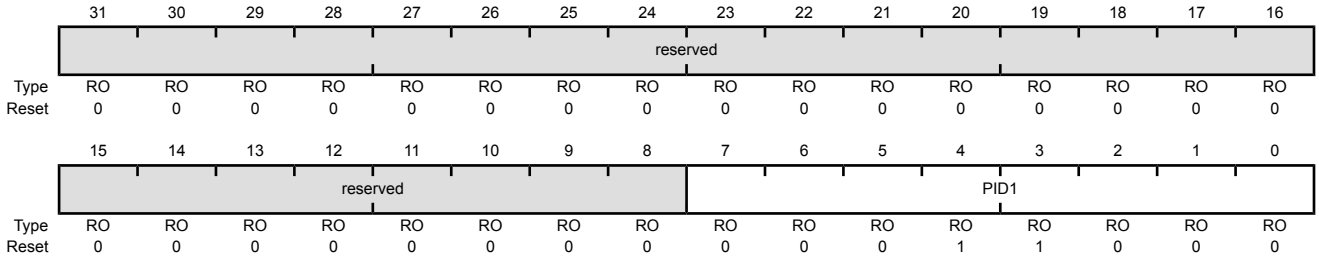

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



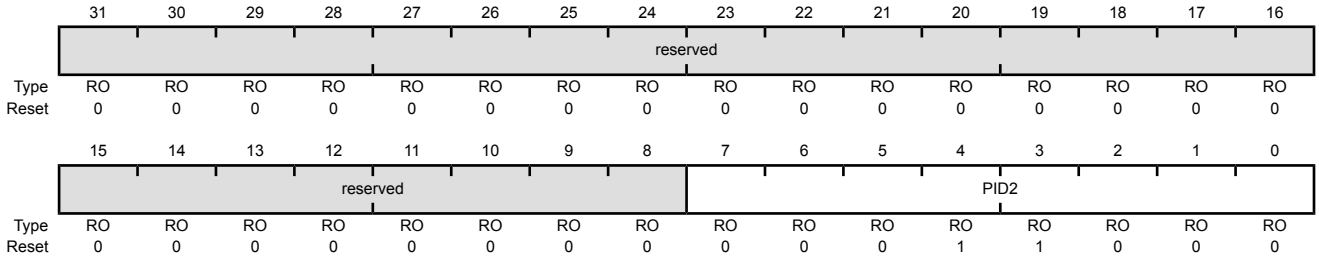

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



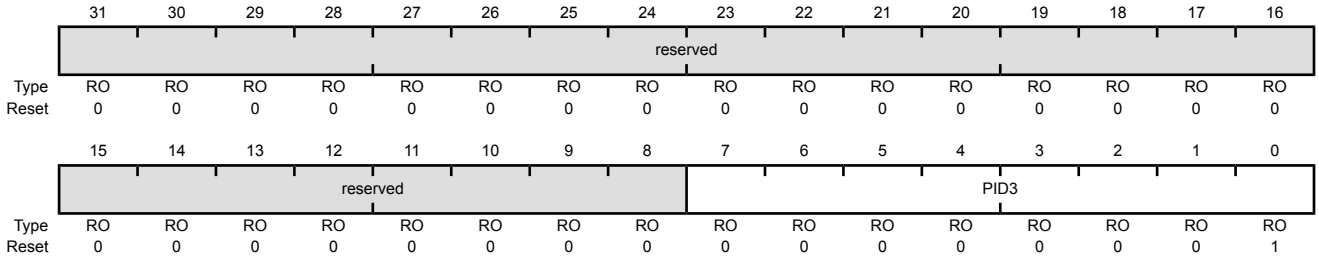

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



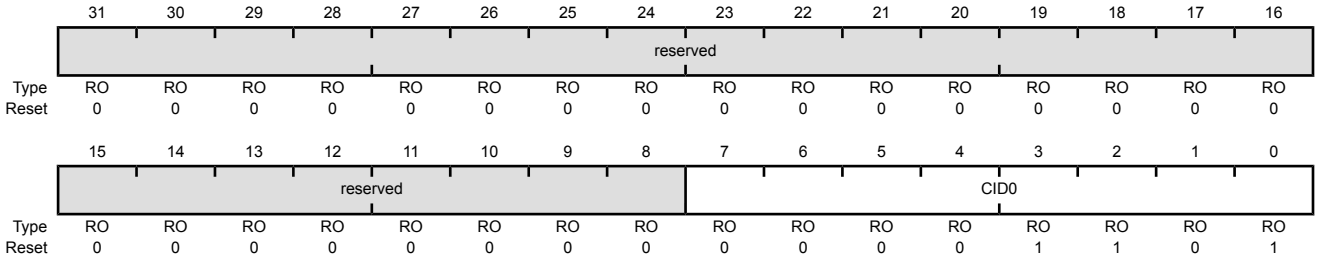

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



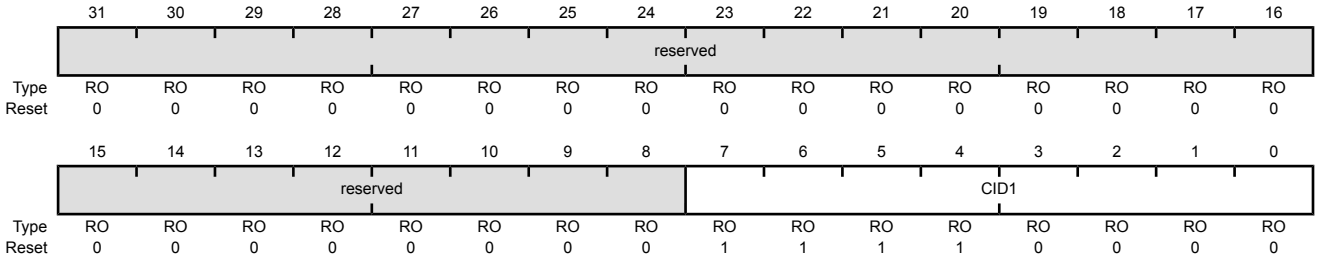

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



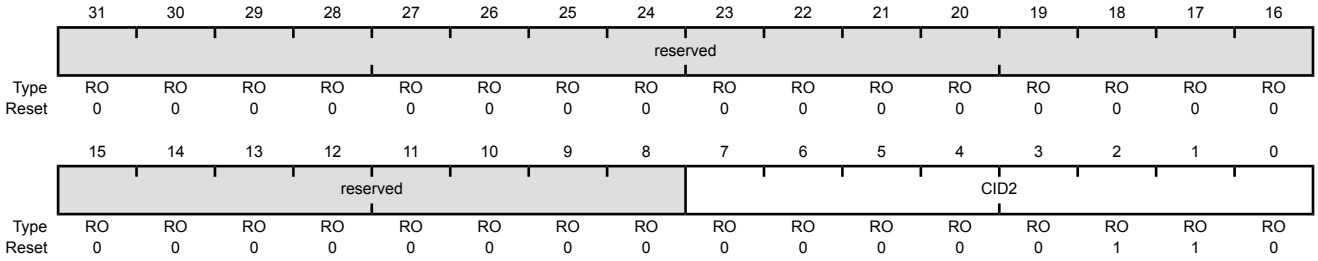

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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



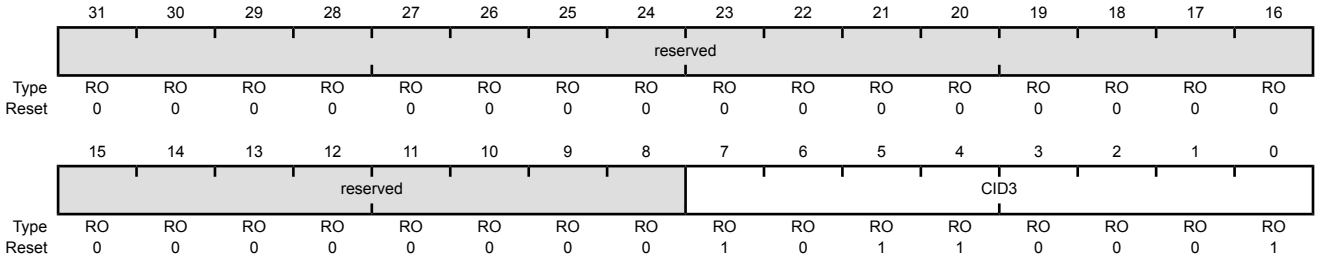

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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] |

# 12 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 16 input channels.

The Stellaris® ADC module features 12-bit conversion resolution and supports 16 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. A digital comparator function is included which allows the conversion value to 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 525.

The Stellaris LM3S1G58 microcontroller provides two ADC modules with each having the following features:

- 16 shared analog input channels
- 12-bit precision ADC with an accurate 10-bit data compatibility mode
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5º to 337.5º
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - GPIO
- Hardware averaging of up to 64 samples
- Digital comparison unit providing eight digital comparators
- Converter uses an internal 3-V reference or an external 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

## 12.1 Block Diagram

The Stellaris microcontroller contains two identical Analog-to-Digital Converter modules. These two modules, ADC0 and ADC1, share the same 16 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 12-1 on page 521 shows how the two modules are connected to analog inputs and the system bus.

**Figure 12-1. Implementation of Two ADC Blocks**

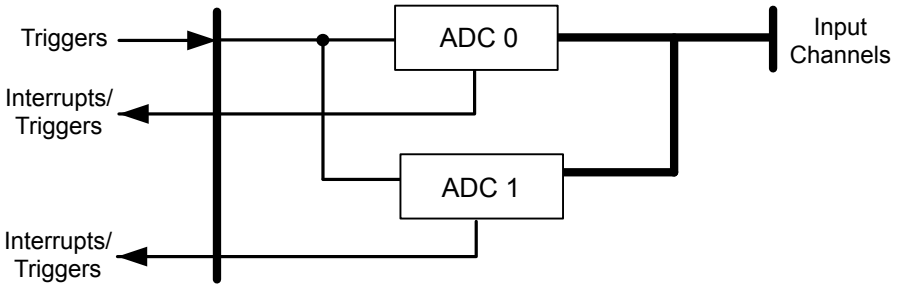
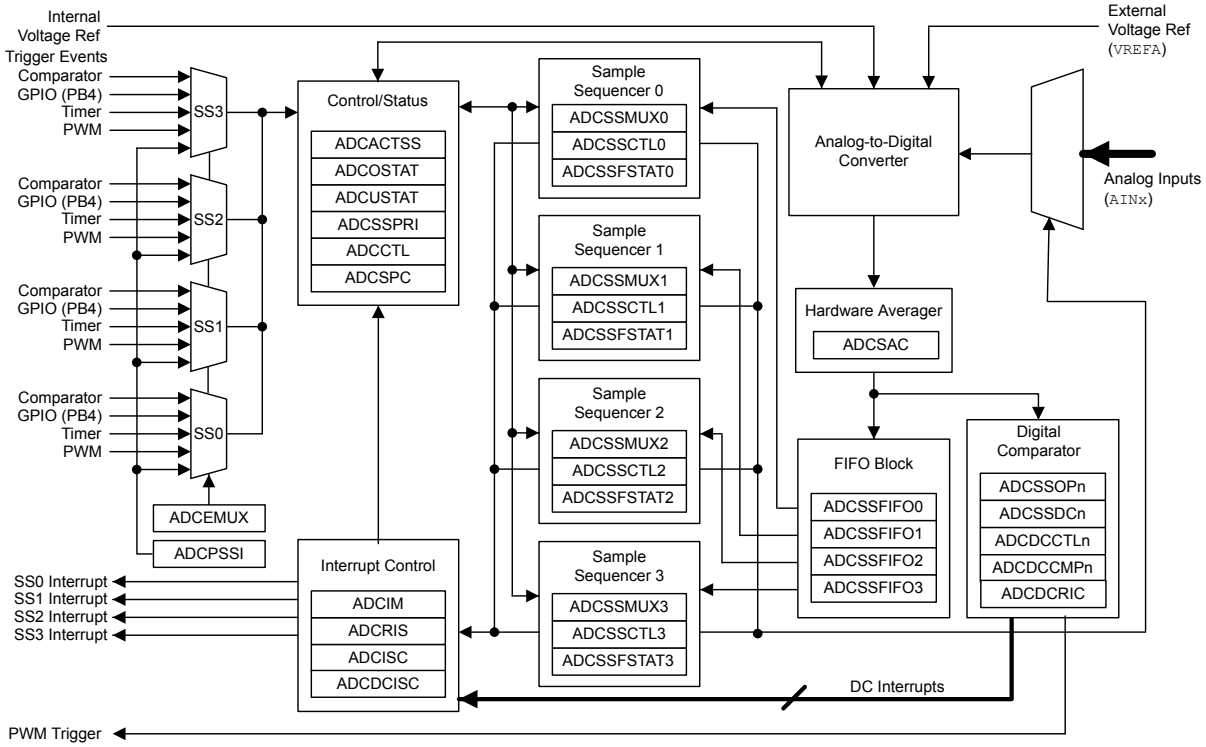



Figure 12-2 on page 521 provides details on the internal configuration of the ADC controls and data registers.

**Figure 12-2. ADC Module Block Diagram**



## 12.2 Signal Description

The following table lists the external signals of the ADC module and describes the function of each. The ADC 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. The AINx and VREFA analog signals are not 5-V tolerant and 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. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

**Table 12-1. Signals for ADC (100LQFP)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| AIN0 | 1 | PE7 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | 2 | PE6 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | 5 | PE5 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | 6 | PE4 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | 100 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | 99 | PD6 | I | Analog | Analog-to-digital converter input 5. |
| AIN6 | 96 | PD5 | I | Analog | Analog-to-digital converter input 6. |
| AIN7 | 95 | PD4 | I | Analog | Analog-to-digital converter input 7. |
| AIN8 | 75 | PE3 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | 74 | PE2 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | 92 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | 91 | PB5 | I | Analog | Analog-to-digital converter input 11. |
| AIN12 | 13 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | 12 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | 11 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | 10 | PD0 | I | Analog | Analog-to-digital converter input 15. |
| VREFA | 90 | PB6 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 12-2. Signals for ADC (108BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| AIN0 | B1 | PE7 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | A1 | PE6 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | B3 | PE5 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | B2 | PE4 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | A2 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | A3 | PD6 | I | Analog | Analog-to-digital converter input 5. |

**Table 12-2. Signals for ADC (108BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| AIN6 | B4 | PD5 | I | Analog | Analog-to-digital converter input 6. |
| AIN7 | A4 | PD4 | I | Analog | Analog-to-digital converter input 7. |
| AIN8 | A12 | PE3 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | B11 | PE2 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | A6 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | B7 | PB5 | I | Analog | Analog-to-digital converter input 11. |
| AIN12 | H1 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | H2 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | G2 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | G1 | PD0 | I | Analog | Analog-to-digital converter input 15. |
| VREFA | A7 | PB6 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 12.3 Functional Description

The Stellaris 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.

### 12.3.1 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 12-3 on page 523 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 12-3. 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)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** 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 561.

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.

### 12.3.2 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

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured for 16-MHz operation automatically by hardware when the system XTAL is selected.

#### 12.3.2.1 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.

#### 12.3.2.2 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 module provides a request signal from each sample sequencer to the associated dedicated channel of the µDMA controller. The ADC does not support single transfer requests. A burst transfer request is asserted when the interrupt bit for the sample sequence is set (IE bit in the **ADCSSCTLn** register is set).

The arbitration size of the µDMA transfer must be a power of 2, and the associated IE bits in the **ADDSSCTLn** register must be set. For example, if the µDMA channel of SS0 has an arbitration size of four, the IE3 bit (4th sample) and the IE7 bit (8th sample) must be set. Thus the µDMA request occurs every time 4 samples have been acquired. No other special steps are needed to enable the ADC module for µDMA operation.

Refer to the "Micro Direct Memory Access (µDMA)" on page 335 for more details about programming the µDMA controller.

#### 12.3.2.3 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.

#### 12.3.2.4 Sampling Events

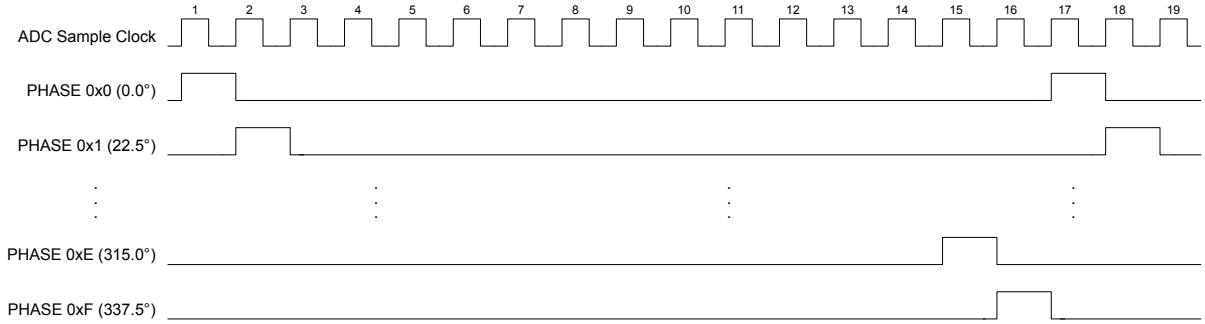
Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), an external signal on GPIO PB4, a GP Timer, and continuous sampling. The processor triggers sampling by setting the SSx bits in the **ADC Processor 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.

#### 12.3.2.5 Sample Phase Control

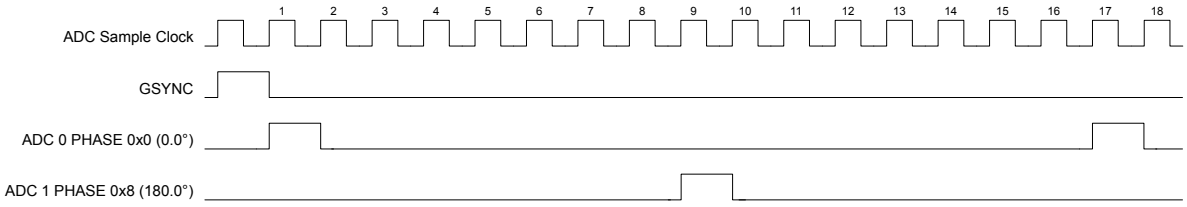
The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. If the converters are running at the same sample rate, they may be configured to start the conversions coincidentally or with one of 15 different discrete phases relative to each other. The sample time can be delayed from the standard sampling time in 22.5° increments up to 337.5º using the **ADC Sample Phase Control (ADCSPC)** register. Figure 12-3 on page 526 shows an example of various phase relationships at a 1 Msps rate.

**Figure 12-3. 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 could sample at the standard position (the PHASE field in the **ADCSPC** register is 0x0). ADC module 1 can be configured to sample at 180 (PHASE = 0x8). The two modules can be be synchronized using the GSYNC and SYNCWAIT bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. Software could then combine the results from the two modules to create a sample rate of two million samples/second at 16 MHz as shown in Figure 12-4 on page 526.

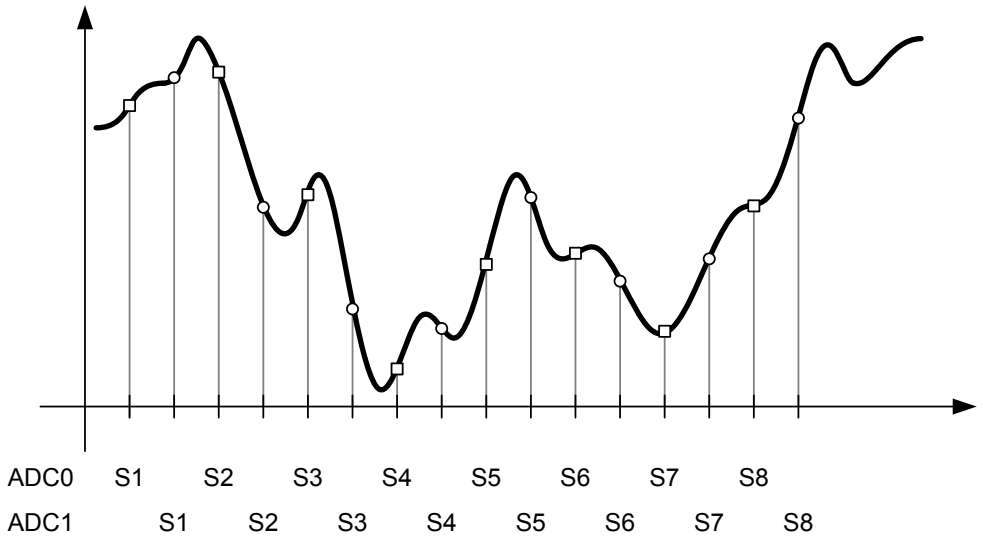
**Figure 12-4. Doubling the ADC Sample Rate**




Using the **ADCSPC** register, ADC0 and ADC1 may provide a number of interesting applications:

- Coincident sampling of different signals. The sample sequence steps run coincidently in both converters.
  - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
  - ADC Module 1, **ADCSPC** = 0x0, sampling AIN1
- Skewed sampling of the same signal. The sample sequence steps are 1/2 of an ADC clock (500 µs for a 1Ms/s ADC) out of phase with each other. This configuration doubles the conversion bandwidth of a single input when software combines the results as shown in Figure 12-5 on page 527.
  - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
  - ADC Module 1, **ADCSPC** = 0x8, sampling AIN0

**Figure 12-5. Skewed Sampling**



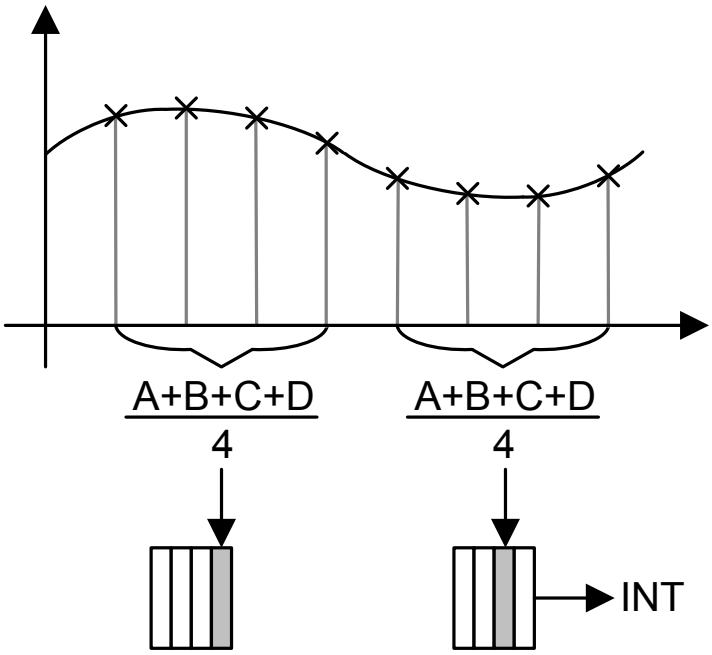

### 12.3.3 Hardware Sample Averaging Circuit

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 563). 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 12-6 on page 528 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 12-6. Sample Averaging Example**




### 12.3.4 Analog-to-Digital Converter

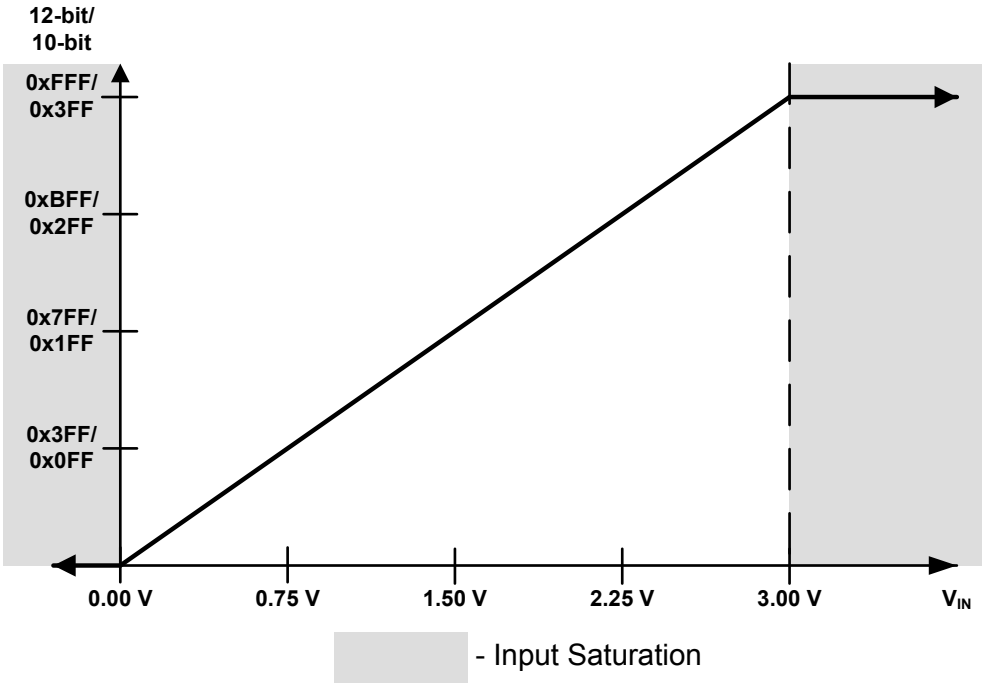
The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 12-bit, low-power, high-precision conversion value. The ADC defaults to a 10-bit conversion result, providing backwards compatibility with previous generations of Stellaris microcontrollers. To enable 12-bit resolution, set the RES bit in the **ADC Control (ADCCTL)** register. The successive-approximation algorithm uses a current mode D/A converter to achieve lower settling time, resulting in higher conversion speeds for the A/D converter. In addition, built-in sample-and-hold circuitry with offset-calibration circuitry improves conversion accuracy. The ADC must be run from the PLL or a 14- to 18-MHz clock source.

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 182). The analog inputs are connected to the ADC through custom pads and specially balanced input paths to minimize the distortion on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in “Analog-to-Digital Converter (ADC)” on page 796.

#### 12.3.4.1 Internal Voltage Reference

The band-gap circuitry generates an internal 3.0 V reference that can be used by the ADC to produce a conversion value from the selected analog input. The range of this conversion value is from 0x000 to 0xFFF in 12-bit mode, or 0x3FF in 10-bit mode. In single-ended-input mode, the 0x000 value corresponds to an analog input voltage of 0.0 V; the 0xFFF in 12-bit mode, or 0x3FF in 10-bit mode value corresponds to an analog input voltage of 3.0 V. This configuration results in a resolution of approximately 0.7 mV in 12-bit mode and 2.9 mV per ADC code in 10-bit mode. While the analog input pads can handle voltages beyond this range, the ADC conversions saturate in under-voltage and over-voltage cases. Figure 12-7 on page 529 shows the ADC conversion function of the analog inputs.

**Figure 12-7. Internal Voltage Conversion Result**




### 12.3.4.2 External Voltage Reference

The ADC can use an external voltage reference to produce the conversion value from the selected analog input by configuring the VREF field in the **ADC Control (ADCCTL)** register. The VREF field specifies whether to use the internal, an external reference in the 3.0 V range, or an external reference in the 1.0 V range. While the range of the conversion value remains the same (0x000 to 0xFFF or 0x3FF), the analog voltage associated with the 0xFFF or 0x3FF value corresponds to the value of the voltage when using the 3.0-V setting and three times the voltage when using the 1.0-V setting, resulting in a smaller voltage resolution per ADC code. Ground is always used as the reference level for the minimum conversion value. Analog input voltages above the external voltage reference saturate to 0xFFF or 0x3FF while those below 0.0 V continue to saturate at 0x000. The $V_{REFA}$ specification defines the useful range for the external voltage reference, see Table 19-22 on page 797. Care must be taken to supply a reference voltage of acceptable quality.

Figure 12-8 on page 530 shows the ADC conversion function of the analog inputs when using anthe 3.0-V setting on the external voltage reference. Figure 12-9 on page 530 shows the ADC conversion function when using the 1.0-V setting on the external voltage reference.

The external voltage reference can be more accurate than the internal reference by using a high-precision source or trimming the source.

**Figure 12-8. External Voltage Conversion Result with 3.0-V Setting**

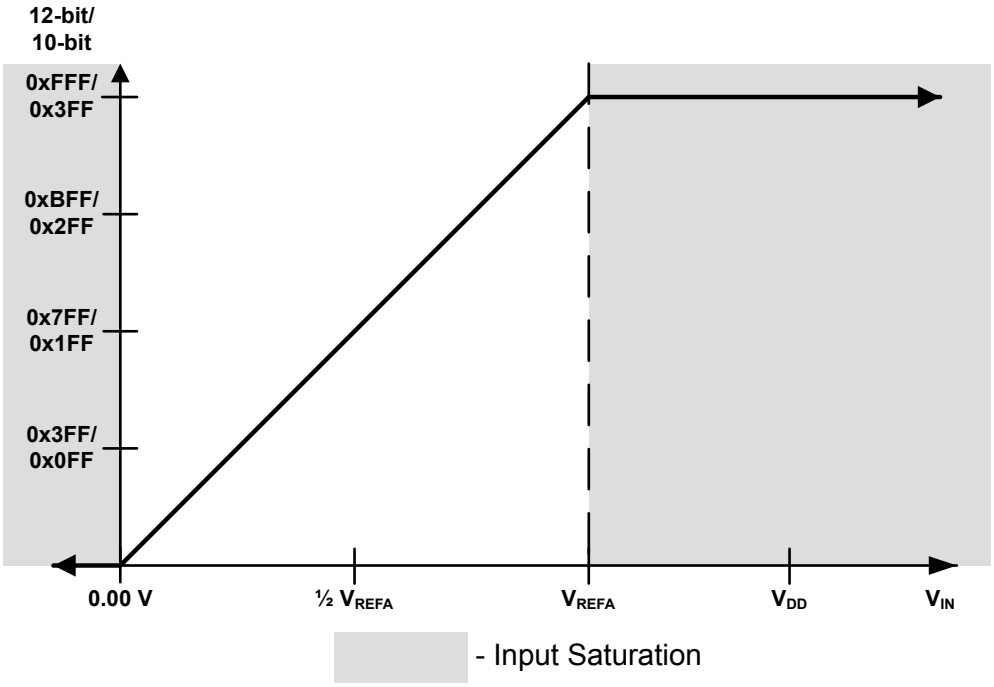
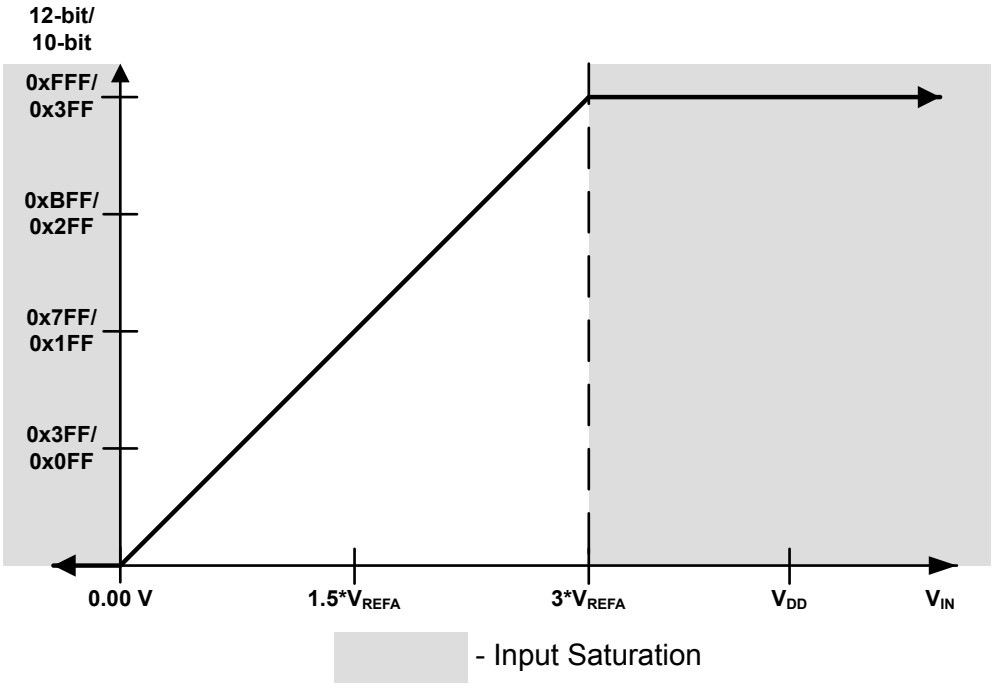



**Figure 12-9. External Voltage Conversion Result with 1.0-V Setting**



### 12.3.5 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 12-4 on page 531). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

**Table 12-4. 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 |

The voltage sampled in differential mode is the difference between the odd and even channels:

$\Delta V$ (differential voltage) = $V_{IN_EVEN}$ (even channel) – $V_{IN_ODD}$ (odd channel), therefore:

- If $\Delta V = 0$, then the conversion result = 0x1FF for 10-bit and 0x7FF for 12-bit
- If $\Delta V > 0$, then the conversion result > 0x1FF (range is 0x1FF–0x3FF) for 10-bit and > 0x7FF (range is 0x7FF - 0xFFF) for 12-bit
- If $\Delta V < 0$, then the conversion result < 0x1FF (range is 0–0x1FF) for 10-bit and < 0x7FF (range is 0 - 0x7FF) for 12-bit

The differential pairs assign polarities to the analog inputs: the even-numbered input is always positive, and the odd-numbered input is always negative. In order for a valid conversion result to appear, the negative input must be in the range of ± 1.5 V of the positive input. If an analog input is greater than 3 V or less than 0 V (the valid range for analog inputs), the input voltage is clipped, meaning it appears as either 3 V or 0 V , respectively, to the ADC.

Figure 12-10 on page 532 shows an example of the negative input centered at 1.5 V. In this configuration, the differential range spans from -1.5 V to 1.5 V. Figure 12-11 on page 532 shows an example where the negative input is centered at 0.75 V, meaning inputs on the positive input saturate past a differential voltage of -0.75 V because the input voltage is less than 0 V. Figure 12-12 on page 533 shows an example of the negative input centered at 2.25 V, where inputs on the positive channel saturate past a differential voltage of 0.75 V since the input voltage would be greater than 3 V.

**Figure 12-10. Differential Sampling Range, $V_{IN_ODD}$ = 1.5 V**

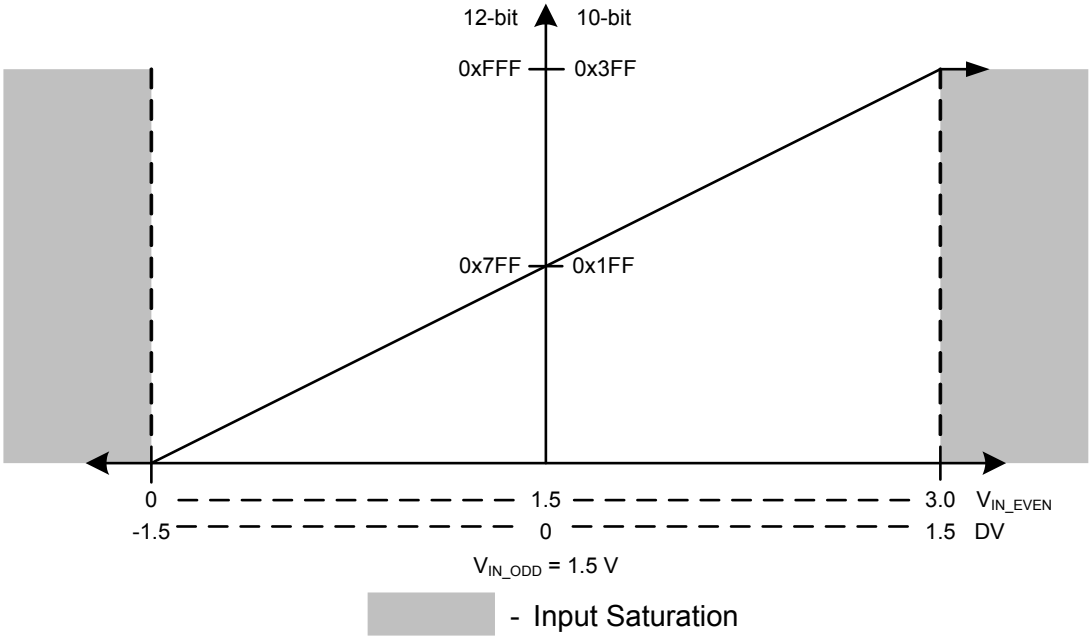



**Figure 12-11. Differential Sampling Range, $V_{IN_ODD}$ = 0.75 V**

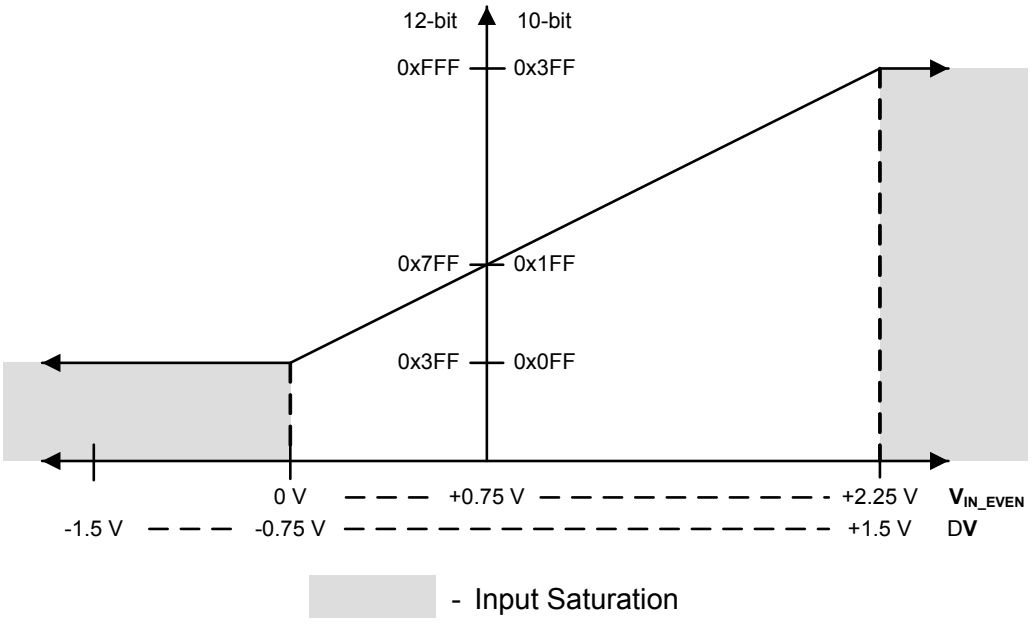
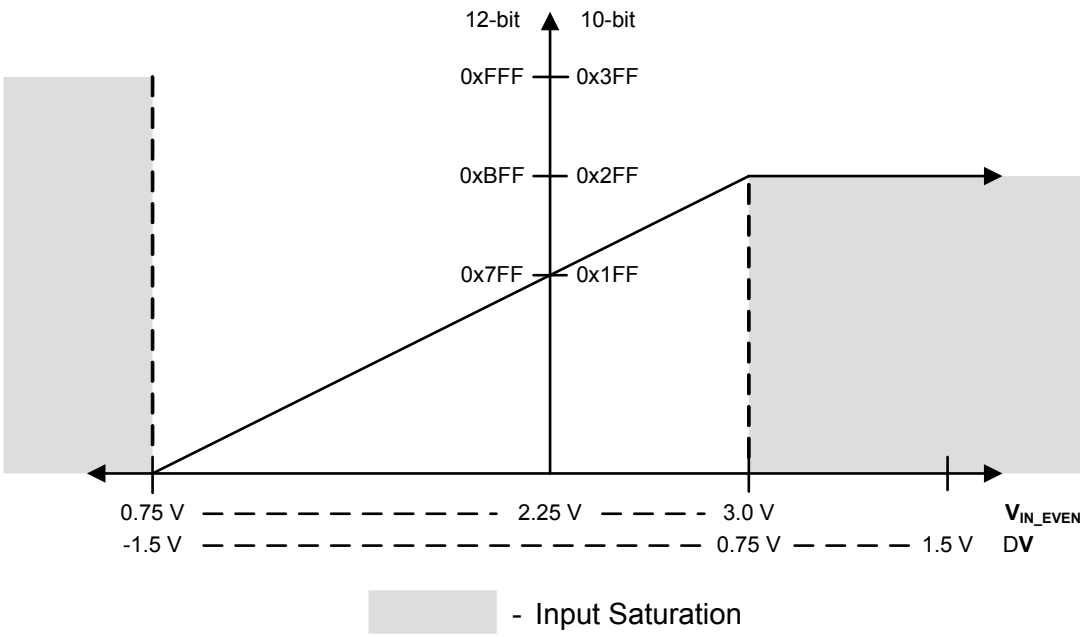


**Figure 12-12. Differential Sampling Range, $V_{IN_ODD}$ = 2.25 V**




## 12.3.6 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.

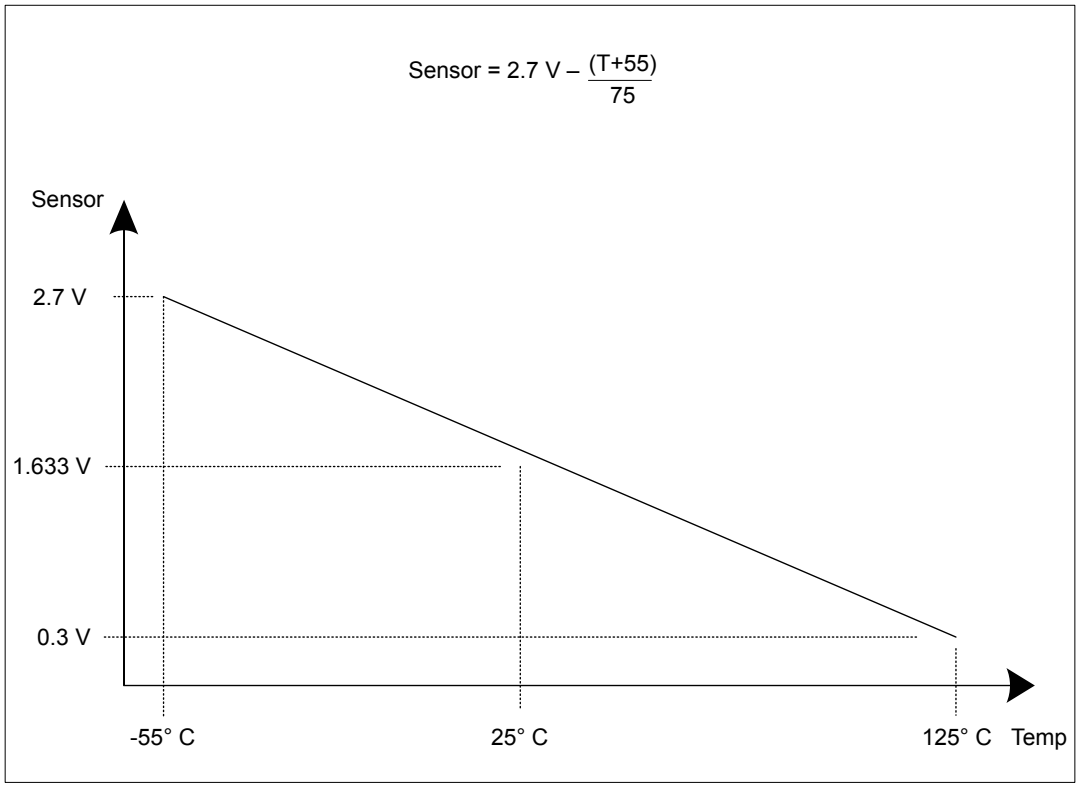
The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC. In addition, the temperature sensor has a second power-down input in the 3.3 V domain which provides control by the Hibernation module.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. This reference voltage, *SENSO*, is given by the following equation:

*SENSO* = 2.7 - ((T + 55) / 75)

This relation is shown in Figure 12-13 on page 534.

**Figure 12-13. 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 temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (in °C) based on the ADC reading:

```
Temperature = 147.5 - ((225 × ADC) / 4095)
```

## 12.3.7 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. If the observed signal moves out of the acceptable range, a processor interrupt can be generated. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be applied to three separate regions (low band, mid band, high band) as defined by the user.

### 12.3.7.1 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 SnDC0P 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.

### 12.3.7.2 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 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.

### 12.3.7.3 Function Ranges

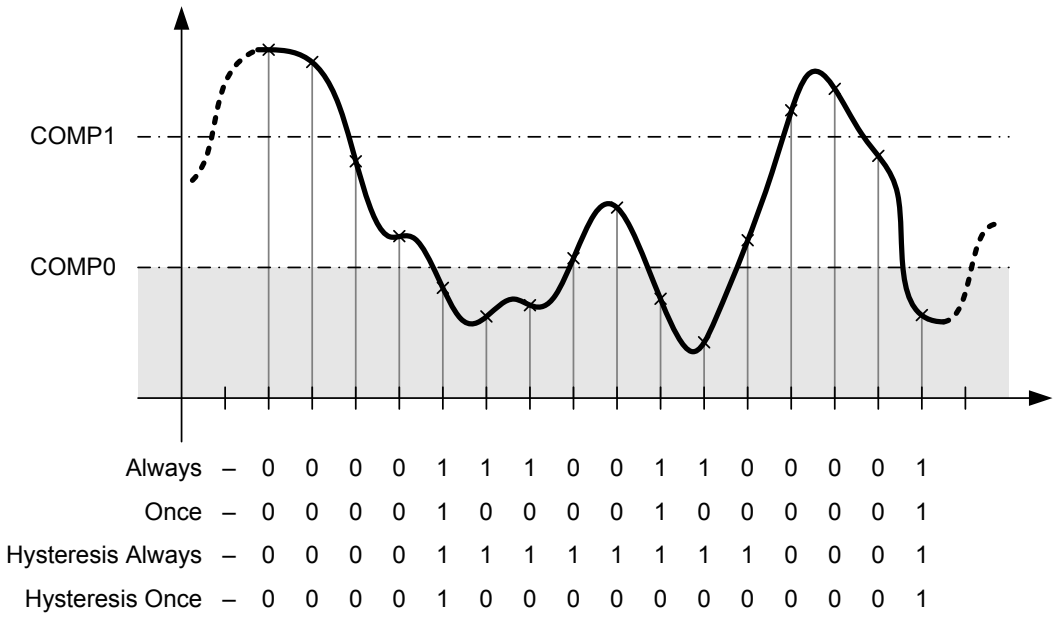
The two comparison values, COMP0 and COMP1, in the **ADC Digital Comparator Range (ADCDCCMPn)** register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than or equal to COMP0), mid-band (greater than COMP0 but less than or equal to COMP1), and high-band (greater than 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, either the CIC field 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 12-14 on page 536. 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 de-asserted and a "1" indicates that the signal is asserted.

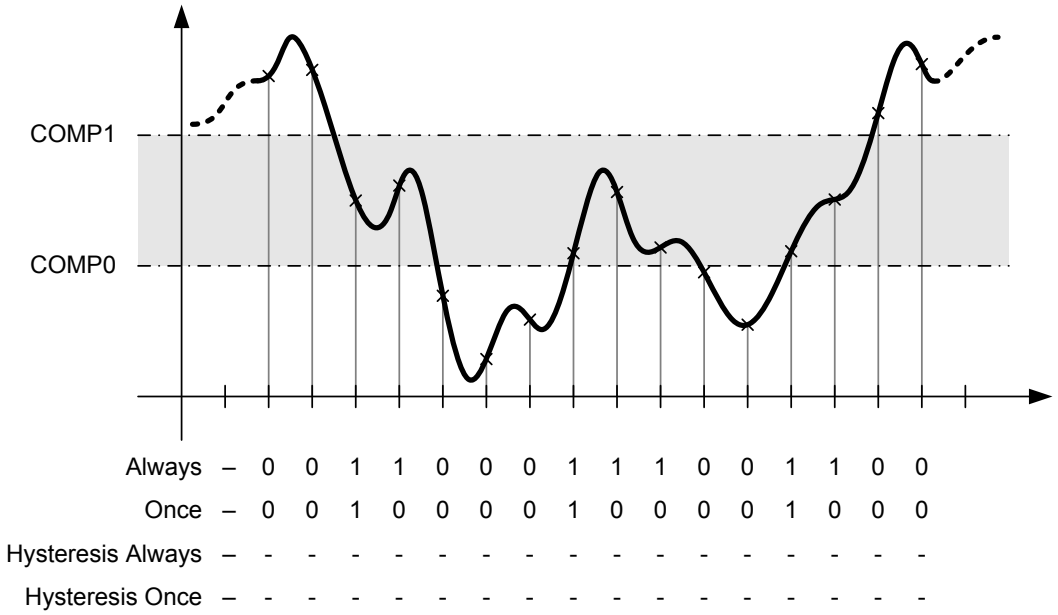
**Figure 12-14. Low-Band Operation (CIC=0x0)**




### *Mid-Band Operation*

To operate in the mid-band region, either the CIC field 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 12-15 on page 537. Note that a "0" in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

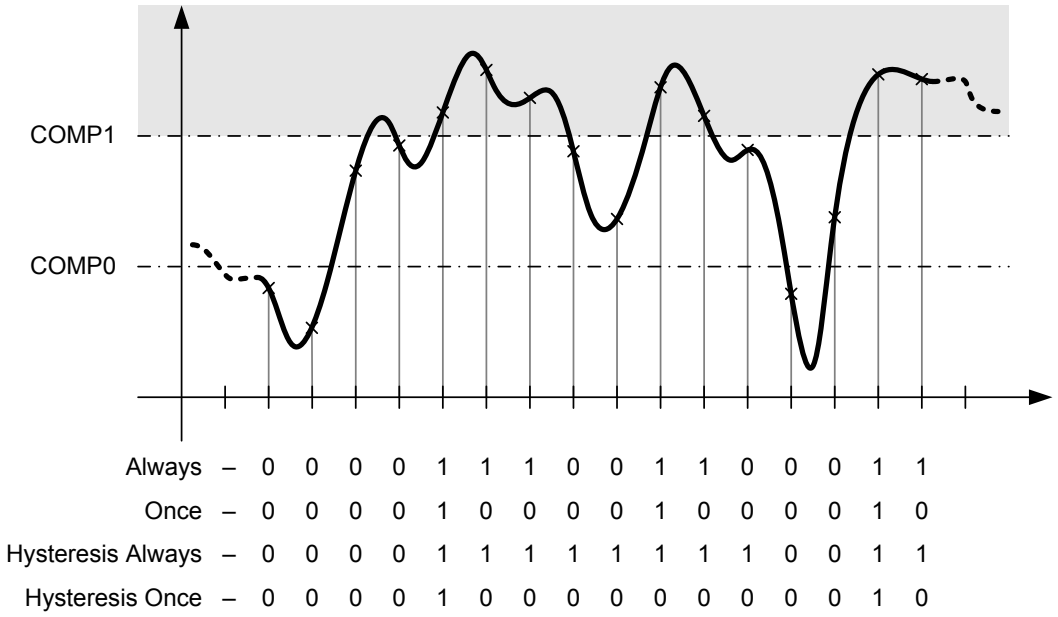
**Figure 12-15. Mid-Band Operation (CIC=0x1)**




### *High-Band Operation*

To operate in the high-band region, either the CIC field 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 12-16 on page 538. 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 de-asserted and a "1" indicates that the signal is asserted.

**Figure 12-16. High-Band Operation (CIC=0x3)**




## 12.4 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and programmed to a supported crystal frequency in the **RCC** register (see page 198). Using unsupported frequencies can cause faulty operation in the ADC module.

### 12.4.1 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 by using the **RCGC0** register (see page 237).
2. Enable the clock to the appropriate GPIO modules via the **RCGC2** register (see page 251). To find out which GPIO ports to enable, refer to “Signal Description” on page 522.
3. Set the GPIO AFSEL bits for the ADC input pins (see page 418). To determine which GPIOs to configure, see Table 17-4 on page 753.
4. Configure the AINx and VREFA signals to be analog inputs by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register (see page 429).
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 434) 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.

### 12.4.2 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. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.
4. 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.
5. If interrupts are to be used, set the corresponding MASK bit in the **ADCIM** register.
6. Enable the sample sequencer logic by setting the corresponding ASENn bit in the **ADCACTSS** register.

## 12.5 Register Map

Table 12-5 on page 539 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 237). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

**Table 12-5. ADC Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | ADCACTSS | R/W | 0x0000.0000 | ADC Active Sample Sequencer | 542 |
| 0x004 | ADCRIS | RO | 0x0000.0000 | ADC Raw Interrupt Status | 543 |
| 0x008 | ADCIM | R/W | 0x0000.0000 | ADC Interrupt Mask | 545 |
| 0x00C | ADCISC | R/W1C | 0x0000.0000 | ADC Interrupt Status and Clear | 547 |
| 0x010 | ADCOSTAT | R/W1C | 0x0000.0000 | ADC Overflow Status | 550 |
| 0x014 | ADCEMUX | R/W | 0x0000.0000 | ADC Event Multiplexer Select | 552 |

**Table 12-5. ADC Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x018 | ADCUSTAT | R/W1C | 0x0000.0000 | ADC Underflow Status | 556 |
| 0x020 | ADCSSPRI | R/W | 0x0000.3210 | ADC Sample Sequencer Priority | 557 |
| 0x024 | ADCSPC | R/W | 0x0000.0000 | ADC Sample Phase Control | 559 |
| 0x028 | ADCPSSI | R/W | - | ADC Processor Sample Sequence Initiate | 561 |
| 0x030 | ADCSAC | R/W | 0x0000.0000 | ADC Sample Averaging Control | 563 |
| 0x034 | ADCDCISC | R/W1C | 0x0000.0000 | ADC Digital Comparator Interrupt Status and Clear | 564 |
| 0x038 | ADCCTL | R/W | 0x0000.0000 | ADC Control | 566 |
| 0x040 | ADCSSMUX0 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 0 | 567 |
| 0x044 | ADCSSCTL0 | R/W | 0x0000.0000 | ADC Sample Sequence Control 0 | 569 |
| 0x048 | ADCSSFIFO0 | RO | - | ADC Sample Sequence Result FIFO 0 | 572 |
| 0x04C | ADCSSFSTAT0 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 0 Status | 573 |
| 0x050 | ADCSSOP0 | R/W | 0x0000.0000 | ADC Sample Sequence 0 Operation | 575 |
| 0x054 | ADCSSDC0 | R/W | 0x0000.0000 | ADC Sample Sequence 0 Digital Comparator Select | 577 |
| 0x060 | ADCSSMUX1 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 1 | 579 |
| 0x064 | ADCSSCTL1 | R/W | 0x0000.0000 | ADC Sample Sequence Control 1 | 580 |
| 0x068 | ADCSSFIFO1 | RO | - | ADC Sample Sequence Result FIFO 1 | 572 |
| 0x06C | ADCSSFSTAT1 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 1 Status | 573 |
| 0x070 | ADCSSOP1 | R/W | 0x0000.0000 | ADC Sample Sequence 1 Operation | 582 |
| 0x074 | ADCSSDC1 | R/W | 0x0000.0000 | ADC Sample Sequence 1 Digital Comparator Select | 583 |
| 0x080 | ADCSSMUX2 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 2 | 579 |
| 0x084 | ADCSSCTL2 | R/W | 0x0000.0000 | ADC Sample Sequence Control 2 | 580 |
| 0x088 | ADCSSFIFO2 | RO | - | ADC Sample Sequence Result FIFO 2 | 572 |
| 0x08C | ADCSSFSTAT2 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 2 Status | 573 |
| 0x090 | ADCSSOP2 | R/W | 0x0000.0000 | ADC Sample Sequence 2 Operation | 582 |
| 0x094 | ADCSSDC2 | R/W | 0x0000.0000 | ADC Sample Sequence 2 Digital Comparator Select | 583 |
| 0x0A0 | ADCSSMUX3 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 3 | 585 |
| 0x0A4 | ADCSSCTL3 | R/W | 0x0000.0002 | ADC Sample Sequence Control 3 | 586 |
| 0x0A8 | ADCSSFIFO3 | RO | - | ADC Sample Sequence Result FIFO 3 | 572 |
| 0x0AC | ADCSSFSTAT3 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 3 Status | 573 |
| 0x0B0 | ADCSSOP3 | R/W | 0x0000.0000 | ADC Sample Sequence 3 Operation | 587 |
| 0x0B4 | ADCSSDC3 | R/W | 0x0000.0000 | ADC Sample Sequence 3 Digital Comparator Select | 588 |
| 0xD00 | ADCDCRIC | R/W | 0x0000.0000 | ADC Digital Comparator Reset Initial Conditions | 589 |

**Table 12-5. ADC Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xE00 | ADCDCCTL0 | R/W | 0x0000.0000 | ADC Digital Comparator Control 0 | 594 |
| 0xE04 | ADCDCCTL1 | R/W | 0x0000.0000 | ADC Digital Comparator Control 1 | 594 |
| 0xE08 | ADCDCCTL2 | R/W | 0x0000.0000 | ADC Digital Comparator Control 2 | 594 |
| 0xE0C | ADCDCCTL3 | R/W | 0x0000.0000 | ADC Digital Comparator Control 3 | 594 |
| 0xE10 | ADCDCCTL4 | R/W | 0x0000.0000 | ADC Digital Comparator Control 4 | 594 |
| 0xE14 | ADCDCCTL5 | R/W | 0x0000.0000 | ADC Digital Comparator Control 5 | 594 |
| 0xE18 | ADCDCCTL6 | R/W | 0x0000.0000 | ADC Digital Comparator Control 6 | 594 |
| 0xE1C | ADCDCCTL7 | R/W | 0x0000.0000 | ADC Digital Comparator Control 7 | 594 |
| 0xE40 | ADCDCCMP0 | R/W | 0x0000.0000 | ADC Digital Comparator Range 0 | 596 |
| 0xE44 | ADCDCCMP1 | R/W | 0x0000.0000 | ADC Digital Comparator Range 1 | 596 |
| 0xE48 | ADCDCCMP2 | R/W | 0x0000.0000 | ADC Digital Comparator Range 2 | 596 |
| 0xE4C | ADCDCCMP3 | R/W | 0x0000.0000 | ADC Digital Comparator Range 3 | 596 |
| 0xE50 | ADCDCCMP4 | R/W | 0x0000.0000 | ADC Digital Comparator Range 4 | 596 |
| 0xE54 | ADCDCCMP5 | R/W | 0x0000.0000 | ADC Digital Comparator Range 5 | 596 |
| 0xE58 | ADCDCCMP6 | R/W | 0x0000.0000 | ADC Digital Comparator Range 6 | 596 |
| 0xE5C | ADCDCCMP7 | R/W | 0x0000.0000 | ADC Digital Comparator Range 7 | 596 |

## 12.6 Register Descriptions

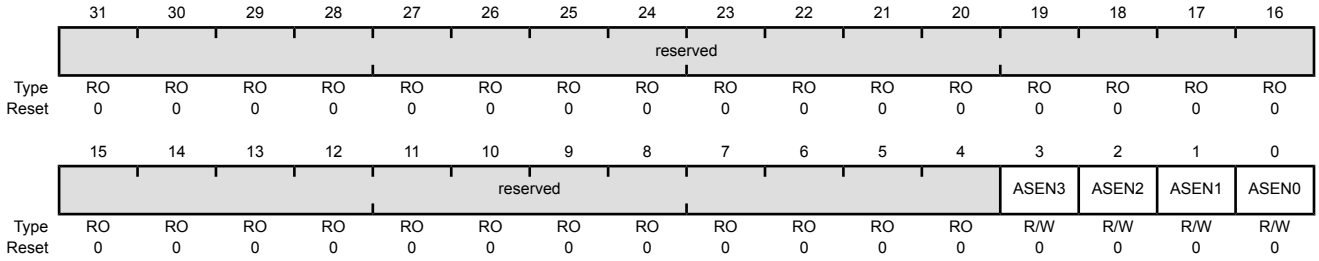
The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

## 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 R/W, reset 0x0000.0000



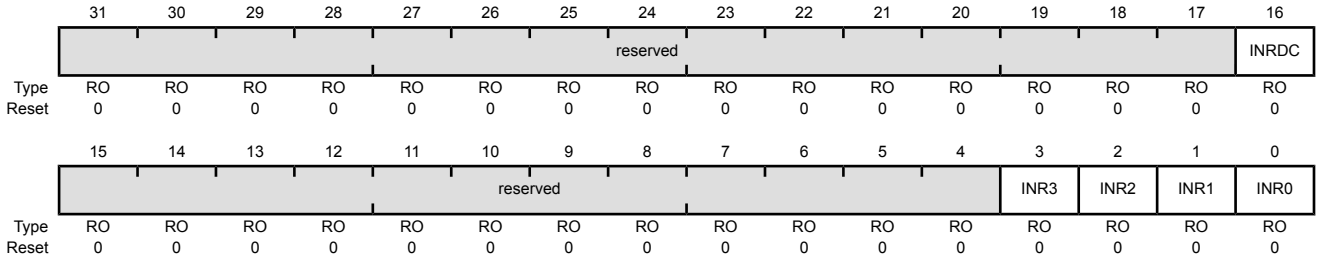

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | ASEN3 | R/W | 0 | ADC SS3 Enable<br>Value Description<br>1 Sample Sequencer 3 is enabled.<br>0 Sample Sequencer 3 is disabled. |
| 2 | ASEN2 | R/W | 0 | ADC SS2 Enable<br>Value Description<br>1 Sample Sequencer 2 is enabled.<br>0 Sample Sequencer 2 is disabled. |
| 1 | ASEN1 | R/W | 0 | ADC SS1 Enable<br>Value Description<br>1 Sample Sequencer 1 is enabled.<br>0 Sample Sequencer 1 is disabled. |
| 0 | ASEN0 | R/W | 0 | ADC SS0 Enable<br>Value Description<br>1 Sample Sequencer 0 is enabled.<br>0 Sample Sequencer 0 is disabled. |

## 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>Value Description<br>1 At least one bit in the **ADCDCISC** register is set, meaning that a digital comparator interrupt has occurred.<br>0 All bits in the **ADCDCISC** register are clear. |
| 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 | INR3 | RO | 0 | SS3 Raw Interrupt Status<br>Value Description<br>1 A sample has completed conversion and the respective **ADCSSCTL3** IEn bit is set, enabling a raw interrupt.<br>0 An interrupt has not occurred.<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>Value Description<br>1 A sample has completed conversion and the respective **ADCSSCTL2** IEn bit is set, enabling a raw interrupt.<br>0 An interrupt has not occurred.<br>This bit is cleared by writing a 1 to the IN2 bit in the **ADCISC** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INR1 | RO | 0 | SS1 Raw Interrupt Status<br><br>Value Description<br>1 A sample has completed conversion and the respective **ADCSSCTL1** IEn bit is set, enabling a raw interrupt.<br>0 An interrupt has not occurred.<br><br>This bit is cleared by writing a 1 to the IN1 bit in the **ADCISC** register. |
| 0 | INR0 | RO | 0 | SS0 Raw Interrupt Status<br><br>Value Description<br>1 A sample has completed conversion and the respective **ADCSSCTL0** IEn bit is set, enabling a raw interrupt.<br>0 An interrupt has not occurred.<br><br>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. 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.

ADC Interrupt Mask (ADCIM)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x008
Type R/W, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | reserved | | | | | | | | | | | | DCONSS3 | DCONSS2 | DCONSS1 | DCONSS0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| 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 | | | | | | | | | | | | MASK3 | MASK2 | MASK1 | MASK0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| 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 | DCONSS3 | R/W | 0 | Digital Comparator Interrupt on SS3<br>Value Description<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.<br>0 The status of the digital comparators does not affect the SS3 interrupt status. |
| 18 | DCONSS2 | R/W | 0 | Digital Comparator Interrupt on SS2<br>Value Description<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.<br>0 The status of the digital comparators does not affect the SS2 interrupt status. |
| 17 | DCONSS1 | R/W | 0 | Digital Comparator Interrupt on SS1<br>Value Description<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.<br>0 The status of the digital comparators does not affect the SS1 interrupt status. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 16 | DCONSS0 | R/W | 0 | Digital Comparator Interrupt on SS0<br><br>Value Description<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.<br>0 The status of the digital comparators does not affect the SS0 interrupt status. |
| 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 | MASK3 | R/W | 0 | SS3 Interrupt Mask<br><br>Value Description<br>1 The raw interrupt signal from Sample Sequencer 3 (**ADCRIS** register INR3 bit) is sent to the interrupt controller.<br>0 The status of Sample Sequencer 3 does not affect the SS3 interrupt status. |
| 2 | MASK2 | R/W | 0 | SS2 Interrupt Mask<br><br>Value Description<br>1 The raw interrupt signal from Sample Sequencer 2 (**ADCRIS** register INR2 bit) is sent to the interrupt controller.<br>0 The status of Sample Sequencer 2 does not affect the SS2 interrupt status. |
| 1 | MASK1 | R/W | 0 | SS1 Interrupt Mask<br><br>Value Description<br>1 The raw interrupt signal from Sample Sequencer 1 (**ADCRIS** register INR1 bit) is sent to the interrupt controller.<br>0 The status of Sample Sequencer 1 does not affect the SS1 interrupt status. |
| 0 | MASK0 | R/W | 0 | SS0 Interrupt Mask<br><br>Value Description<br>1 The raw interrupt signal from Sample Sequencer 0 (**ADCRIS** register INR0 bit) is sent to the interrupt controller.<br>0 The status of Sample Sequencer 0 does not affect the SS0 interrupt status. |

## 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 R/W1C, 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 | | | | | | | | | | | | IN3 | IN2 | IN1 | IN0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C |
| 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>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>0 No interrupt has occurred or the interrupt is masked.<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>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>0 No interrupt has occurred or the interrupt is masked.<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>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>0 No interrupt has occurred or the interrupt is masked.<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>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>0 No interrupt has occurred or the interrupt is masked.<br><br>This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the **ADCRIS** register. |
| 15:4 | 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 | R/W1C | 0 | SS3 Interrupt Status and Clear<br><br>Value Description<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>0 No interrupt has occurred or the interrupt is masked.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit in the **ADCRIS** register. |
| 2 | IN2 | R/W1C | 0 | SS2 Interrupt Status and Clear<br><br>Value Description<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>0 No interrupt has occurred or the interrupt is masked.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit in the **ADCRIS** register. |

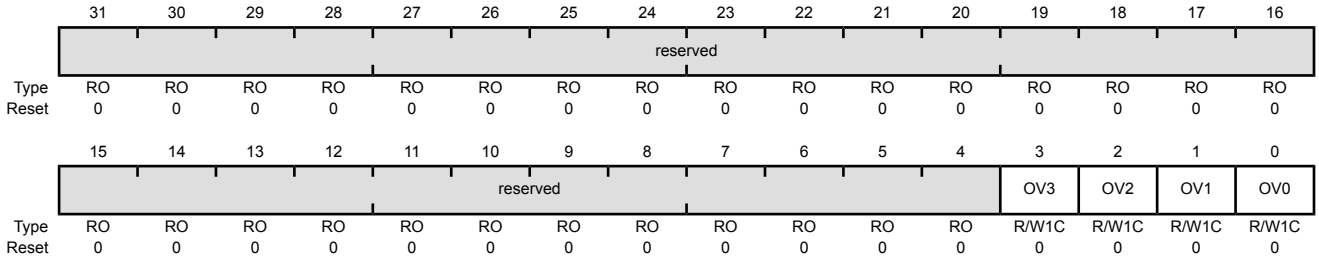
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | IN1 | R/W1C | 0 | SS1 Interrupt Status and Clear<br><br>Value Description<br>1 Both the INR1 bit in the **ADCRIS** register and the MASK1 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.<br>0 No interrupt has occurred or the interrupt is masked.<br><br>This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit in the **ADCRIS** register. |
| 0 | IN0 | R/W1C | 0 | SS0 Interrupt Status and Clear<br><br>Value Description<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>0 No interrupt has occurred or the interrupt is masked.<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 R/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 | OV3 | R/W1C | 0 | SS3 FIFO Overflow<br><br>Value Description<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>0 The FIFO has not overflowed.<br><br>This bit is cleared by writing a 1. |
| 2 | OV2 | R/W1C | 0 | SS2 FIFO Overflow<br><br>Value Description<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>0 The FIFO has not overflowed.<br><br>This bit is cleared by writing a 1. |
| 1 | OV1 | R/W1C | 0 | SS1 FIFO Overflow<br><br>Value Description<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>0 The FIFO has not overflowed.<br><br>This bit is cleared by writing a 1. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | OV0 | R/W1C | 0 | SS0 FIFO Overflow |

| Value | Description |
|---|---|
| 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. |
| 0 | The FIFO has not overflowed. |

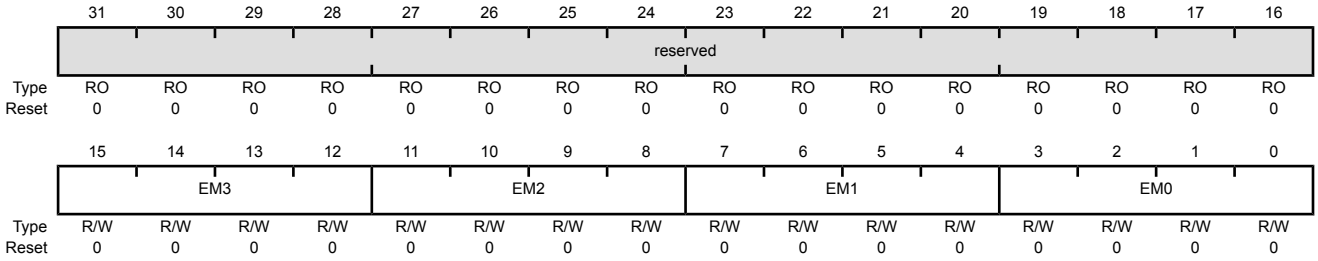
This bit is cleared by writing a 1.

## 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 R/W, 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 | EM3 | R/W | 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 | reserved |
| 0x2 | reserved |
| 0x3 | reserved |
| 0x4 | External (GPIO PB4)<br>This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403).<br>**Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input. |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 470). |
| 0x6 | reserved |
| 0x7 | reserved |
| 0x8 | reserved |
| 0x9 | reserved |
| 0xA-0xE | reserved |
| 0xF | Always (continuously sample) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:8 | EM2 | R/W | 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 | reserved |
| 0x2 | reserved |
| 0x3 | reserved |
| 0x4 | External (GPIO PB4)<br>This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403).<br>**Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input. |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 470). |
| 0x6 | reserved |
| 0x7 | reserved |
| 0x8 | reserved |
| 0x9 | reserved |
| 0xA-0xE | reserved |
| 0xF | Always (continuously sample) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:4 | EM1 | R/W | 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 | reserved |
| 0x2 | reserved |
| 0x3 | reserved |
| 0x4 | External (GPIO PB4)<br>This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403).<br>**Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input. |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 470). |
| 0x6 | reserved |
| 0x7 | reserved |
| 0x8 | reserved |
| 0x9 | reserved |
| 0xA-0xE | reserved |
| 0xF | Always (continuously sample) |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | EM0 | R/W | 0x0 | SS0 Trigger Select<br>This field selects the trigger source for Sample Sequencer 0<br>The valid configurations for this field are: |

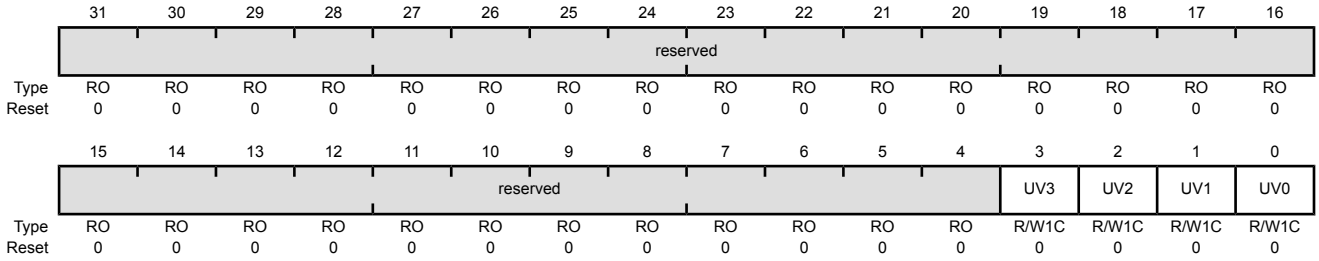
| Value | Event |
|---|---|
| 0x0 | Processor (default)<br>The trigger is initiated by setting the SSn bit in the **ADCPSSI** register. |
| 0x1 | reserved |
| 0x2 | reserved |
| 0x3 | reserved |
| 0x4 | External (GPIO PB4)<br>This trigger is connected to the GPIO interrupt for PB4 (see "ADC Trigger Source" on page 403).<br>**Note:** PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input. |
| 0x5 | Timer<br>In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 470). |
| 0x6 | reserved |
| 0x7 | reserved |
| 0x8 | reserved |
| 0x9 | reserved |
| 0xA-0xE | reserved |
| 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 R/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 | UV3 | R/W1C | 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>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.<br>0 The FIFO has not underflowed. |
| 2 | UV2 | R/W1C | 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 | R/W1C | 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 | R/W1C | 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 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 R/W, 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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. |

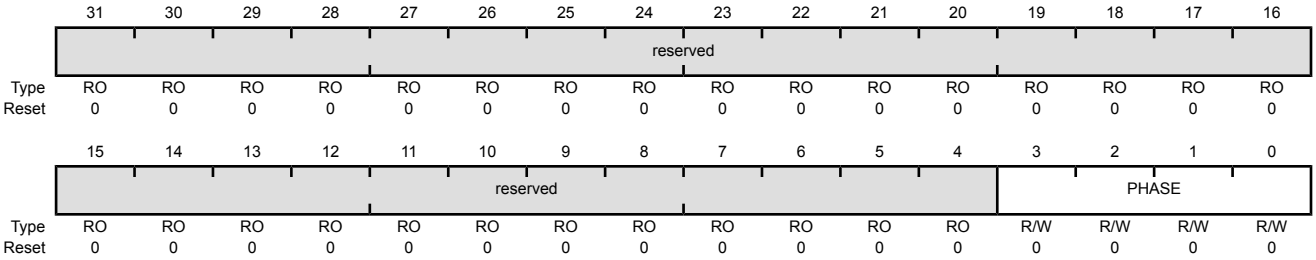
## Register 9: ADC Sample Phase Control (ADCSPC), offset 0x024

This register allows the ADC module to sample at one of 16 different discrete phases from 0.0° through 337.5°. For example, the sample rate could be effectively doubled by sampling a signal using one ADC module configured with the standard sample time and the second ADC module configured with a 180.0° phase lag.

**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 R/W, 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | PHASE | R/W | 0x0 | Phase Difference<br>This field selects the sample phase difference from the standard sample time. |

| Value | Description |
|---|---|
| 0x0 | ADC sample lags by 0.0° |
| 0x1 | ADC sample lags by 22.5° |
| 0x2 | ADC sample lags by 45.0° |
| 0x3 | ADC sample lags by 67.5° |
| 0x4 | ADC sample lags by 90.0° |
| 0x5 | ADC sample lags by 112.5° |
| 0x6 | ADC sample lags by 135.0° |
| 0x7 | ADC sample lags by 157.5° |
| 0x8 | ADC sample lags by 180.0° |
| 0x9 | ADC sample lags by 202.5° |
| 0xA | ADC sample lags by 225.0° |
| 0xB | ADC sample lags by 247.5° |
| 0xC | ADC sample lags by 270.0° |
| 0xD | ADC sample lags by 292.5° |
| 0xE | ADC sample lags by 315.0° |
| 0xF | ADC sample lags by 337.5° |

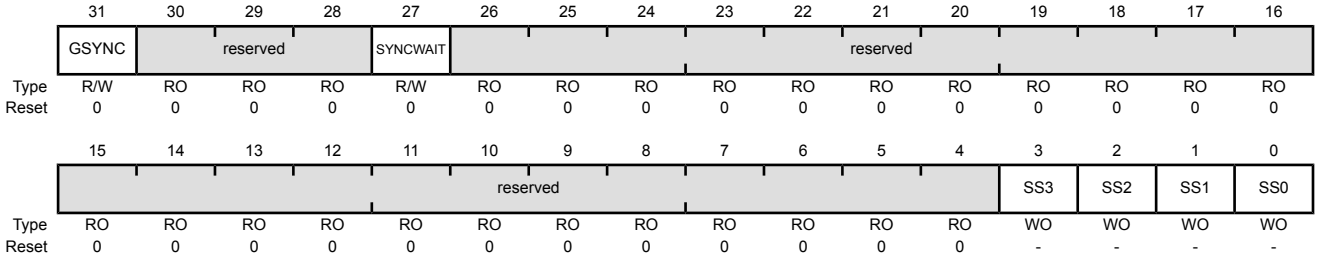
## Register 10: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

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 R/W, reset -




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GSYNC | R/W | 0 | Global Synchronize<br>Value Description<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.<br>0 This bit is cleared once sampling has been initiated. |
| 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 | R/W | 0 | Synchronize Wait<br>Value Description<br>1 This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set.<br>0 Sampling begins when a sample sequence has been initiated. |
| 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. |

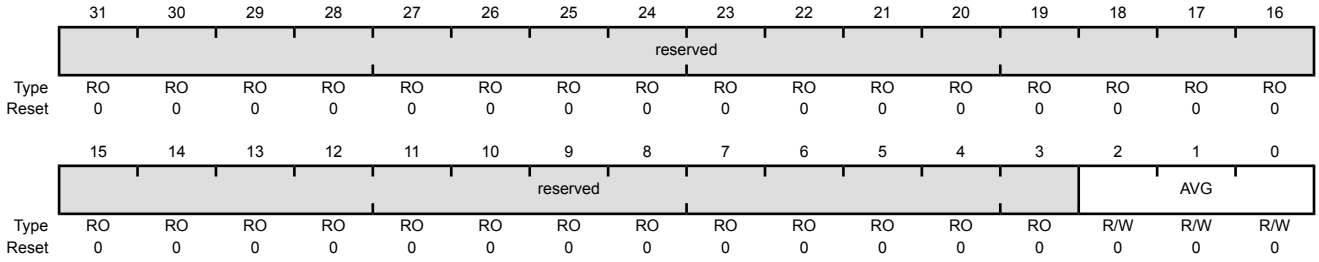
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | SS3 | WO | - | SS3 Initiate<br><br>Value Description<br>1 Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the **ADCACTSS** register.<br>0 No effect.<br><br>Only a write by software is valid; a read of this register returns no meaningful data. |
| 2 | SS2 | WO | - | SS2 Initiate<br><br>Value Description<br>1 Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the **ADCACTSS** register.<br>0 No effect.<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>1 Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the **ADCACTSS** register.<br>0 No effect.<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>1 Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the **ADCACTSS** register.<br>0 No effect.<br><br>Only a write by software is valid; a read of this register returns no meaningful data. |

## Register 11: 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 R/W, 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 | R/W | 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. |

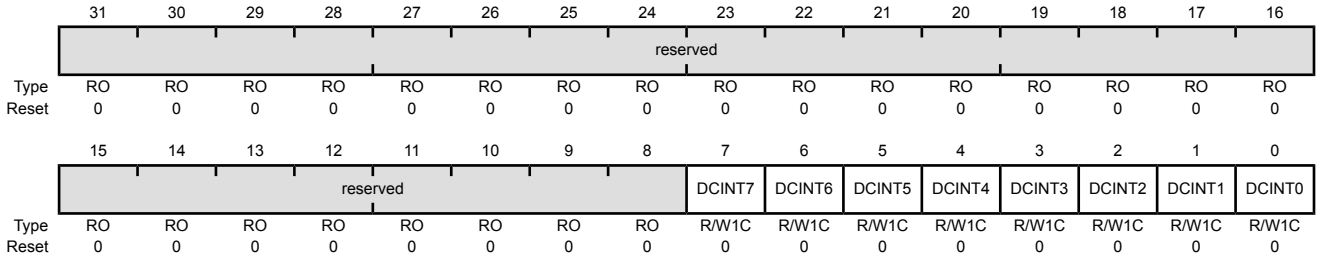
| Value | Description |
|---|---|
| 0x0 | No hardware oversampling |
| 0x1 | 2x hardware oversampling |
| 0x2 | 4x hardware oversampling |
| 0x3 | 8x hardware oversampling |
| 0x4 | 16x hardware oversampling |
| 0x5 | 32x hardware oversampling |
| 0x6 | 64x hardware oversampling |
| 0x7 | reserved |

## Register 12: 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 R/W1C, 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 | R/W1C | 0 | Digital Comparator 7 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 7 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |
| 6 | DCINT6 | R/W1C | 0 | Digital Comparator 6 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 6 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |
| 5 | DCINT5 | R/W1C | 0 | Digital Comparator 5 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 5 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |

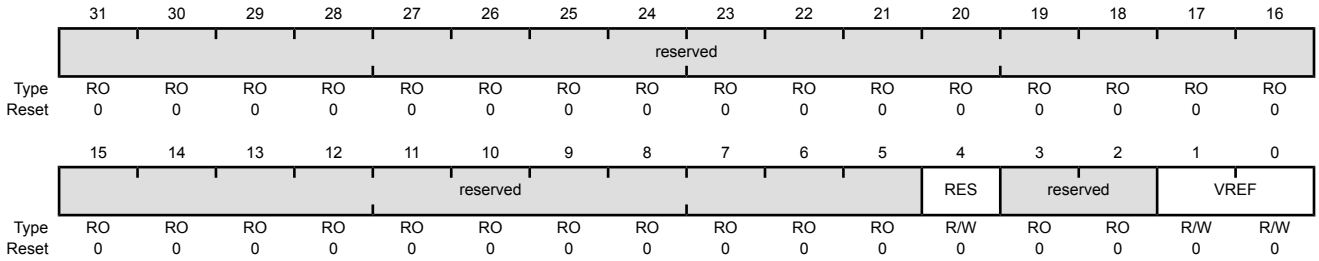
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | DCINT4 | R/W1C | 0 | Digital Comparator 4 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 4 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |
| 3 | DCINT3 | R/W1C | 0 | Digital Comparator 3 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 3 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |
| 2 | DCINT2 | R/W1C | 0 | Digital Comparator 2 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 2 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |
| 1 | DCINT1 | R/W1C | 0 | Digital Comparator 1 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 1 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |
| 0 | DCINT0 | R/W1C | 0 | Digital Comparator 0 Interrupt Status and Clear<br><br>Value Description<br>1 Digital Comparator 0 has generated an interrupt.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1. |

## Register 13: ADC Control (ADCCTL), offset 0x038

This register configures various ADC module attributes, including the ADC resolution and the voltage reference. The resolution of the ADC defaults to 10-bit for backwards compatibility with other members of the Stellaris family, but can be configured to 12-bit resolution. The voltage reference for the conversion can be the internal 3.0-V reference, an external voltage reference in the range of 2.4 V to 3.06 V, or an external voltage reference in the range of 0.8 V to 1.02 V.

ADC Control (ADCCTL)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x038
Type R/W, reset 0x0000.0000



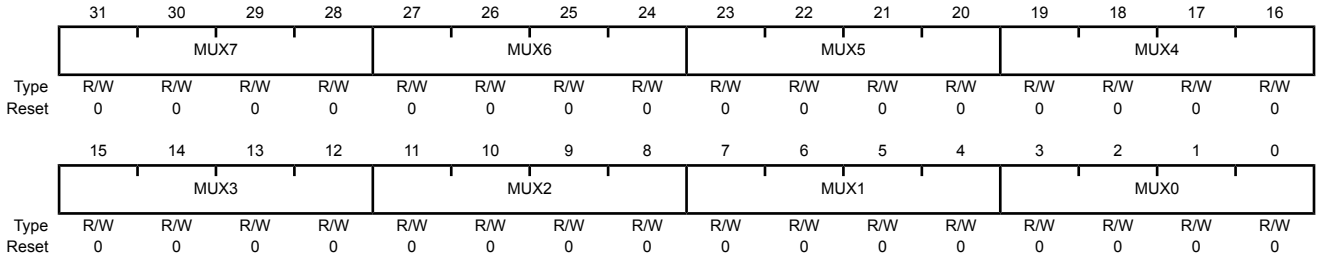

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | RES | R/W | 0 | Sample Resolution<br><br>Value Description<br>1 The ADC returns 12-bit data to the FIFOs.<br>0 The ADC returns 10-bit data to the FIFOs. |
| 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 | VREF | R/W | 0x0 | Voltage Reference Select<br><br>Value Description<br>0x0 Internal Reference<br>The internal reference as the voltage reference. The conversion range is from 0 V to 3.0 V.<br>0x1 3.0 V External Reference<br>A 3.0 V external VREFA input is the voltage reference. The ADC conversion range is 0.0 V to the voltage of the VREFA input.<br>0x2 Reserved<br>0x3 1.0 V External Reference<br>A 1.0 V external VREFA input is the voltage reference. The ADC conversion range is 0.0 V to three times the voltage of the VREFA input. |

## Register 14: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x040
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | MUX7 | R/W | 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 indicates the input is AIN1. |
| 27:24 | MUX6 | R/W | 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 | R/W | 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 | R/W | 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. |
| 15:12 | MUX3 | R/W | 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 | R/W | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:4 | MUX1 | R/W | 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 | R/W | 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 15: 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 R/W, 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 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | 0 | 8th Sample Temp Sensor Select |

| Value | Description |
|---|---|
| 1 | The temperature sensor is read during the eighth sample of the sample sequence. |
| 0 | The input pin specified by the **ADCSSMUXn** register is read during the eighth sample of the sample sequence. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 30 | IE7 | R/W | 0 | 8th Sample Interrupt Enable |

| Value | Description |
|---|---|
| 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. |
| 0 | The raw interrupt is not asserted to the interrupt controller. |

It is legal to have multiple samples within a sequence generate interrupts.

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 29 | END7 | R/W | 0 | 8th Sample is End of Sequence |

| Value | Description |
|---|---|
| 1 | The eighth sample is the last sample of the sequence. |
| 0 | Another sample in the sequence is the final sample. |

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 | R/W | 0 | 8th Sample Diff Input Select<br><br>Value Description<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>0 The analog inputs are not differentially sampled.<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 | R/W | 0 | 7th Sample Temp Sensor Select<br>Same definition as TS7 but used during the seventh sample. |
| 26 | IE6 | R/W | 0 | 7th Sample Interrupt Enable<br>Same definition as IE7 but used during the seventh sample. |
| 25 | END6 | R/W | 0 | 7th Sample is End of Sequence<br>Same definition as END7 but used during the seventh sample. |
| 24 | D6 | R/W | 0 | 7th Sample Diff Input Select<br>Same definition as D7 but used during the seventh sample. |
| 23 | TS5 | R/W | 0 | 6th Sample Temp Sensor Select<br>Same definition as TS7 but used during the sixth sample. |
| 22 | IE5 | R/W | 0 | 6th Sample Interrupt Enable<br>Same definition as IE7 but used during the sixth sample. |
| 21 | END5 | R/W | 0 | 6th Sample is End of Sequence<br>Same definition as END7 but used during the sixth sample. |
| 20 | D5 | R/W | 0 | 6th Sample Diff Input Select<br>Same definition as D7 but used during the sixth sample. |
| 19 | TS4 | R/W | 0 | 5th Sample Temp Sensor Select<br>Same definition as TS7 but used during the fifth sample. |
| 18 | IE4 | R/W | 0 | 5th Sample Interrupt Enable<br>Same definition as IE7 but used during the fifth sample. |
| 17 | END4 | R/W | 0 | 5th Sample is End of Sequence<br>Same definition as END7 but used during the fifth sample. |
| 16 | D4 | R/W | 0 | 5th Sample Diff Input Select<br>Same definition as D7 but used during the fifth sample. |
| 15 | TS3 | R/W | 0 | 4th Sample Temp Sensor Select<br>Same definition as TS7 but used during the fourth sample. |
| 14 | IE3 | R/W | 0 | 4th Sample Interrupt Enable<br>Same definition as IE7 but used during the fourth sample. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 13 | END3 | R/W | 0 | 4th Sample is End of Sequence<br>Same definition as END7 but used during the fourth sample. |
| 12 | D3 | R/W | 0 | 4th Sample Diff Input Select<br>Same definition as D7 but used during the fourth sample. |
| 11 | TS2 | R/W | 0 | 3rd Sample Temp Sensor Select<br>Same definition as TS7 but used during the third sample. |
| 10 | IE2 | R/W | 0 | 3rd Sample Interrupt Enable<br>Same definition as IE7 but used during the third sample. |
| 9 | END2 | R/W | 0 | 3rd Sample is End of Sequence<br>Same definition as END7 but used during the third sample. |
| 8 | D2 | R/W | 0 | 3rd Sample Diff Input Select<br>Same definition as D7 but used during the third sample. |
| 7 | TS1 | R/W | 0 | 2nd Sample Temp Sensor Select<br>Same definition as TS7 but used during the second sample. |
| 6 | IE1 | R/W | 0 | 2nd Sample Interrupt Enable<br>Same definition as IE7 but used during the second sample. |
| 5 | END1 | R/W | 0 | 2nd Sample is End of Sequence<br>Same definition as END7 but used during the second sample. |
| 4 | D1 | R/W | 0 | 2nd Sample Diff Input Select<br>Same definition as D7 but used during the second sample. |
| 3 | TS0 | R/W | 0 | 1st Sample Temp Sensor Select<br>Same definition as TS7 but used during the first sample. |
| 2 | IE0 | R/W | 0 | 1st Sample Interrupt Enable<br>Same definition as IE7 but used during the first sample. |
| 1 | END0 | R/W | 0 | 1st Sample is End of Sequence<br>Same definition as END7 but used during the first sample. |
| 0 | D0 | R/W | 0 | 1st Sample Diff Input Select<br>Same definition as D7 but used during the first sample. |

## Register 16: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

## Register 17: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

## Register 18: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

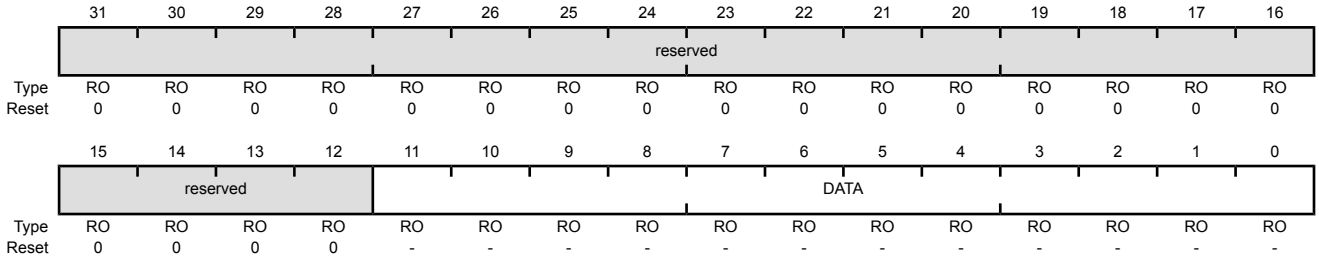
## Register 19: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

**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 20: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

## Register 21: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

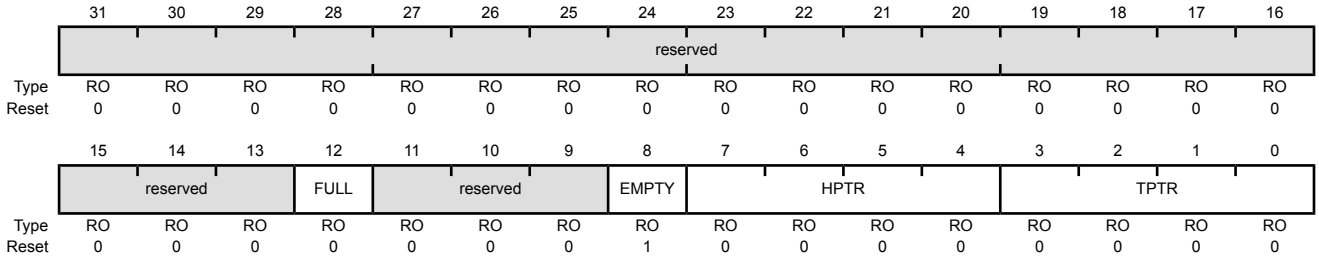
## Register 22: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

## Register 23: 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 0 Status (ADCSSFSTAT0)

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>1 The FIFO is currently full.<br>0 The FIFO is not 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>1 The FIFO is currently empty.<br>0 The FIFO is not 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 24: 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 R/W, 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 | R/W | 0 | Sample 7 Digital Comparator Operation<br>Value Description<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.<br>0 The eighth sample is saved in Sample Sequence FIFO0. |
| 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 | R/W | 0 | Sample 6 Digital Comparator Operation<br>Same definition as S7DCOP 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 | R/W | 0 | Sample 5 Digital Comparator Operation<br>Same definition as S7DCOP 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 | R/W | 0 | Sample 4 Digital Comparator Operation<br>Same definition as S7DCOP 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. |

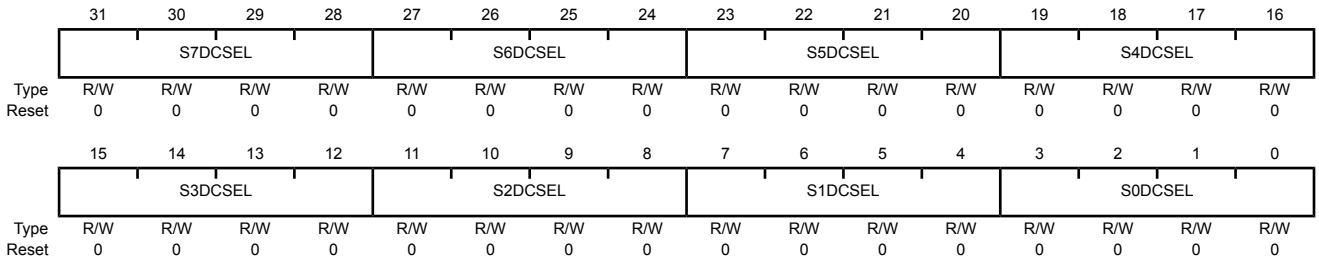
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12 | S3DCOP | R/W | 0 | Sample 3 Digital Comparator Operation<br>Same definition as S7DCOP but used during the fourth sample. |
| 11:9 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | S2DCOP | R/W | 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 | R/W | 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 | R/W | 0 | Sample 0 Digital Comparator Operation<br>Same definition as S7DCOP but used during the first sample. |

## Register 25: 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 R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:28 | S7DCSEL | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 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 | R/W | 0x0 | Sample 0 Digital Comparator Select<br>This field has the same encodings as S7DCSEL but is used during the first sample. |

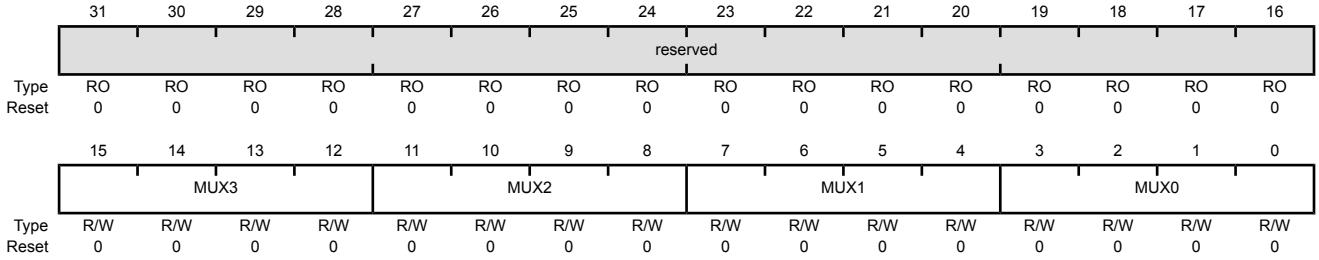
## Register 26: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

## Register 27: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16 bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 567 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x060
Type R/W, 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 | R/W | 0x0 | 4th Sample Input Select |
| 11:8 | MUX2 | R/W | 0x0 | 3rd Sample Input Select |
| 7:4 | MUX1 | R/W | 0x0 | 2nd Sample Input Select |
| 3:0 | MUX0 | R/W | 0x0 | 1st Sample Input Select |

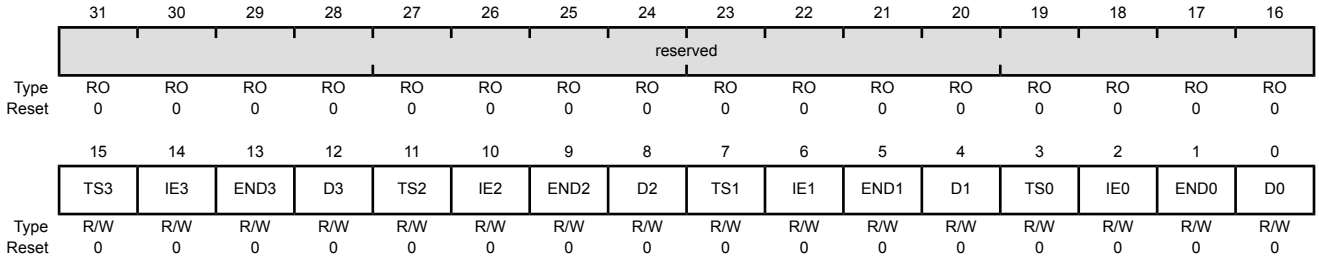
## Register 28: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

## Register 29: 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 569 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control 1 (ADCSSCTL1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x064
Type R/W, 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 | R/W | 0 | 4th Sample Temp Sensor Select<br>Same definition as TS7 but used during the fourth sample. |
| 14 | IE3 | R/W | 0 | 4th Sample Interrupt Enable<br>Same definition as IE7 but used during the fourth sample. |
| 13 | END3 | R/W | 0 | 4th Sample is End of Sequence<br>Same definition as END7 but used during the fourth sample. |
| 12 | D3 | R/W | 0 | 4th Sample Diff Input Select<br>Same definition as D7 but used during the fourth sample. |
| 11 | TS2 | R/W | 0 | 3rd Sample Temp Sensor Select<br>Same definition as TS7 but used during the third sample. |
| 10 | IE2 | R/W | 0 | 3rd Sample Interrupt Enable<br>Same definition as IE7 but used during the third sample. |
| 9 | END2 | R/W | 0 | 3rd Sample is End of Sequence<br>Same definition as END7 but used during the third sample. |
| 8 | D2 | R/W | 0 | 3rd Sample Diff Input Select<br>Same definition as D7 but used during the third sample. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | TS1 | R/W | 0 | 2nd Sample Temp Sensor Select<br>Same definition as TS7 but used during the second sample. |
| 6 | IE1 | R/W | 0 | 2nd Sample Interrupt Enable<br>Same definition as IE7 but used during the second sample. |
| 5 | END1 | R/W | 0 | 2nd Sample is End of Sequence<br>Same definition as END7 but used during the second sample. |
| 4 | D1 | R/W | 0 | 2nd Sample Diff Input Select<br>Same definition as D7 but used during the second sample. |
| 3 | TS0 | R/W | 0 | 1st Sample Temp Sensor Select<br>Same definition as TS7 but used during the first sample. |
| 2 | IE0 | R/W | 0 | 1st Sample Interrupt Enable<br>Same definition as IE7 but used during the first sample. |
| 1 | END0 | R/W | 0 | 1st Sample is End of Sequence<br>Same definition as END7 but used during the first sample. |
| 0 | D0 | R/W | 0 | 1st Sample Diff Input Select<br>Same definition as D7 but used during the first sample. |

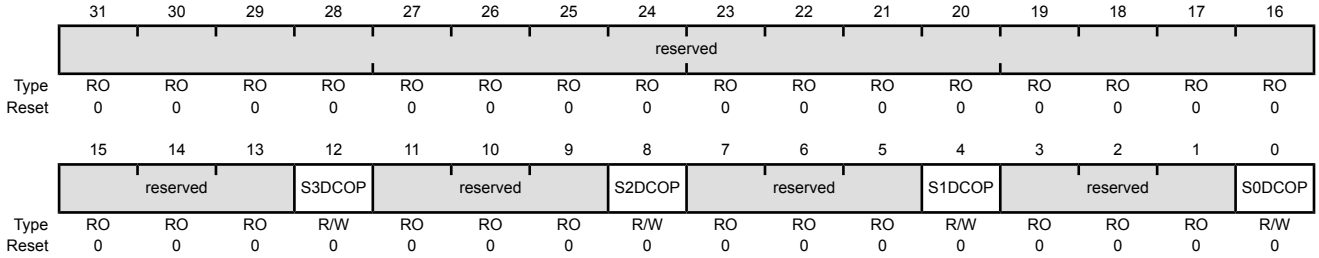
## Register 30: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070

## Register 31: 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 1 Operation (ADCSSOP1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x070
Type R/W, 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 | R/W | 0 | Sample 3 Digital Comparator Operation<br>Value Description<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.<br>0 The fourth sample is saved in Sample Sequence FIFOn. |
| 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 | R/W | 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 | R/W | 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 | R/W | 0 | Sample 0 Digital Comparator Operation<br>Same definition as S3DCOP but used during the first sample. |

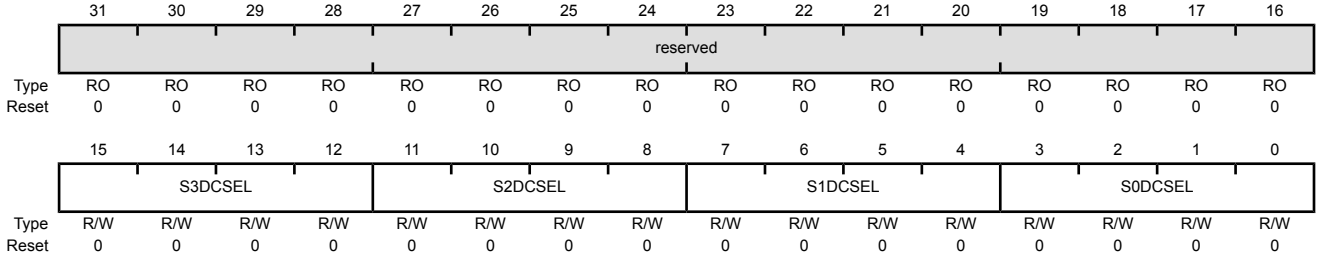
## Register 32: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074

## Register 33: 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 1 Digital Comparator Select (ADCSSDC1)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x074
Type R/W, 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 | R/W | 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 | R/W | 0x0 | Sample 2 Digital Comparator Select<br>This field has the same encodings as S3DCSEL but is used during the third sample. |

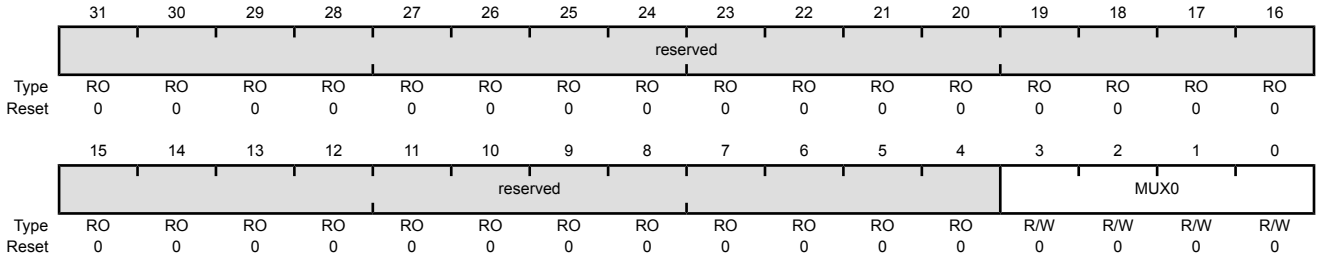
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7:4 | S1DCSEL | R/W | 0x0 | Sample 1 Digital Comparator Select<br>This field has the same encodings as S3DCSEL but is used during the second sample. |
| 3:0 | S0DCSEL | R/W | 0x0 | Sample 0 Digital Comparator Select<br>This field has the same encodings as S3DCSEL but is used during the first sample. |

## Register 34: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for the sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 567 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A0
Type R/W, reset 0x0000.0000



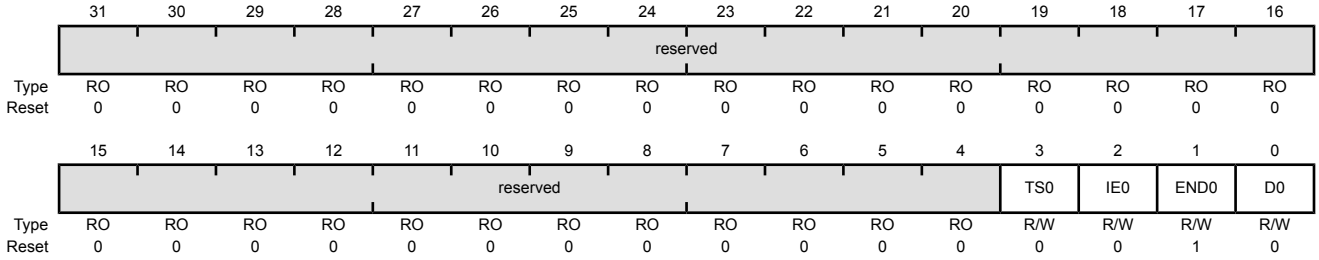

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3:0 | MUX0 | R/W | 0 | 1st Sample Input Select |

## Register 35: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. The END0 bit is always set as this sequencer can execute only one sample. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 569 for detailed bit descriptions.

ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A4
Type R/W, reset 0x0000.0002



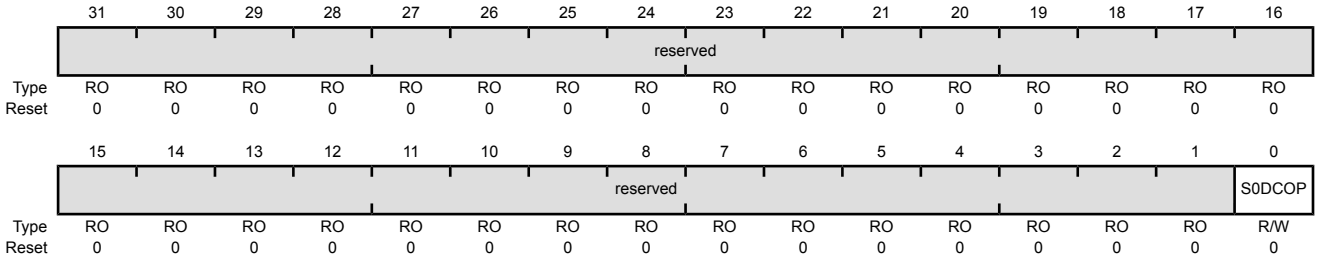

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | TS0 | R/W | 0 | 1st Sample Temp Sensor Select<br>Same definition as TS7 but used during the first sample. |
| 2 | IE0 | R/W | 0 | 1st Sample Interrupt Enable<br>Same definition as IE7 but used during the first sample. |
| 1 | END0 | R/W | 1 | 1st Sample is End of Sequence<br>Same definition as END7 but used during the first sample.<br>Because this sequencer has only one entry, this bit must be set. |
| 0 | D0 | R/W | 0 | 1st Sample Diff Input Select<br>Same definition as D7 but used during the first sample. |

## Register 36: 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 R/W, reset 0x0000.0000



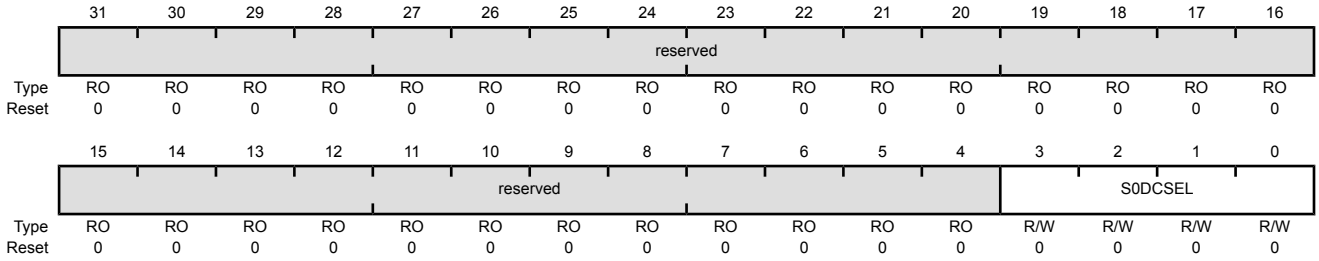

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | S0DCOP | R/W | 0 | Sample 0 Digital Comparator Operation<br>Value Description<br>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.<br>0 The sample is saved in Sample Sequence FIFO3. |

## Register 37: 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 R/W, 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 | R/W | 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 38: 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 R/W, 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 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | R/W | 0 | Digital Comparator Trigger 7<br><br>Value Description<br>1 Resets the Digital Comparator 7 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Trigger 6<br><br>Value Description<br>1 Resets the Digital Comparator 6 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Trigger 5<br><br>Value Description<br>1 Resets the Digital Comparator 5 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Trigger 4<br><br>Value Description<br>1 Resets the Digital Comparator 4 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Trigger 3<br><br>Value Description<br>1 Resets the Digital Comparator 3 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Trigger 2<br><br>Value Description<br>1 Resets the Digital Comparator 2 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Trigger 1<br><br>Value Description<br>1 Resets the Digital Comparator 1 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Trigger 0<br><br>Value Description<br>1 Resets the Digital Comparator 0 trigger unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Interrupt 7<br><br>Value Description<br>1 Resets the Digital Comparator 7 interrupt unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Interrupt 6<br><br>Value Description<br>1 Resets the Digital Comparator 6 interrupt unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Interrupt 5<br><br>Value Description<br>1 Resets the Digital Comparator 5 interrupt unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Interrupt 4<br><br>Value Description<br>1 Resets the Digital Comparator 4 interrupt unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Interrupt 3<br><br>Value Description<br>1 Resets the Digital Comparator 3 interrupt unit to its initial conditions.<br>0 No effect.<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 | R/W | 0 | Digital Comparator Interrupt 2<br><br>Value Description<br>1 Resets the Digital Comparator 2 interrupt unit to its initial conditions.<br>0 No effect.<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. |

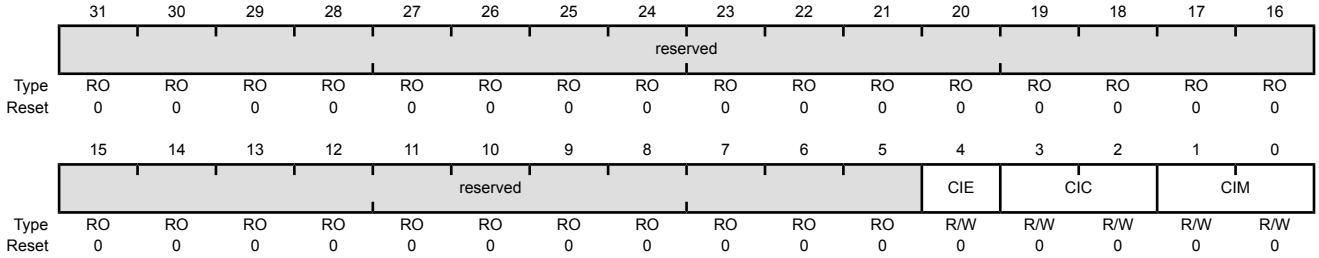
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | DCINT1 | R/W | 0 | Digital Comparator Interrupt 1<br><br>Value Description<br>1 Resets the Digital Comparator 1 interrupt unit to its initial conditions.<br>0 No effect.<br><br>When the interrupt has been cleared, this bit 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. |
| 0 | DCINT0 | R/W | 0 | Digital Comparator Interrupt 0<br><br>Value Description<br>1 Resets the Digital Comparator 0 interrupt unit to its initial conditions.<br>0 No effect.<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. |

### Register 39: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00
### Register 40: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04
### Register 41: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08
### Register 42: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C
### Register 43: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10
### Register 44: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14
### Register 45: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18
### Register 46: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C

This register provides the comparison encodings that generate an interrupt.

ADC Digital Comparator Control 0 (ADCDCCTL0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE00
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | 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. |
| 4 | CIE | R/W | 0 | Comparison Interrupt Enable |

| Value | Description |
|---|---|
| 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. |
| 0 | Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:2 | CIC | R/W | 0x0 | Comparison Interrupt Condition<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>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 | R/W | 0x0 | Comparison Interrupt Mode<br>This field specifies the mode by which the interrupt comparison is made.<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>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. |

### Register 47: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40
### Register 48: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44
### Register 49: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48
### Register 50: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C
### Register 51: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50
### Register 52: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54
### Register 53: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58
### Register 54: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C

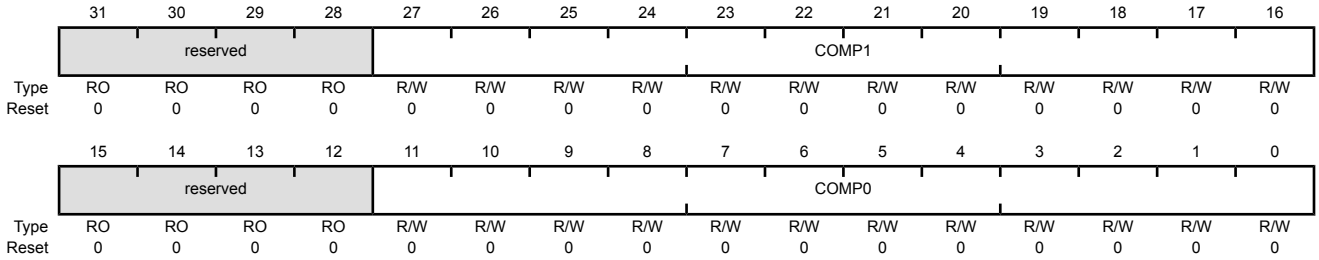
This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region.

**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.

If the RES bit in the **ADCCTL** register is clear, selecting 10-bit resolution, use only bits [25:16] in the COMP1 field and bits [9:0] in the COMP0 field; otherwise unexpected results can occur.

ADC Digital Comparator Range 0 (ADCDCCMP0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0xE40
Type R/W, 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 | R/W | 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 11:0 | COMP0 | R/W | 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. |

# 13 Universal Asynchronous Receivers/Transmitters (UARTs)

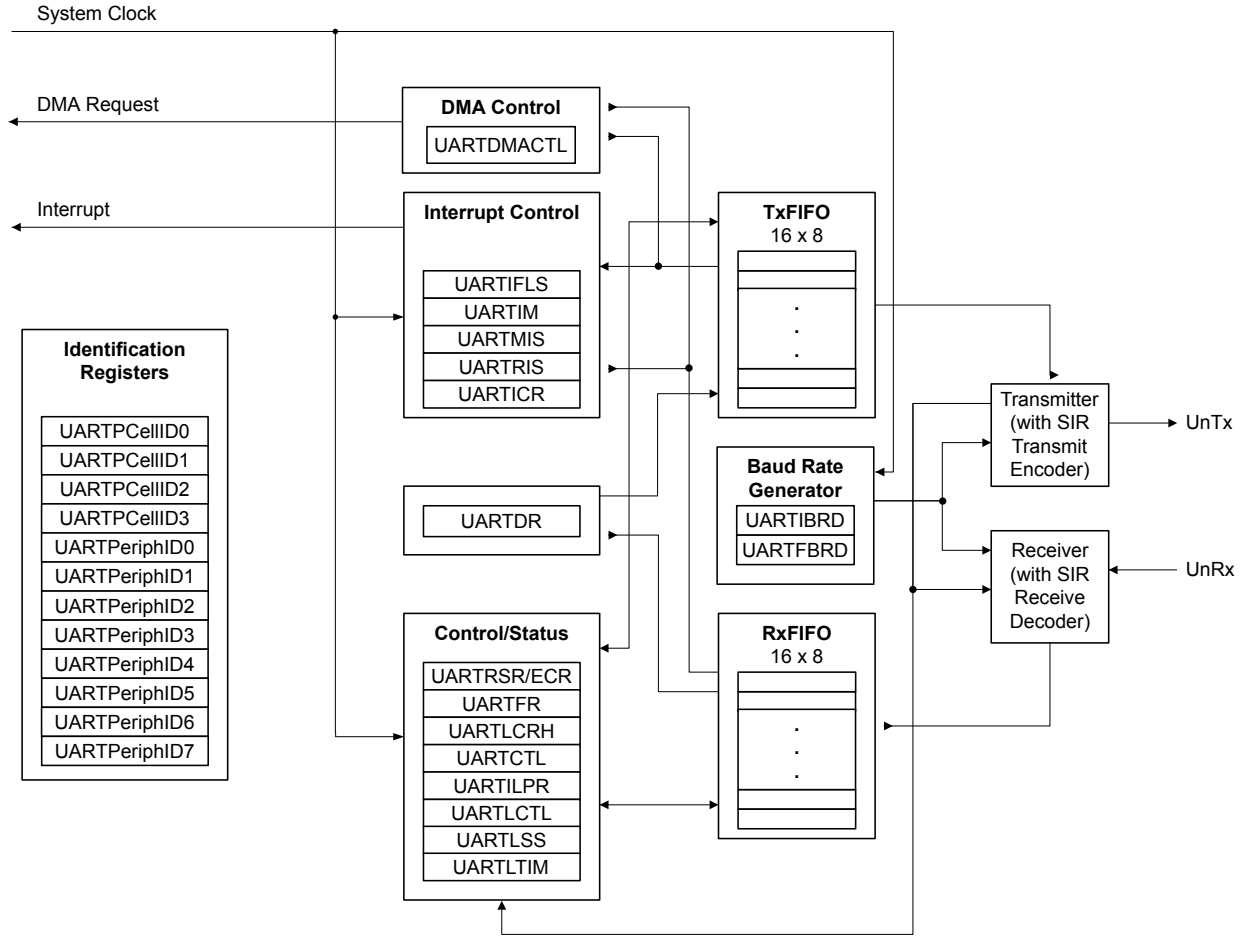
The Stellaris® LM3S1G58 controller includes three Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 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
- Full modem handshake support (on UART1)
- LIN protocol 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

## 13.1 Block Diagram

**Figure 13-1. UART Module Block Diagram**




## 13.2 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, with the exception of the U0Rx and U0Tx pins which default to the UART function. 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 418) 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 435) 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 395.

### Table 13-1. Signals for UART (100LQFP)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| U0Rx | 26 | PA0 (1) | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| U0Tx | 27 | PA1 (1) | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U1DSR | 47 | PF0 (9) | I | TTL | UART module 1 Data Set Ready modem output control line. |
| U1Rx | 10<br>12<br>23<br>26<br>66<br>92 | PD0 (5)<br>PD2 (1)<br>PC6 (5)<br>PA0 (9)<br>PB0 (5)<br>PB4 (7) | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| U1Tx | 11<br>13<br>22<br>27<br>67<br>91 | PD1 (5)<br>PD3 (1)<br>PC7 (5)<br>PA1 (9)<br>PB1 (5)<br>PB5 (7) | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U2Rx | 10<br>19<br>92<br>96 | PD0 (4)<br>PG0 (1)<br>PB4 (4)<br>PD5 (9) | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| U2Tx | 6<br>11<br>18<br>99 | PE4 (5)<br>PD1 (4)<br>PG1 (1)<br>PD6 (9) | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

### Table 13-2. Signals for UART (108BGA)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| U0Rx | L3 | PA0 (1) | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| U0Tx | M3 | PA1 (1) | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U1DSR | M9 | PF0 (9) | I | TTL | UART module 1 Data Set Ready modem output control line. |
| U1Rx | G1<br>H2<br>M2<br>L3<br>E12<br>A6 | PD0 (5)<br>PD2 (1)<br>PC6 (5)<br>PA0 (9)<br>PB0 (5)<br>PB4 (7) | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| U1Tx | G2<br>H1<br>L2<br>M3<br>D12<br>B7 | PD1 (5)<br>PD3 (1)<br>PC7 (5)<br>PA1 (9)<br>PB1 (5)<br>PB5 (7) | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |

**Table 13-2. Signals for UART (108BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| U2Rx | G1<br>K1<br>A6<br>B4 | PD0 (4)<br>PG0 (1)<br>PB4 (4)<br>PD5 (9) | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| U2Tx | B2<br>G2<br>K2<br>A3 | PE4 (5)<br>PD1 (4)<br>PG1 (1)<br>PD6 (9) | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 13.3 Functional Description

Each Stellaris 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 625). 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.

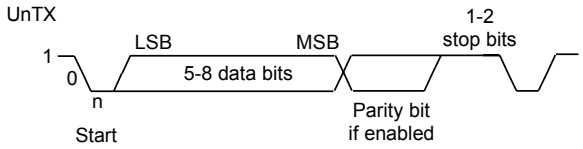
The UART module also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

### 13.3.1 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 13-2 on page 601 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 13-2. UART Character Frame**




### 13.3.2 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 divider 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 621) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor**

**(UARTFBRD)** register (see page 622). 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).

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 623), 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

### 13.3.3 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 617) 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 601).

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.

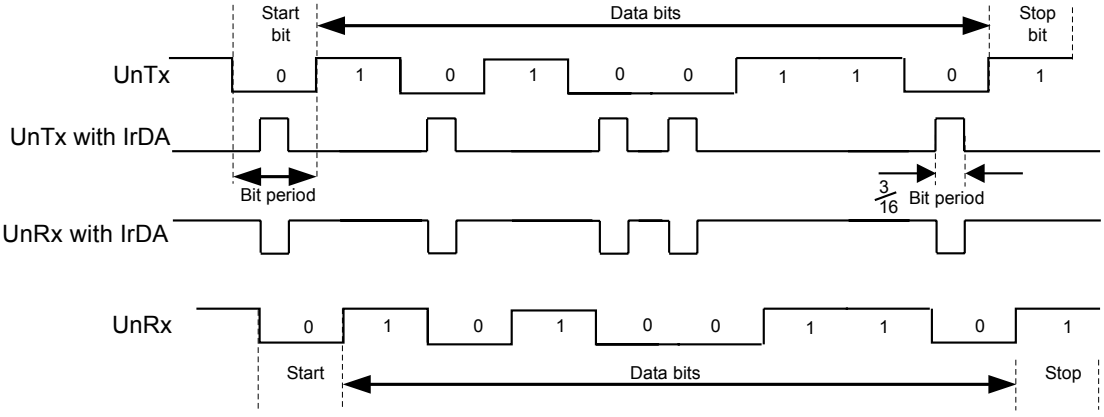
### 13.3.4 Serial IR (SIR)

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 **UARTCR** register. See page 620 for more information on IrDA low-power pulse-duration configuration.

Figure 13-3 on page 603 shows the UART transmit and receive signals, with and without IrDA modulation.

**Figure 13-3. 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.

### 13.3.5 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.

### 13.3.6 Modem Handshake Support

This section describes how to configure and use the modem flow control and status signals for UART1 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.

#### 13.3.6.1 Signaling

The status signals provided by UART1 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:

- U1CTS is Clear To Send
- U1DSR is Data Set Ready
- U1DCD is Data Carrier Detect
- U1RI is Ring Indicator
- U1RTS is Request To Send
- U1DTR is Data Terminal Ready

When used as a DCE, the the modem flow control and status signals are defined as:

- U1CTS is Request To Send
- U1DSR is Data Terminal Ready
- U1RTS is Clear To Send

- U1DTR 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.

#### 13.3.6.2 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 U1RTS output to the Clear-To-Send input on the receiving device, and connecting the Request-To-Send output on the receiving device to the U1CTS input.

The U1CTS input controls the transmitter. The transmitter may only transmit data when the U1CTS input is asserted. The U1RTS output signal indicates the state of the receive FIFO. U1CTS 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 13-3 on page 605.

**Table 13-3. 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 U1RTS output value through the **UARTCTL** register Request to Send (RTS) bit, and the status of the RTS bit should be ignored.

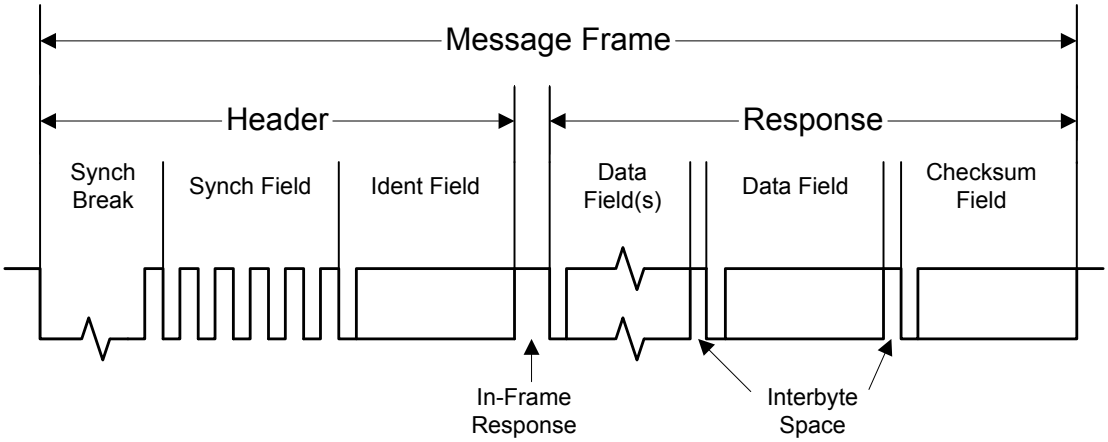
***Software Flow Control (Modem Status Interrupts)***

Software flow control between two devices is accomplished by using interrupts to indicate the status of the UART. Interrupts may be generated for the U1DSR, U1DCD, U1CTS, and U1RI 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.

### 13.3.7 LIN Support

The UART module offers hardware support for the LIN protocol as either a master or a slave. The LIN mode is enabled by setting the LIN bit in the **UARTCTL** register. A LIN message is identified by the use of a Sync Break at the beginning of the message. The Sync Break is a transmission of a series of 0s. The Sync Break is followed by the Sync data field (0x55). Figure 13-4 on page 606 illustrates the structure of a LIN message.

**Figure 13-4. LIN Message**




The UART should be configured as followed to operate in LIN mode:

1. Configure the UART for 1 start bit, 8 data bits, no parity, and 1 stop bit. Enable the Transmit FIFO.
2. Set the LIN bit in the **UARTCTL** register.

When preparing to send a LIN message, the TXFIFO should contain the Sync data (0x55) at FIFO location 0 and the Identifier data at location 1, followed by the data to be transmitted, and with the checksum in the final FIFO entry.

#### 13.3.7.1 LIN Master

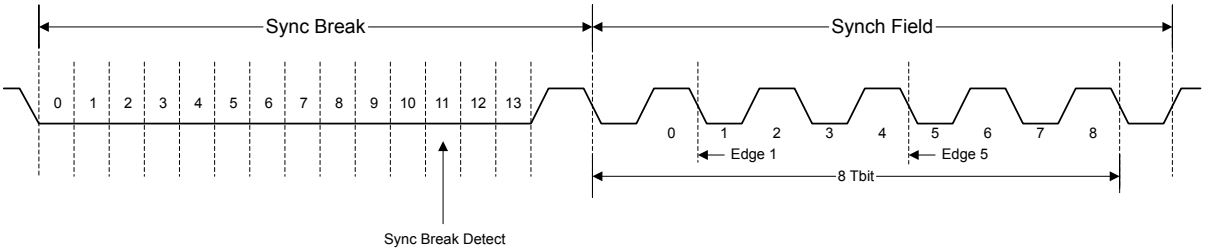
The UART is enabled to be the LIN master by setting the MASTER bit in the **UARTLCTL** register. The length of the Sync Break is programmable using the BLEN field in the **UARTLCTL** register and can be 13-16 bits (baud clock cycles).

#### 13.3.7.2 LIN Slave

The LIN UART slave is required to adjust its baud rate to that of the LIN master. In slave mode, the LIN UART recognizes the Sync Break, which must be at least 13 bits in duration. A timer is provided to capture timing data on the 1st and 5th falling edges of the Sync field so that the baud rate can be adjusted to match the master.

After detecting a Sync Break, the UART waits for the synchronization field. The first falling edge generates an interrupt using the LME1RIS bit in the **UARTRIS** register, and the timer value is captured and stored in the **UARTLSS** register (T1). On the fifth falling edge, a second interrupt is generated using the LME5RIS bit in the **UARTRIS** register, and the timer value is captured again (T2). The actual baud rate can be calculated using (T2-T1)/8, and the local baud rate should be adjusted as needed. Figure 13-5 on page 607 illustrates the synchronization field.

**Figure 13-5. LIN Synchronization Field**




### 13.3.8 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 612). 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 623).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 617) 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.

The trigger points at which the FIFOs generate interrupts is controlled via the **UART Interrupt FIFO Level Select (UARTIFLS)** register (see page 629). 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.

### 13.3.9 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 638).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 631) by setting the corresponding IM bits. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 635).

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 641).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. 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.

### 13.3.10 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 625). 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.

### 13.3.11 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.

If DMA is enabled, then the µDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the UART interrupt vector. Therefore, if interrupts are used for UART operation and DMA is enabled, the UART interrupt handler must be designed to handle the µDMA completion interrupt.

See “Micro Direct Memory Access (µDMA)” on page 335 for more details about programming the µDMA controller.

## 13.4 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

1. The peripheral clock must be enabled by setting the UART0, UART1, or UART2 bits in the **RCGC1** register (see page 243).

2. The clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 251).

3. Set the GPIO AFSEL bits for the appropriate pins (see page 418). To determine which GPIOs to configure, see Table 17-4 on page 753.

4. Configure the GPIO current level and/or slew rate as specified for the mode selected (see page 420 and page 428).

5. Configure the PMCn fields in the **GPIOPCTL** register to assign the UART signals to the appropriate pins (see page 435 and Table 17-5 on page 757).

To use the UART, the peripheral clock must be enabled by setting the appropriate bit in the **RCGC1** register (page 243). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register (page 251) in the System Control module. To find out which GPIO port to enable, refer to Table 17-5 on page 757.

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 601, 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 621) should be set to 10 decimal or 0xA. The value to be loaded into the **UARTFBRD** register (see page 622) 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. Optionally, configure the µDMA channel (see “Micro Direct Memory Access (µDMA)” on page 335) and enable the DMA option(s) in the **UARTDMACTL** register.

6. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

## 13.5 Register Map

Table 13-4 on page 610 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

Note that the UART module clock must be enabled before the registers can be programmed (see page 243). There must be a delay of 3 system clocks after the UART module clock is enabled before any UART module registers are accessed.

**Note:** The UART must be disabled (see the UARTEN bit in the **UARTCTL** register on page 625) 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.

**Table 13-4. UART Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | UARTDR | R/W | 0x0000.0000 | UART Data | 612 |
| 0x004 | UARTRSR/UARTECR | R/W | 0x0000.0000 | UART Receive Status/Error Clear | 614 |
| 0x018 | UARTFR | RO | 0x0000.0090 | UART Flag | 617 |
| 0x020 | UARTILPR | R/W | 0x0000.0000 | UART IrDA Low-Power Register | 620 |
| 0x024 | UARTIBRD | R/W | 0x0000.0000 | UART Integer Baud-Rate Divisor | 621 |
| 0x028 | UARTFBRD | R/W | 0x0000.0000 | UART Fractional Baud-Rate Divisor | 622 |
| 0x02C | UARTLCRH | R/W | 0x0000.0000 | UART Line Control | 623 |
| 0x030 | UARTCTL | R/W | 0x0000.0300 | UART Control | 625 |
| 0x034 | UARTIFLS | R/W | 0x0000.0012 | UART Interrupt FIFO Level Select | 629 |
| 0x038 | UARTIM | R/W | 0x0000.0000 | UART Interrupt Mask | 631 |
| 0x03C | UARTRIS | RO | 0x0000.000F | UART Raw Interrupt Status | 635 |
| 0x040 | UARTMIS | RO | 0x0000.0000 | UART Masked Interrupt Status | 638 |
| 0x044 | UARTICR | W1C | 0x0000.0000 | UART Interrupt Clear | 641 |
| 0x048 | UARTDMACTL | R/W | 0x0000.0000 | UART DMA Control | 643 |
| 0x090 | UARTLCTL | R/W | 0x0000.0000 | UART LIN Control | 644 |
| 0x094 | UARTLSS | RO | 0x0000.0000 | UART LIN Snap Shot | 645 |
| 0x098 | UARTLTIM | RO | 0x0000.0000 | UART LIN Timer | 646 |
| 0xFD0 | UARTPeriphID4 | RO | 0x0000.0000 | UART Peripheral Identification 4 | 647 |
| 0xFD4 | UARTPeriphID5 | RO | 0x0000.0000 | UART Peripheral Identification 5 | 648 |
| 0xFD8 | UARTPeriphID6 | RO | 0x0000.0000 | UART Peripheral Identification 6 | 649 |

**Table 13-4. UART Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0xFDC | UARTPeriphID7 | RO | 0x0000.0000 | UART Peripheral Identification 7 | 650 |
| 0xFE0 | UARTPeriphID0 | RO | 0x0000.0060 | UART Peripheral Identification 0 | 651 |
| 0xFE4 | UARTPeriphID1 | RO | 0x0000.0000 | UART Peripheral Identification 1 | 652 |
| 0xFE8 | UARTPeriphID2 | RO | 0x0000.0018 | UART Peripheral Identification 2 | 653 |
| 0xFEC | UARTPeriphID3 | RO | 0x0000.0001 | UART Peripheral Identification 3 | 654 |
| 0xFF0 | UARTPCellID0 | RO | 0x0000.000D | UART PrimeCell Identification 0 | 655 |
| 0xFF4 | UARTPCellID1 | RO | 0x0000.00F0 | UART PrimeCell Identification 1 | 656 |
| 0xFF8 | UARTPCellID2 | RO | 0x0000.0005 | UART PrimeCell Identification 2 | 657 |
| 0xFFC | UARTPCellID3 | RO | 0x0000.00B1 | UART PrimeCell Identification 3 | 658 |

## 13.6 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
Offset 0x000
Type R/W, 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 | | | | OE | BE | PE | FE | DATA | | | | | | | |
| RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 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 | OE | RO | 0 | UART Overrun Error<br>Value Description<br>1 New data was received when the FIFO was full, resulting in data loss.<br>0 No data has been lost due to a FIFO overrun. |
| 10 | BE | RO | 0 | UART Break Error<br>Value Description<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>0 No break condition has occurred<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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 9 | PE | RO | 0 | UART Parity Error<br><br>Value Description<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>0 No parity error has occurred<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>1 The received character does not have a valid stop bit (a valid stop bit is 1).<br>0 No framing error has occurred |
| 7:0 | DATA | R/W | 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.

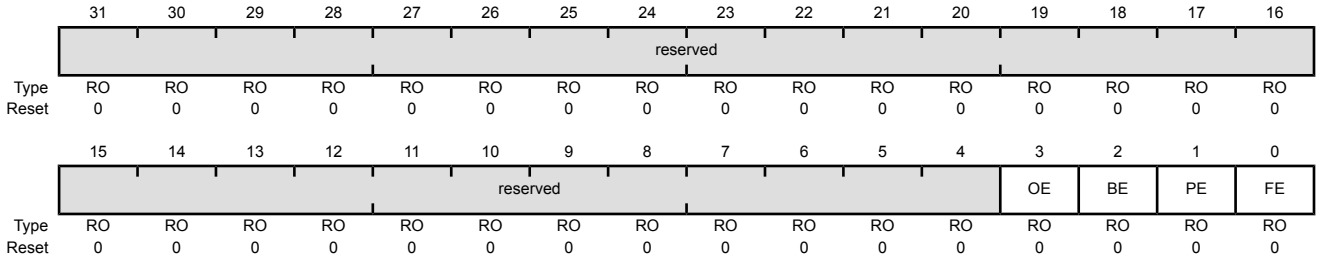
The **UARTRSR** register cannot be written.

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
Offset 0x004
Type RO, reset 0x0000.0000



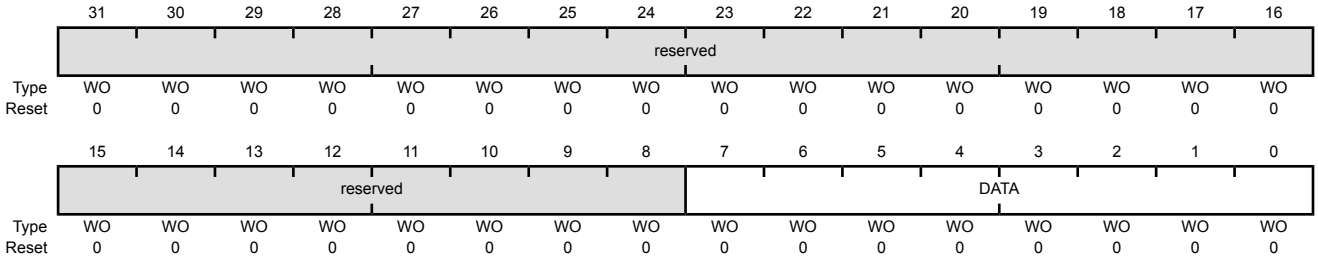

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | OE | RO | 0 | UART Overrun Error<br><br>Value Description<br>1 New data was received when the FIFO was full, resulting in data loss.<br>0 No data has been lost due to a FIFO overrun.<br><br>This bit is cleared by a write to **UARTECR**.<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>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>0 No break condition has occurred<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>1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register.<br>0 No parity error has occurred<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>1 The received character does not have a valid stop bit (a valid stop bit is 1).<br>0 No framing error has occurred<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
Offset 0x004
Type WO, reset 0x0000.0000



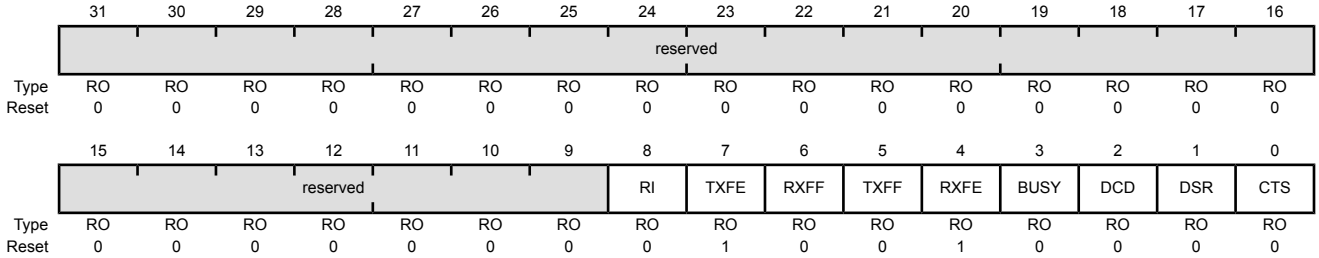
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | WO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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. The RI, DCD, DSR and CTS bits indicate the modem flow control and status. Note that the modem bits are only implemented on UART1 and are reserved on UART0 and UART2.

UART Flag (UARTFR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x018
Type RO, reset 0x0000.0090




| 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><br>Value Description<br>1 The U1RI signal is asserted.<br>0 The U1RI signal is not asserted.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 7 | TXFE | RO | 1 | UART Transmit FIFO Empty<br><br>The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register.<br><br>Value Description<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.<br>0 The transmitter has data to transmit. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 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>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.<br>0 The receiver can receive data. |
| 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>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.<br>0 The transmitter is not 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>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.<br>0 The receiver is not empty. |
| 3 | BUSY | RO | 0 | UART Busy<br><br>Value Description<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>0 The UART is not busy.<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>1 The U1DCD signal is asserted.<br>0 The U1DCD signal is not asserted.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | DSR | RO | 0 | Data Set Ready<br><br>Value Description<br>1 The U1DSR signal is asserted.<br>0 The U1DSR signal is not asserted.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 0 | CTS | RO | 0 | Clear To Send<br><br>Value Description<br>1 The U1CTS signal is asserted.<br>0 The U1CTS signal is not asserted.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

## 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 / $F_{IrLPBaud16}$

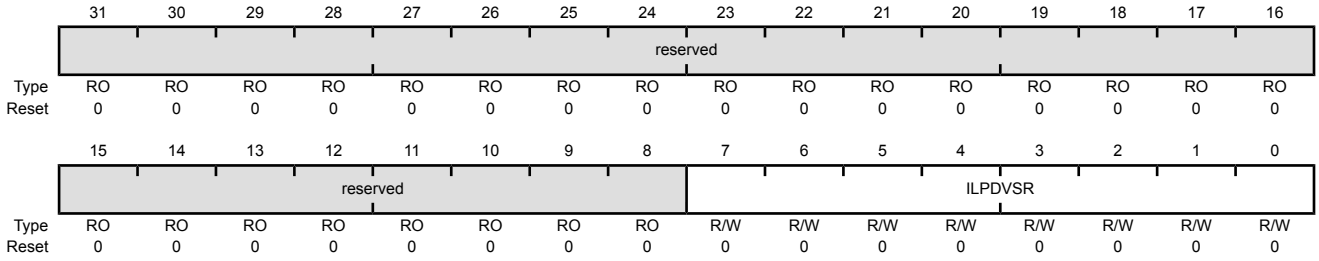
where $F_{IrLPBaud16}$ is nominally 1.8432 MHz.

The divisor 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.

**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
Offset 0x020
Type R/W, reset 0x0000.0000



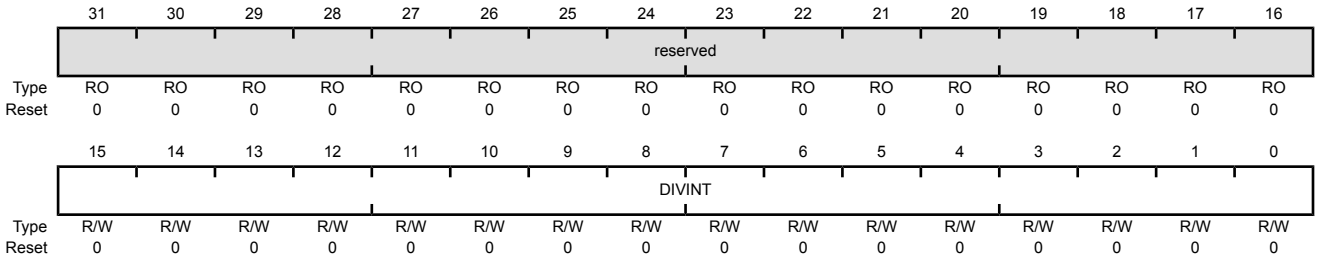

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | ILPDVSR | R/W | 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 601 for configuration details.

UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x024
Type R/W, reset 0x0000.0000



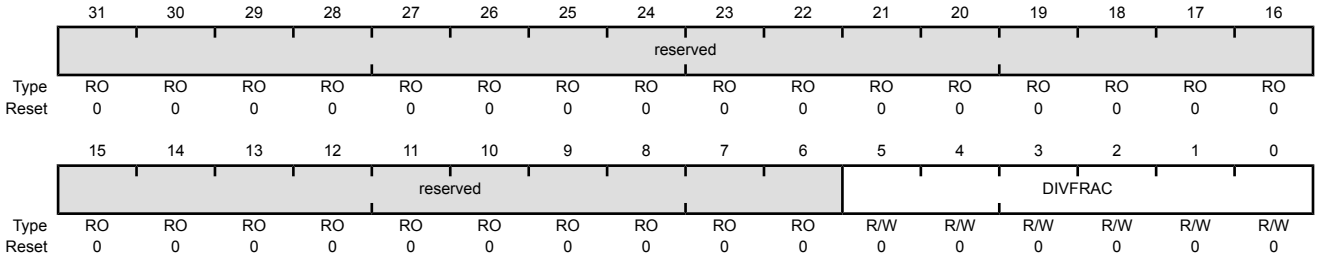

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | DIVINT | R/W | 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 601 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x028
Type R/W, 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 | R/W | 0x0 | Fractional Baud-Rate Divisor |

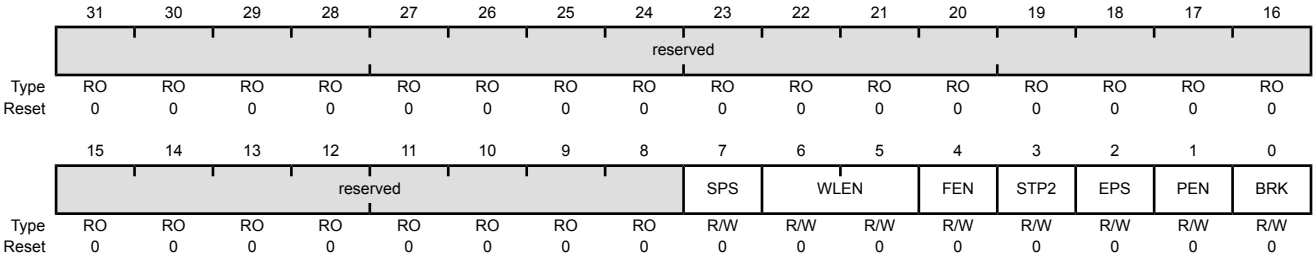
## Register 7: UART Line Control (UARTLCRH), offset 0x02C

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
Offset 0x02C
Type R/W, 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 | R/W | 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 | R/W | 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 |
| 4 | FEN | R/W | 0 | UART Enable FIFOs<br>Value Description<br>1 The transmit and receive FIFO buffers are enabled (FIFO mode).<br>0 The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | STP2 | R/W | 0 | UART Two Stop Bits Select<br><br>Value Description<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.<br>0 One stop bit is transmitted at the end of a frame. |
| 2 | EPS | R/W | 0 | UART Even Parity Select<br><br>Value Description<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>0 Odd parity is performed, which checks for an odd number of 1s.<br><br>This bit has no effect when parity is disabled by the PEN bit. |
| 1 | PEN | R/W | 0 | UART Parity Enable<br><br>Value Description<br>1 Parity checking and generation is enabled.<br>0 Parity is disabled and no parity bit is added to the data frame. |
| 0 | BRK | R/W | 0 | UART Send Break<br><br>Value Description<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).<br>0 Normal use. |

## 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.

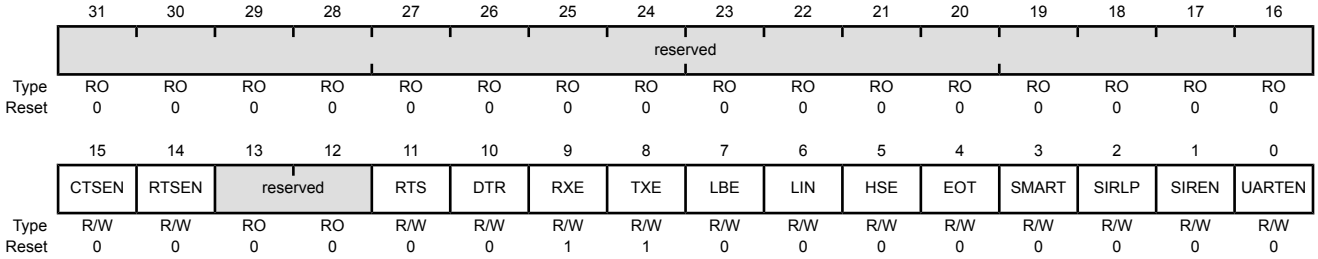
Note that bits [15:14,11:10] are only implemented on UART1. These bits are reserved on UART0 and UART2.

**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
Offset 0x030
Type R/W, reset 0x0000.0300




| 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 | CTSEN | R/W | 0 | Enable Clear To Send<br>Value Description<br>1 CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.<br>0 CTS hardware flow control is disabled.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 14 | RTSEN | R/W | 0 | Enable Request to Send<br>Value Description<br>1 RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.<br>0 RTS hardware flow control is disabled.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 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 | R/W | 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.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 10 | DTR | R/W | 0 | Data Terminal Ready<br>This bit sets the state of the U1DTR output.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 9 | RXE | R/W | 1 | UART Receive Enable<br>Value Description<br>1 The receive section of the UART is enabled.<br>0 The receive section of the UART is disabled.<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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TXE | R/W | 1 | UART Transmit Enable<br><br>Value Description<br>1 The transmit section of the UART is enabled.<br>0 The transmit section of the UART is disabled.<br><br>If the UART is disabled in the middle of a transmission, it completes the current character before stopping.<br><br>**Note:** To enable transmission, the UARTEN bit must also be set. |
| 7 | LBE | R/W | 0 | UART Loop Back Enable<br><br>Value Description<br>1 The UnTx path is fed through the UnRx path.<br>0 Normal operation. |
| 6 | LIN | R/W | 0 | LIN Mode Enable<br><br>Value Description<br>1 The UART operates in LIN mode.<br>0 Normal operation. |
| 5 | HSE | R/W | 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 621) and page 622).<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 | R/W | 0 | End of Transmission<br>This bit determines the behavior of the TXRIS bit in the **UARTRIS** register.<br><br>Value Description<br>1 The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer.<br>0 The TXRIS bit is set when the transmit FIFO condition specified in **UARTIFLS** is met. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3 | SMART | R/W | 0 | ISO 7816 Smart Card Support<br><br>Value Description<br>1 The UART operates in Smart Card mode.<br>0 Normal operation.<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><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. |
| 2 | SIRLP | R/W | 0 | UART SIR Low-Power Mode<br><br>This bit selects the IrDA encoding mode.<br><br>Value Description<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>0 Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.<br><br>Setting this bit uses less power, but might reduce transmission distances. See page 620 for more information. |
| 1 | SIREN | R/W | 0 | UART SIR Enable<br><br>Value Description<br>1 The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.<br>0 Normal operation. |
| 0 | UARTEN | R/W | 0 | UART Enable<br><br>Value Description<br>1 The UART is enabled.<br>0 The UART is disabled.<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

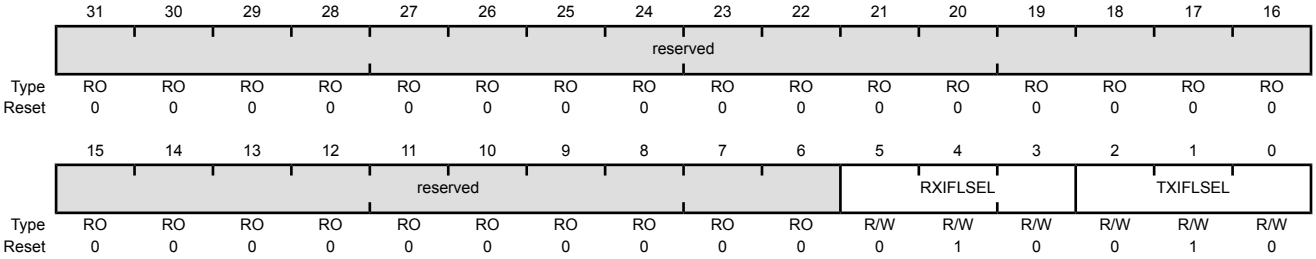
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the **UARTRIS** register are triggered.

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
Offset 0x034
Type R/W, 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 | R/W | 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 | R/W | 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 625), 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

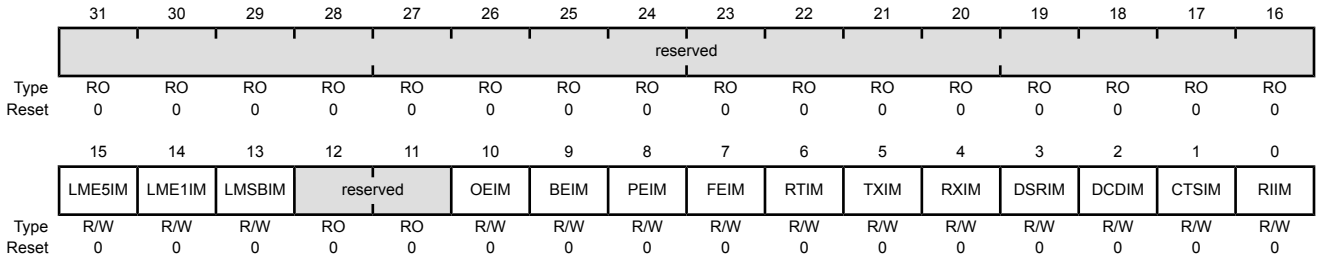
The **UARTIM** register is the interrupt mask set/clear register.

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 that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x038
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | 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 | LME5IM | R/W | 0 | LIN Mode Edge 5 Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the LME5RIS bit in the **UARTRIS** register is set.<br>0 The LME5RIS interrupt is suppressed and not sent to the interrupt controller. |
| 14 | LME1IM | R/W | 0 | LIN Mode Edge 1 Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the LME1RIS bit in the **UARTRIS** register is set.<br>0 The LME1RIS interrupt is suppressed and not sent to the interrupt controller. |
| 13 | LMSBIM | R/W | 0 | LIN Mode Sync Break Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the LMSBRIS bit in the **UARTRIS** register is set.<br>0 The LMSBRIS interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12:11 | 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 | OEIM | R/W | 0 | UART Overrun Error Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the OERIS bit in the **UARTRIS** register is set.<br>0 The OERIS interrupt is suppressed and not sent to the interrupt controller. |
| 9 | BEIM | R/W | 0 | UART Break Error Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the BERIS bit in the **UARTRIS** register is set.<br>0 The BERIS interrupt is suppressed and not sent to the interrupt controller. |
| 8 | PEIM | R/W | 0 | UART Parity Error Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the PERIS bit in the **UARTRIS** register is set.<br>0 The PERIS interrupt is suppressed and not sent to the interrupt controller. |
| 7 | FEIM | R/W | 0 | UART Framing Error Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the FERIS bit in the **UARTRIS** register is set.<br>0 The FERIS interrupt is suppressed and not sent to the interrupt controller. |
| 6 | RTIM | R/W | 0 | UART Receive Time-Out Interrupt Mask<br>Value Description<br>1 An interrupt is sent to the interrupt controller when the RTRIS bit in the **UARTRIS** register is set.<br>0 The RTRIS interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TXIM | R/W | 0 | UART Transmit Interrupt Mask<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the TXRIS bit in the **UARTRIS** register is set.<br>0 The TXRIS interrupt is suppressed and not sent to the interrupt controller. |
| 4 | RXIM | R/W | 0 | UART Receive Interrupt Mask<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the RXRIS bit in the **UARTRIS** register is set.<br>0 The RXRIS interrupt is suppressed and not sent to the interrupt controller. |
| 3 | DSRIM | R/W | 0 | UART Data Set Ready Modem Interrupt Mask<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the DSRRIS bit in the **UARTRIS** register is set.<br>0 The DSRRIS interrupt is suppressed and not sent to the interrupt controller.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 2 | DCDIM | R/W | 0 | UART Data Carrier Detect Modem Interrupt Mask<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the DCDRIS bit in the **UARTRIS** register is set.<br>0 The DCDRIS interrupt is suppressed and not sent to the interrupt controller.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 1 | CTSIM | R/W | 0 | UART Clear to Send Modem Interrupt Mask<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the CTSRIS bit in the **UARTRIS** register is set.<br>0 The CTSRIS interrupt is suppressed and not sent to the interrupt controller.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RIIM | R/W | 0 | UART Ring Indicator Modem Interrupt Mask<br><br>Value Description<br>1 An interrupt is sent to the interrupt controller when the RIRIS bit in the **UARTRIS** register is set.<br>0 The RIRIS interrupt is suppressed and not sent to the interrupt controller.<br><br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

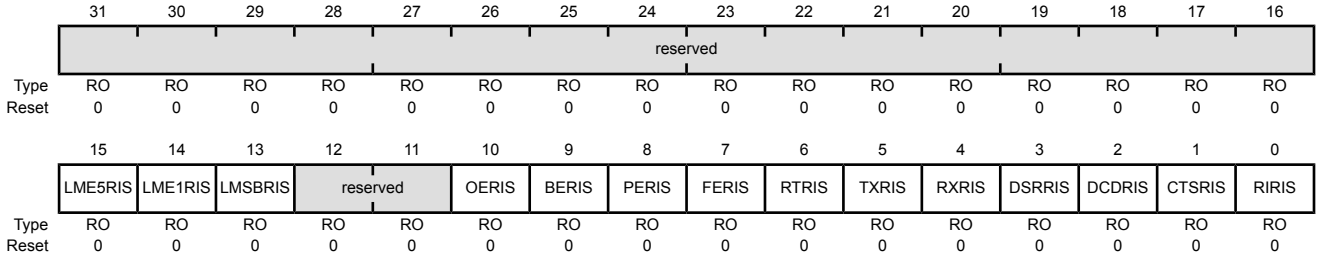
## Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

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.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x03C
Type RO, reset 0x0000.000F




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | 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 | LME5RIS | RO | 0 | LIN Mode Edge 5 Raw Interrupt Status<br><br>Value Description<br>1 The timer value at the 5th falling edge of the LIN Sync Field has been captured.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the `LME5IC` bit in the **UARTICR** register. |
| 14 | LME1RIS | RO | 0 | LIN Mode Edge 1 Raw Interrupt Status<br><br>Value Description<br>1 The timer value at the 1st falling edge of the LIN Sync Field has been captured.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the `LME1IC` bit in the **UARTICR** register. |
| 13 | LMSBRIS | RO | 0 | LIN Mode Sync Break Raw Interrupt Status<br><br>Value Description<br>1 A LIN Sync Break has been detected.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the `LMSBIC` bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12:11 | 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 | OERIS | RO | 0 | UART Overrun Error Raw Interrupt Status<br>Value Description<br>1 An overrun error has occurred.<br>0 No interrupt<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>Value Description<br>1 A break error has occurred.<br>0 No interrupt<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>Value Description<br>1 A parity error has occurred.<br>0 No interrupt<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>Value Description<br>1 A framing error has occurred.<br>0 No interrupt<br>This bit is cleared by writing a 1 to the FEIC bit in the **UARTICR** register. |
| 6 | RTRIS | RO | 0 | UART Receive Time-Out Raw Interrupt Status<br>Value Description<br>1 A receive time out has occurred.<br>0 No interrupt<br>This bit is cleared by writing a 1 to the RTIC bit in the **UARTICR** register. |
| 5 | TXRIS | RO | 0 | UART Transmit Raw Interrupt Status<br>Value Description<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>0 No interrupt<br>This bit is cleared by writing a 1 to the TXIC bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | RXRIS | RO | 0 | UART Receive Raw Interrupt Status<br><br>Value Description<br>1 The receive FIFO level has passed through the condition defined in the **UARTIFLS** register.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register. |
| 3 | DSRRIS | RO | 0 | UART Data Set Ready Modem Raw Interrupt Status<br><br>Value Description<br>1 Data Set Ready used for software flow control.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the DSRIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 2 | DCDRIS | RO | 0 | UART Data Carrier Detect Modem Raw Interrupt Status<br><br>Value Description<br>1 Data Carrier Detect used for software flow control.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the DCDIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 1 | CTSRIS | RO | 0 | UART Clear to Send Modem Raw Interrupt Status<br><br>Value Description<br>1 Clear to Send used for software flow control.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the CTSIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 0 | RIRIS | RO | 0 | UART Ring Indicator Modem Raw Interrupt Status<br><br>Value Description<br>1 Ring Indicator used for software flow control.<br>0 No interrupt<br><br>This bit is cleared by writing a 1 to the RIIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

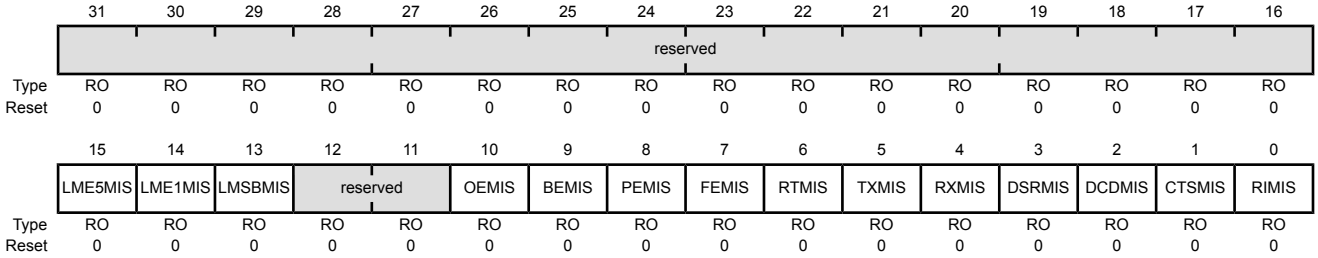
## Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

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.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x040
Type RO, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | 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 | LME5MIS | RO | 0 | LIN Mode Edge 5 Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to the 5th falling edge of the LIN Sync Field.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the `LME5IC` bit in the **UARTICR** register. |
| 14 | LME1MIS | RO | 0 | LIN Mode Edge 1 Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to the 1st falling edge of the LIN Sync Field.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the `LME1IC` bit in the **UARTICR** register. |
| 13 | LMSBMIS | RO | 0 | LIN Mode Sync Break Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to the receipt of a LIN Sync Break.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the `LMSBIC` bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 12:11 | 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 | OEMIS | RO | 0 | UART Overrun Error Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to an overrun error.<br>0 An interrupt has not occurred or is masked.<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>1 An unmasked interrupt was signaled due to a break error.<br>0 An interrupt has not occurred or is masked.<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>1 An unmasked interrupt was signaled due to a parity error.<br>0 An interrupt has not occurred or is masked.<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>1 An unmasked interrupt was signaled due to a framing error.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the FEIC bit in the **UARTICR** register. |
| 6 | RTMIS | RO | 0 | UART Receive Time-Out Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to a receive time out.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the RTIC bit in the **UARTICR** register. |
| 5 | TXMIS | RO | 0 | UART Transmit Masked Interrupt Status<br><br>Value Description<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>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the TXIC bit in the **UARTICR** register. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | RXMIS | RO | 0 | UART Receive Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to passing through the specified receive FIFO level.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the RXIC bit in the **UARTICR** register. |
| 3 | DSRMIS | RO | 0 | UART Data Set Ready Modem Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to Data Set Ready.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the DSRIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 2 | DCDMIS | RO | 0 | UART Data Carrier Detect Modem Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to Data Carrier Detect.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the DCDIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 1 | CTSMIS | RO | 0 | UART Clear to Send Modem Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to Clear to Send.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the CTSIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 0 | RIMIS | RO | 0 | UART Ring Indicator Modem Masked Interrupt Status<br><br>Value Description<br>1 An unmasked interrupt was signaled due to Ring Indicator.<br>0 An interrupt has not occurred or is masked.<br><br>This bit is cleared by writing a 1 to the RIIC bit in the **UARTICR** register.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

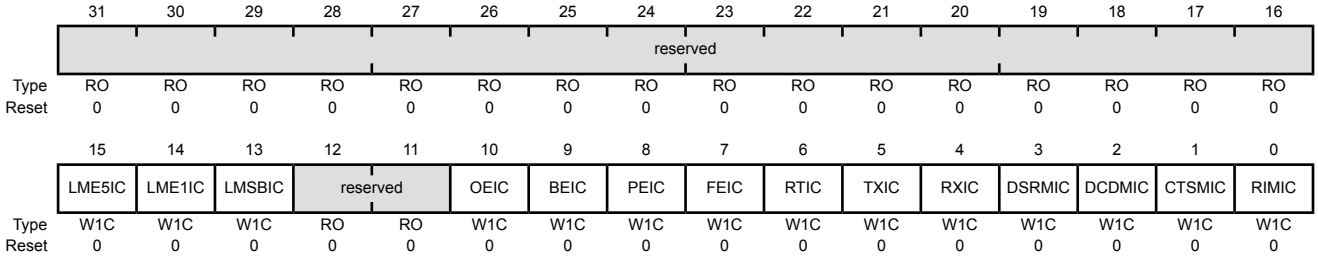
## Register 13: UART Interrupt Clear (UARTICR), offset 0x044

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
Offset 0x044
Type W1C, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | 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 | LME5IC | W1C | 0 | LIN Mode Edge 5 Interrupt Clear<br>Writing a 1 to this bit clears the LME5RIS bit in the **UARTRIS** register and the LME5MIS bit in the **UARTMIS** register. |
| 14 | LME1IC | W1C | 0 | LIN Mode Edge 1 Interrupt Clear<br>Writing a 1 to this bit clears the LME1RIS bit in the **UARTRIS** register and the LME1MIS bit in the **UARTMIS** register. |
| 13 | LMSBIC | W1C | 0 | LIN Mode Sync Break Interrupt Clear<br>Writing a 1 to this bit clears the LMSBRIS bit in the **UARTRIS** register and the LMSBMIS bit in the **UARTMIS** register. |
| 12:11 | 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 | 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. |
| 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. |

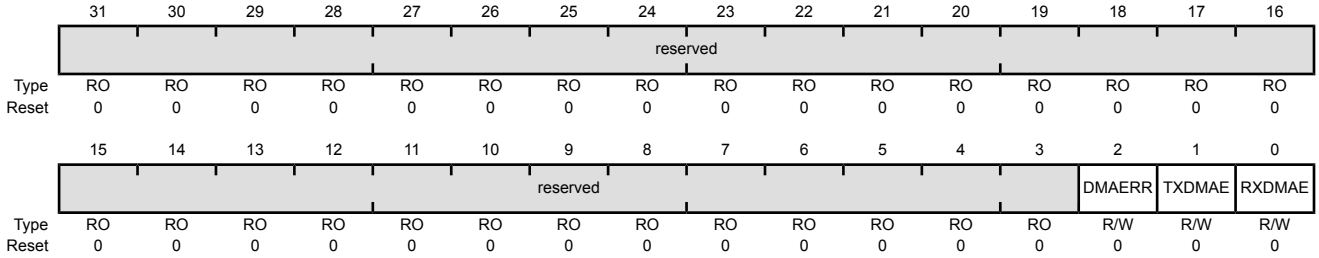
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 7 | FEIC | W1C | 0 | Framing Error Interrupt Clear<br>Writing a 1 to this bit clears the FERIS bit in the **UARTRIS** register and the FEMIS bit in the **UARTMIS** register. |
| 6 | RTIC | W1C | 0 | Receive Time-Out Interrupt Clear<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.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 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.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 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.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 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.<br>This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

## 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
Offset 0x048
Type R/W, reset 0x0000.0000



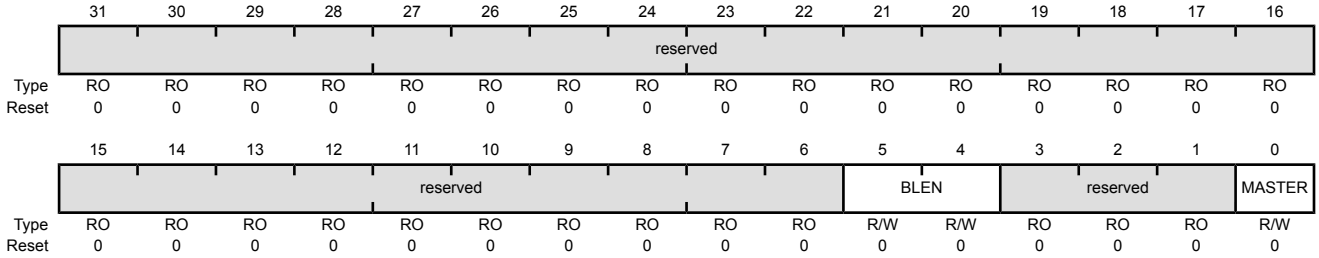

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x00000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DMAERR | R/W | 0 | DMA on Error<br>Value Description<br>1 µDMA receive requests are automatically disabled when a receive error occurs.<br>0 µDMA receive requests are unaffected when a receive error occurs. |
| 1 | TXDMAE | R/W | 0 | Transmit DMA Enable<br>Value Description<br>1 µDMA for the transmit FIFO is enabled.<br>0 µDMA for the transmit FIFO is disabled. |
| 0 | RXDMAE | R/W | 0 | Receive DMA Enable<br>Value Description<br>1 µDMA for the receive FIFO is enabled.<br>0 µDMA for the receive FIFO is disabled. |

## Register 15: UART LIN Control (UARTLCTL), offset 0x090

The **UARTLCTL** register is the configures the operation of the UART when in LIN mode.

UART LIN Control (UARTLCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x090
Type R/W, reset 0x0000.0000



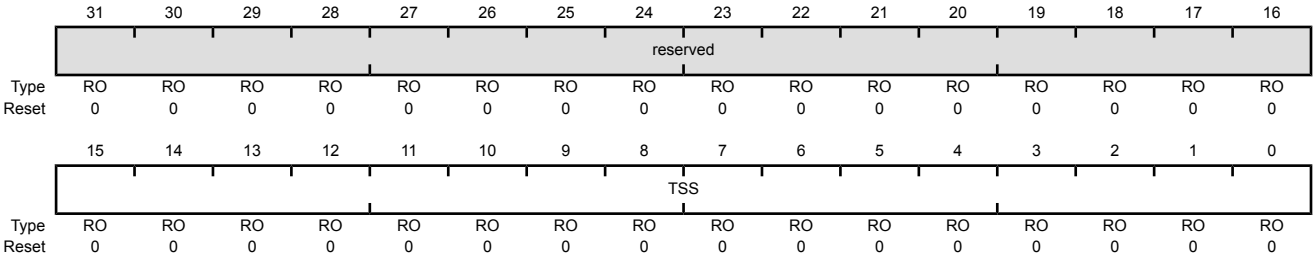

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:6 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5:4 | BLEN | R/W | 0x0 | Sync Break Length<br><br>Value Description<br>0x3 Sync break length is 16T bits<br>0x2 Sync break length is 15T bits<br>0x1 Sync break length is 14T bits<br>0x0 Sync break length is 13T bits (default) |
| 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 | MASTER | R/W | 0 | LIN Master Enable<br><br>Value Description<br>1 The UART operates as a LIN master.<br>0 The UART operates as a LIN slave. |

## Register 16: UART LIN Snap Shot (UARTLSS), offset 0x094

The **UARTLSS** register captures the free-running timer value when either the Sync Edge 1 or the Sync Edge 5 is detected in LIN mode.

UART LIN Snap Shot (UARTLSS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x094
Type RO, reset 0x0000.0000



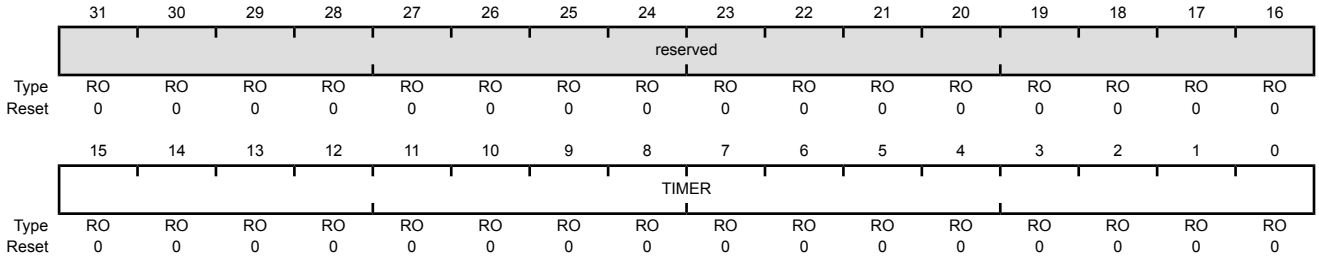

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TSS | RO | 0x0000 | Timer Snap Shot<br>This field contains the value of the free-running timer when either the Sync Edge 5 or the Sync Edge 1 was detected. |

## Register 17: UART LIN Timer (UARTLTIM), offset 0x098

The **UARTLTIM** register contains the current timer value for the free-running timer that is used to calculate the baud rate when in LIN slave mode. The value in this register is used along with the value in the **UART LIN Snap Shot (UARTLSS)** register to adjust the baud rate to match that of the master.

UART LIN Timer (UARTLTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x098
Type RO, reset 0x0000.0000



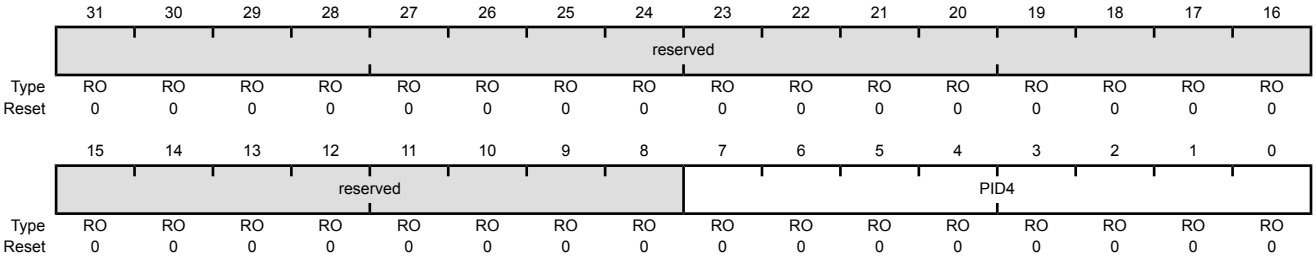

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TIMER | RO | 0x0000 | Timer Value<br>This field contains the value of the free-running timer. |

## Register 18: 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
Offset 0xFD0
Type RO, reset 0x0000.0000



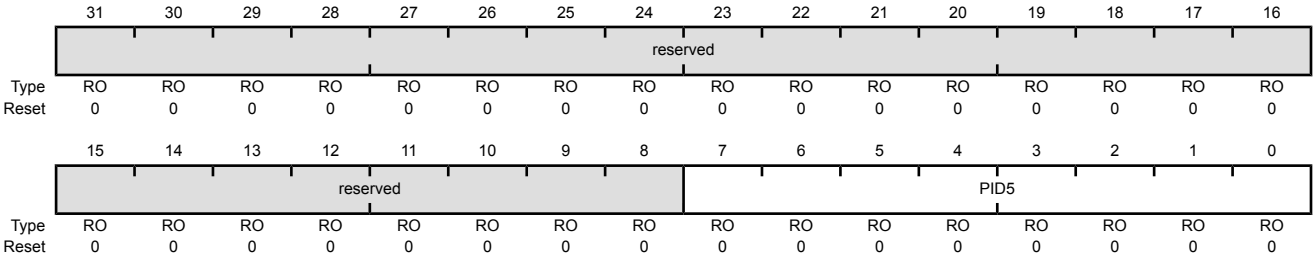

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | UART Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 19: 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
Offset 0xFD4
Type RO, reset 0x0000.0000



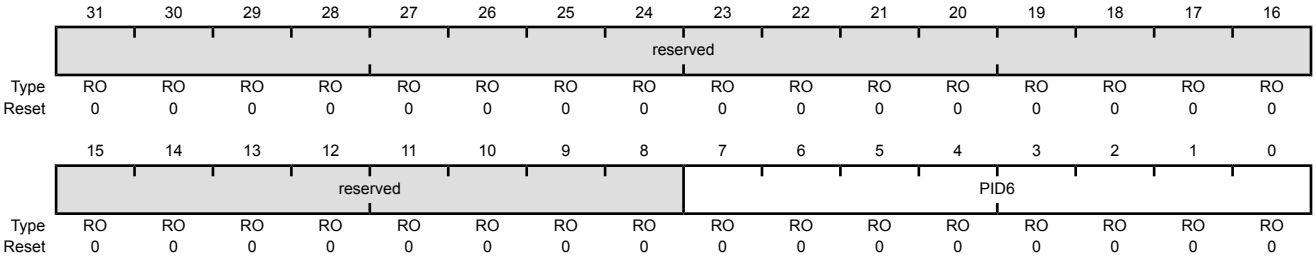

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 20: 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
Offset 0xFD8
Type RO, reset 0x0000.0000



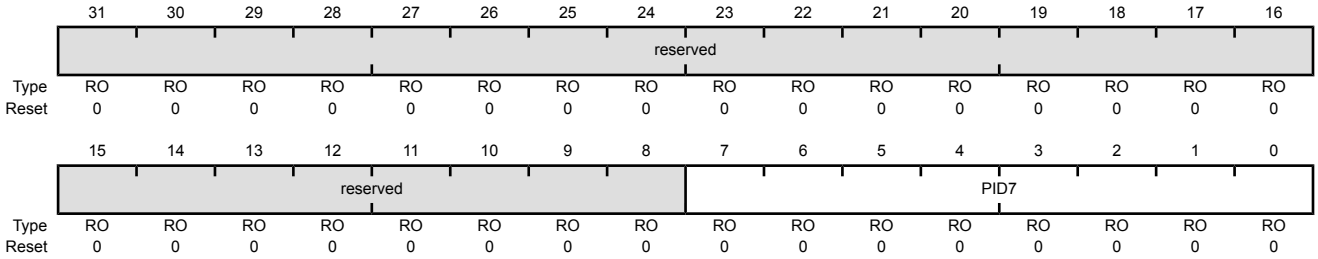

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 21: 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
Offset 0xFDC
Type RO, reset 0x0000.0000



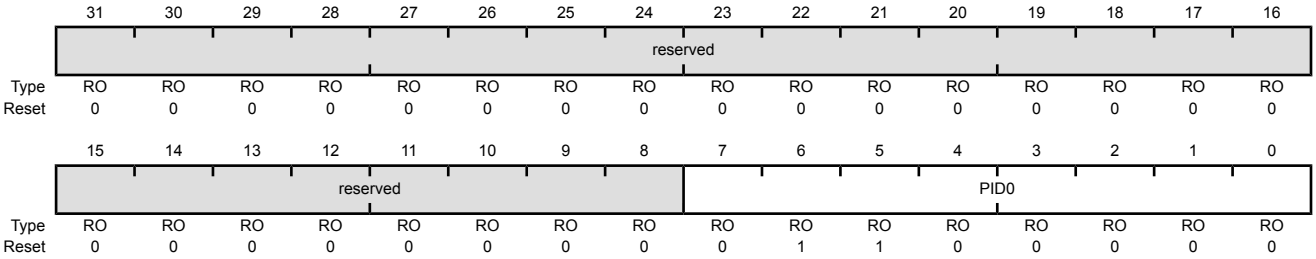

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 22: 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
Offset 0xFE0
Type RO, reset 0x0000.0060



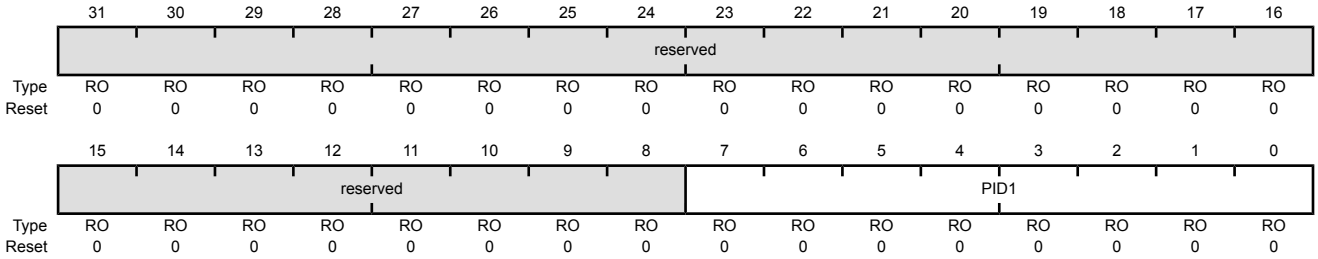

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x60 | UART Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 23: 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
Offset 0xFE4
Type RO, reset 0x0000.0000



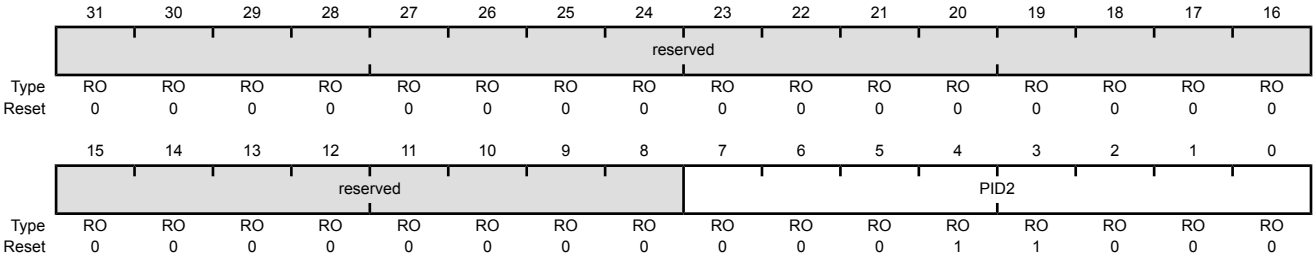

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 24: 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
Offset 0xFE8
Type RO, reset 0x0000.0018



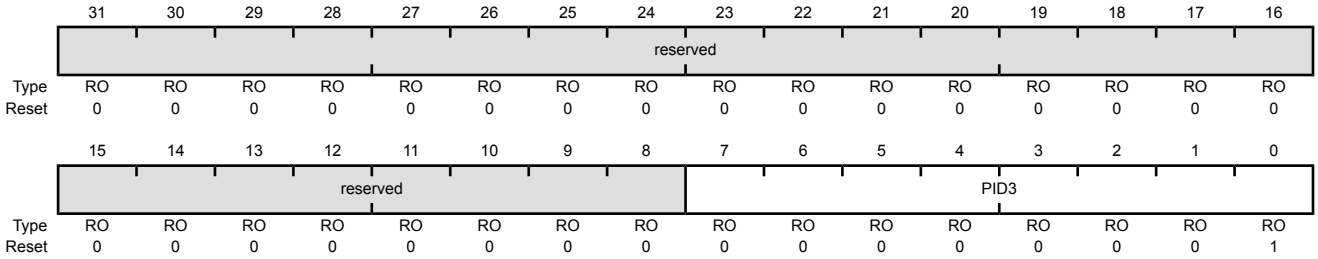

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 25: 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
Offset 0xFEC
Type RO, reset 0x0000.0001



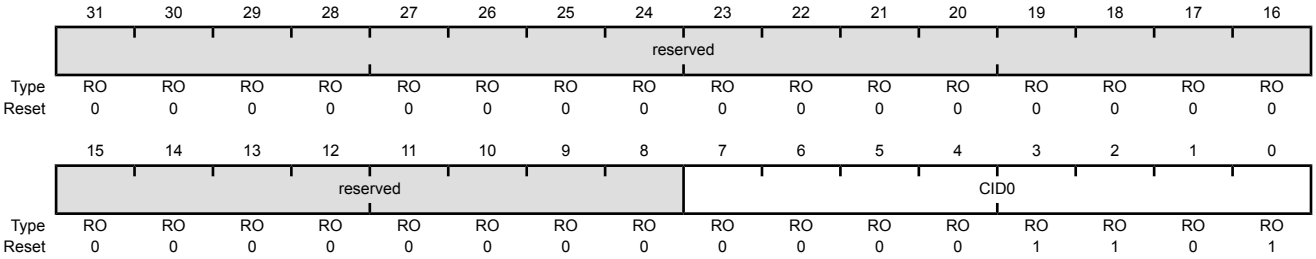

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 26: 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
Offset 0xFF0
Type RO, reset 0x0000.000D



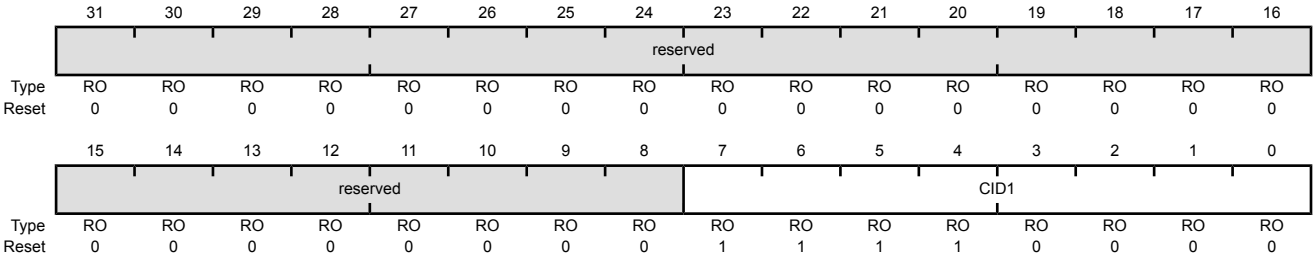

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | UART PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 27: 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
Offset 0xFF4
Type RO, reset 0x0000.00F0



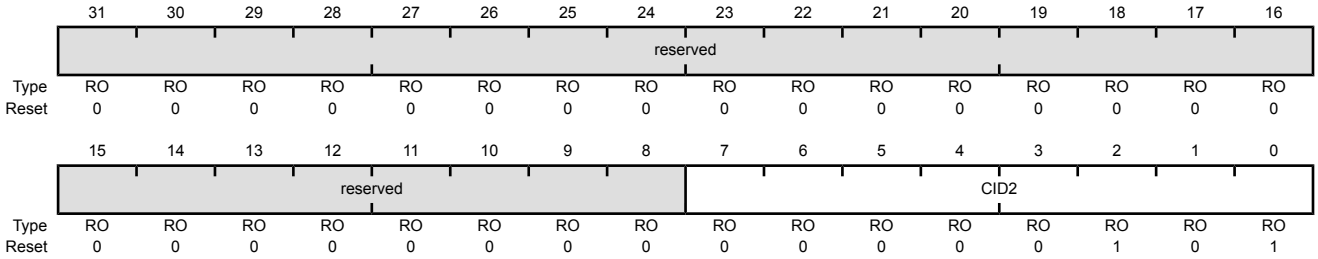

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | UART PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 28: 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
Offset 0xFF8
Type RO, reset 0x0000.0005



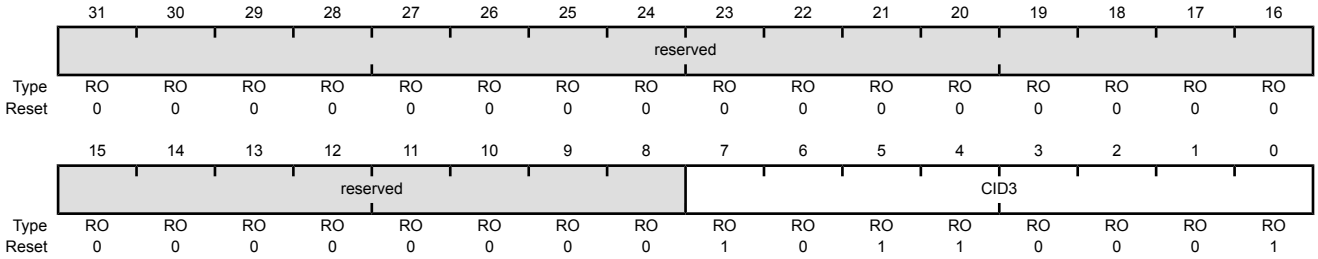

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | UART PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 29: 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
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. |

# 14 Synchronous Serial Interface (SSI)

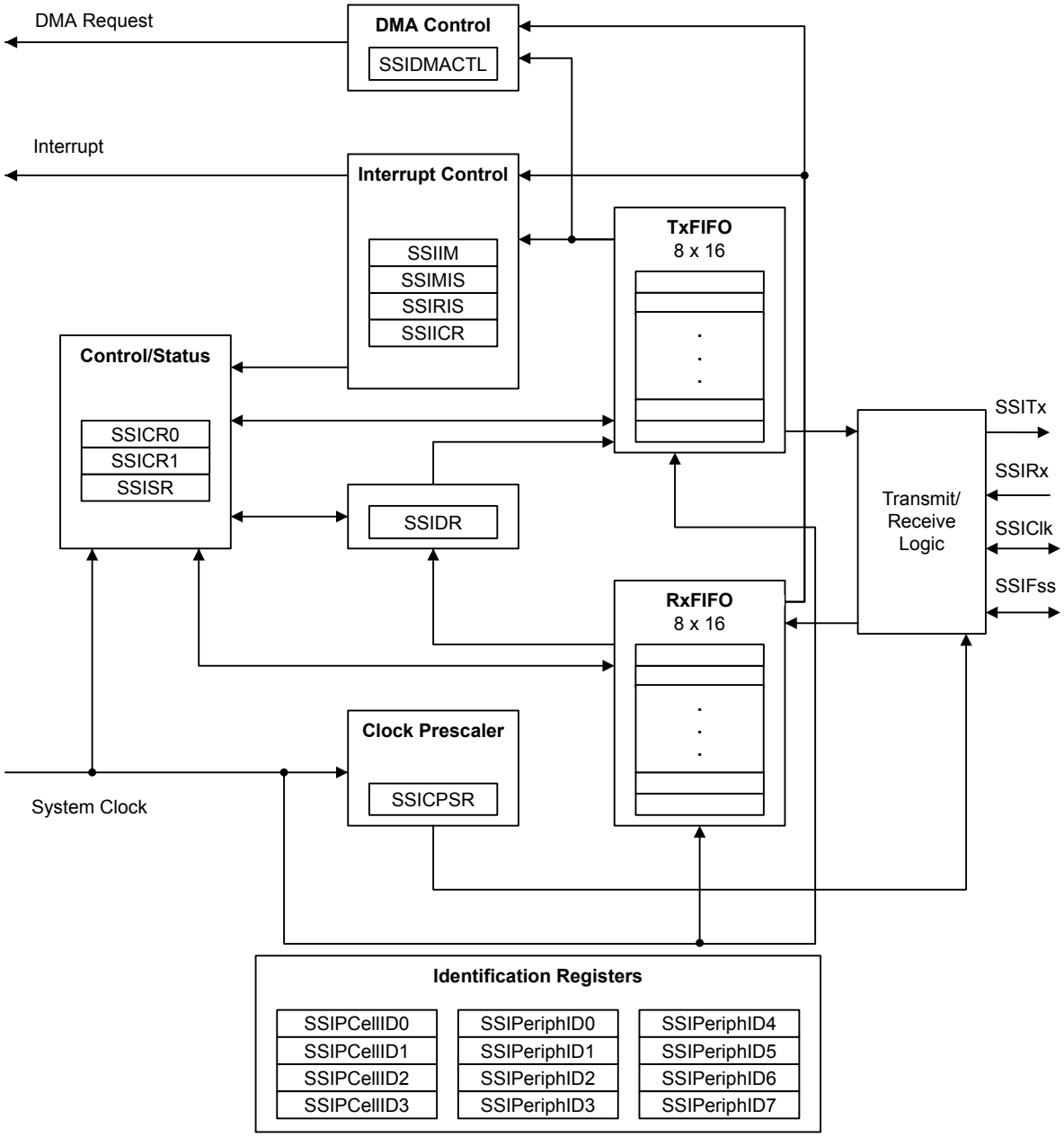
The Stellaris® microcontroller includes two Synchronous Serial Interface (SSI) modules. Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris LM3S1G58 controller includes two SSI modules with the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- 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 FIFO contains 4 entries

## 14.1 Block Diagram

**Figure 14-1. SSI Module Block Diagram**




## 14.2 Signal Description

The following table lists the external signals of the SSI module and describes the function of each. The SSI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the SSI0Clk, SSI0Fss, SSI0Rx, and SSI0Tx pins which default to the SSI function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the SSI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 418) should be set to choose the SSI function. The number in

parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 435) to assign the SSI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 395.

**Table 14-1. Signals for SSI (100LQFP)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| SSI0Clk | 28 | PA2 (1) | I/O | TTL | SSI module 0 clock. |
| SSI0Fss | 29 | PA3 (1) | I/O | TTL | SSI module 0 frame. |
| SSI0Rx | 30 | PA4 (1) | I | TTL | SSI module 0 receive. |
| SSI0Tx | 31 | PA5 (1) | O | TTL | SSI module 0 transmit. |
| SSI1Clk | 60<br>72 | PF2 (9)<br>PE0 (2) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | 59<br>73 | PF3 (9)<br>PE1 (2) | I/O | TTL | SSI module 1 frame. |
| SSI1Rx | 58<br>74 | PF4 (9)<br>PE2 (2) | I | TTL | SSI module 1 receive. |
| SSI1Tx | 46<br>75 | PF5 (9)<br>PE3 (2) | O | TTL | SSI module 1 transmit. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 14-2. Signals for SSI (108BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| SSI0Clk | M4 | PA2 (1) | I/O | TTL | SSI module 0 clock. |
| SSI0Fss | L4 | PA3 (1) | I/O | TTL | SSI module 0 frame. |
| SSI0Rx | L5 | PA4 (1) | I | TTL | SSI module 0 receive. |
| SSI0Tx | M5 | PA5 (1) | O | TTL | SSI module 0 transmit. |
| SSI1Clk | J11<br>A11 | PF2 (9)<br>PE0 (2) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | J12<br>B12 | PF3 (9)<br>PE1 (2) | I/O | TTL | SSI module 1 frame. |
| SSI1Rx | L9<br>B11 | PF4 (9)<br>PE2 (2) | I | TTL | SSI module 1 receive. |
| SSI1Tx | L8<br>A12 | PF5 (9)<br>PE3 (2) | O | TTL | SSI module 1 transmit. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 14.3 Functional Description

The SSI 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 SSI 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 688).

### 14.3.1 Bit Rate Generation

The SSI 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 681). 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 674).

The frequency of the output clock SSIClk is defined by:

```
SSIClk = SysClk / (CPSDVSR * (1 + SCR))
```

**Note:** For master mode, the system clock must be at least two times faster than the SSIClk, with the restriction that SSIClk cannot be faster than 25 MHz. For slave mode, the system clock must be at least 12 times faster than the SSIClk.

See “Synchronous Serial Interface (SSI)” on page 797 to view SSI timing parameters.

### 14.3.2 FIFO Operation

#### 14.3.2.1 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 678), 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 serial conversion and transmission to the attached slave or master, respectively, through the SSITx pin.

In slave mode, the 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 SSI bit in the **RGCG1** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The SSI can be configured to generate an interrupt or a µDMA request when the FIFO is empty.

#### 14.3.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the SSIRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

### 14.3.3 Interrupts

The SSI 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

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI generates a single interrupt request to the controller regardless of the number of active interrupts. Each of the four individual maskable interrupts can be masked by clearing the appropriate bit in the **SSI Interrupt Mask (SSIIM)** register (see page 682). 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 683 and page 685, respectively).

The receive FIFO has a time-out period that is 32 periods at the rate of SSIClk (whether or not SSIClk is currently active) and is started when the RX FIFO goes from EMPTY to not-EMPTY. If the RX FIFO is emptied before 32 clocks have passed, the time-out period is reset. As a result, the ISR should clear the Receive FIFO Time-out Interrupt just after reading out the RX FIFO by writing a 1 to the RTIC bit in the **SSI Interrupt Clear (SSIICR)** register. The interrupt should not be cleared so late that the ISR returns before the interrupt is actually cleared, or the ISR may be re-activated unnecessarily.

The End-of-Transmission (EOT) interrupt indicates that the data has been transmitted completely. This interrupt can be used to indicate when it is safe to turn off the SSI 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.

### 14.3.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (SSIClk) is held inactive while the SSI is idle, and SSIClk transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSIClk is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (SSIFss) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the SSIFss 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 SSI and the off-chip slave device drive their output data on the rising edge of SSIClk and latch data from the other device on the falling edge.

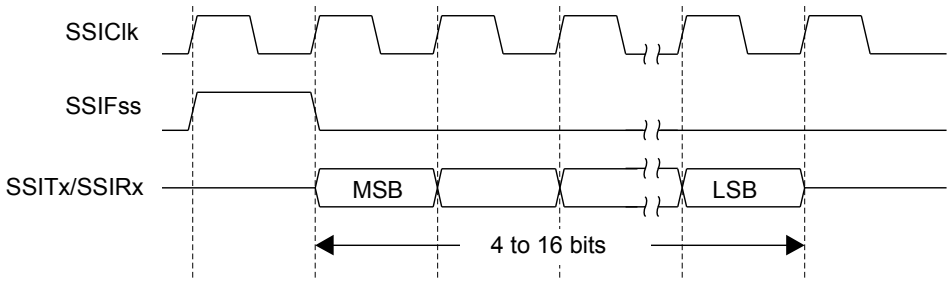
Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no

incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

#### 14.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 14-2 on page 664 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

**Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)**

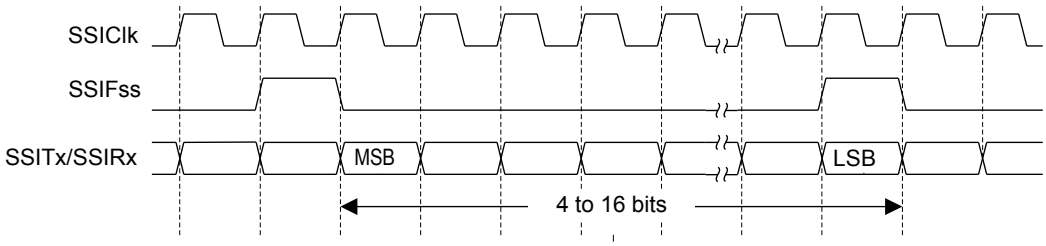



In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4 to 16-bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of SSIClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSIClk after the LSB has been latched.

Figure 14-3 on page 664 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

**Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer)**



### 14.3.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits in the **SSISCR0** control register.

#### *SPO Clock Polarity Bit*

When the SPO clock polarity control bit is clear, it produces a steady state Low value on the SSIClk pin. If the SPO bit is set, a steady state High value is placed on the SSIClk pin when data is not being transferred.

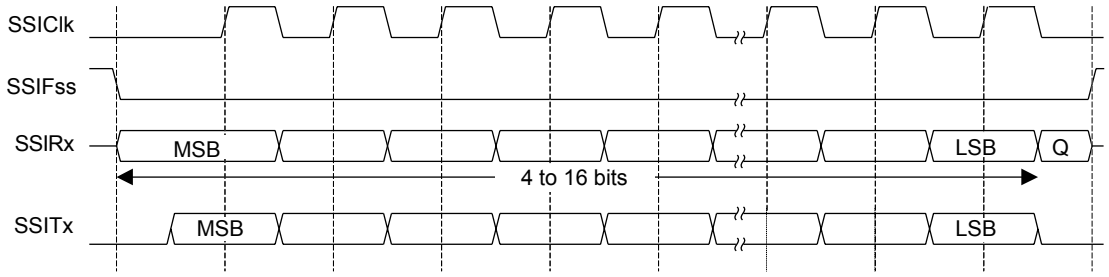
#### *SPH Phase Control Bit*

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.

### 14.3.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

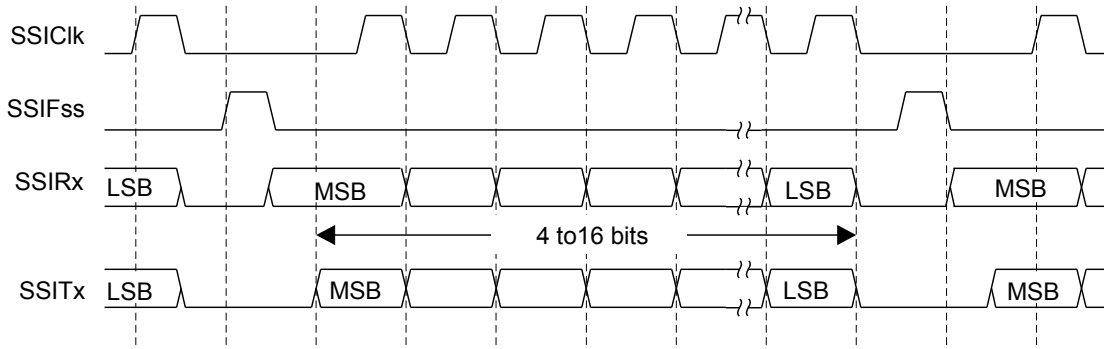
Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 14-4 on page 665 and Figure 14-5 on page 665.

**Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0**




**Note:** Q is undefined.

**Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0**




In this configuration, during idle periods:

- SSIClk is forced Low

- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, causing slave data to be enabled onto the SSIRx input line of the master. The master SSITx output pad is enabled.

One half SSIClk period later, valid master data is transferred to the SSITx pin. Once both the master and slave data have been set, the SSIClk master clock pin goes High after one additional half SSIClk period.

The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

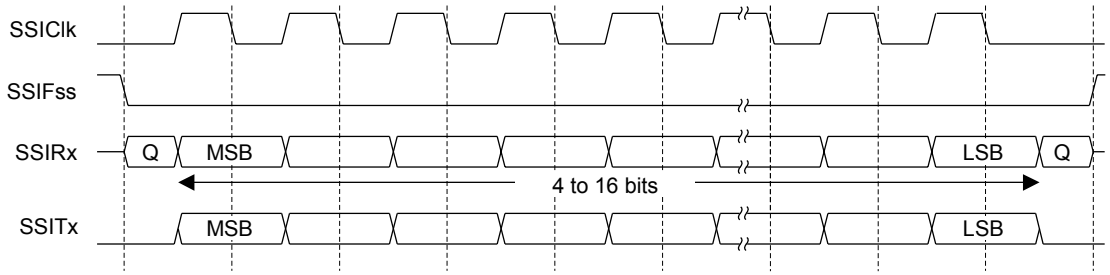
In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss 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 SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

#### 14.3.4.4 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 14-6 on page 666, which covers both single and continuous transfers.

**Figure 14-6. Freescale SPI Frame Format with SPO=0 and SPH=1**




**Note:** Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad

- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output is enabled. After an additional one-half SSIClk period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the SSIClk is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

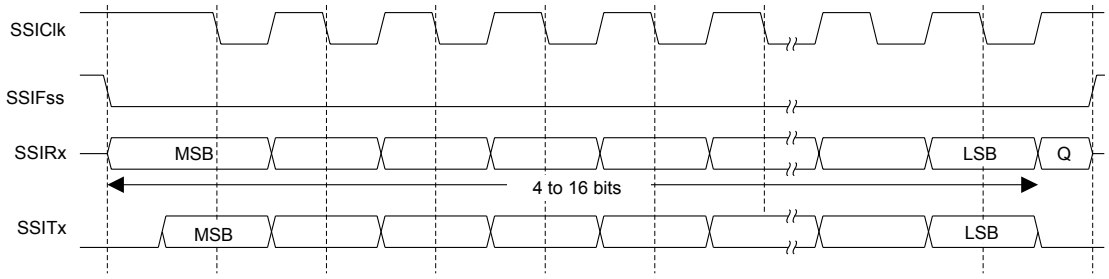
In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words, and termination is the same as that of the single word transfer.

### 14.3.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

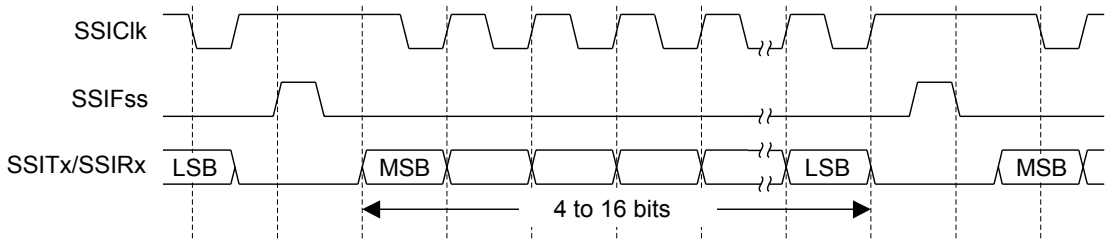
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 14-7 on page 667 and Figure 14-8 on page 667.

**Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0**




**Note:** Q is undefined.

**Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0**




In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, causing slave data to be immediately transferred onto the SSIRx line of the master. The master SSITx output pad is enabled.

One-half period later, valid master data is transferred to the SSITx line. Once both the master and slave data have been set, the SSIClk master clock pin becomes Low after one additional half SSIClk period, meaning that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

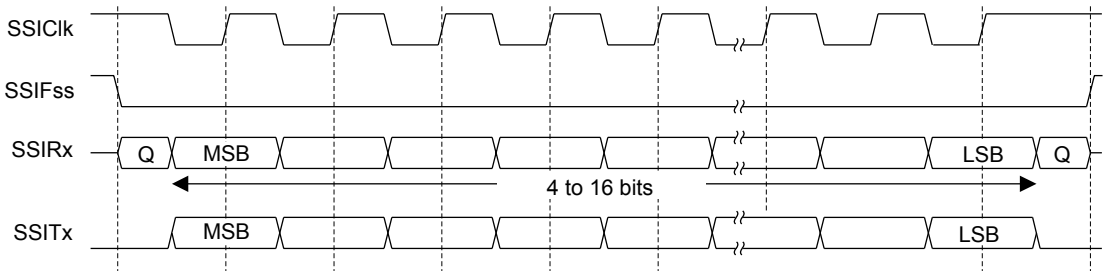
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss 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 SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

### 14.3.4.6 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 14-9 on page 668, which covers both single and continuous transfers.

**Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1**




**Note:** Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output pad is enabled. After an additional one-half SSIClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSIClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSIClk signal.

After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

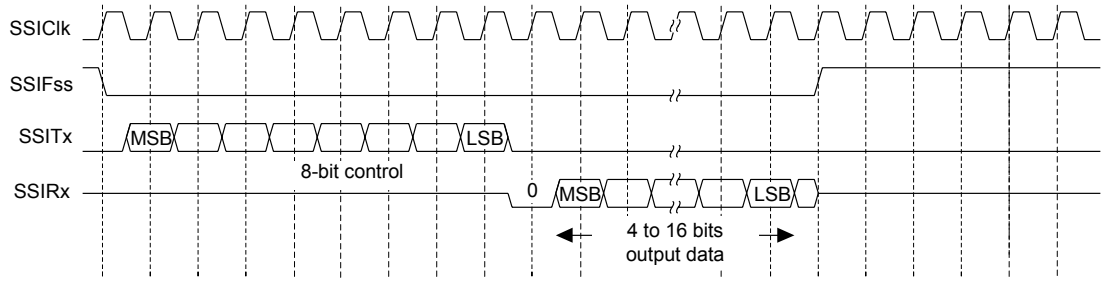
For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

#### 14.3.4.7 MICROWIRE Frame Format

Figure 14-10 on page 669 shows the MICROWIRE frame format for a single frame. Figure 14-11 on page 670 shows the same format when back-to-back frames are transmitted.

**Figure 14-10. MICROWIRE Frame Format (Single Frame)**




MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex and uses a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

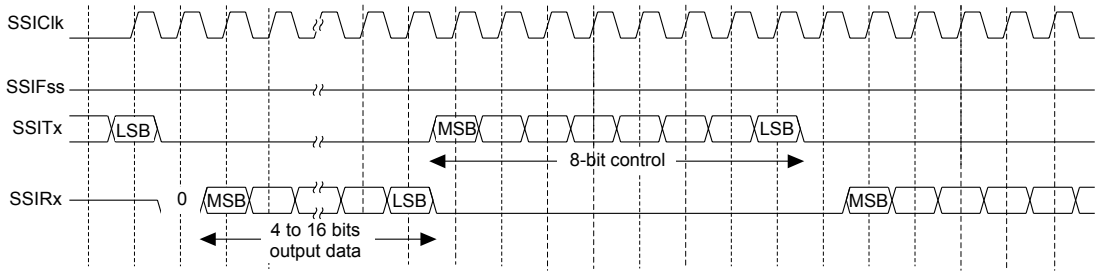
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic and the MSB of the 8-bit control frame to be shifted out onto the SSITx pin. SSIFss remains Low for the duration of the frame transmission. The SSIRx pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on each rising edge of SSIClk. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the SSIRx line on the falling edge of SSIClk. The SSI in turn latches each bit on the rising edge of SSIClk. At the end of the frame, for single transfers, the SSIFss signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, causing the data to be transferred to the receive FIFO.

**Note:** The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter or when the SSIFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.

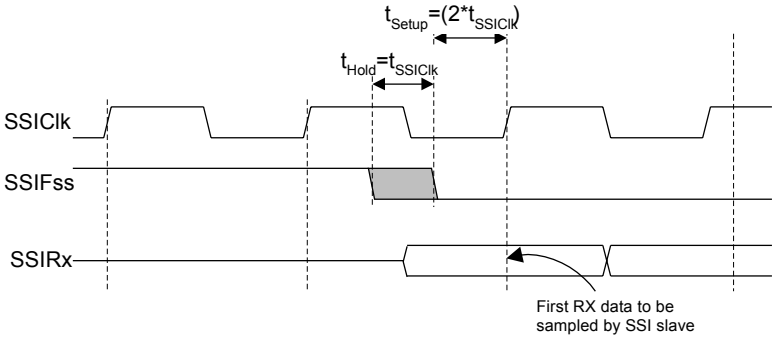
**Figure 14-11. MICROWIRE Frame Format (Continuous Transfer)**




In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of SSIClk after SSIFss has gone Low. Masters that drive a free-running SSIClk must ensure that the SSIFss signal has sufficient setup and hold margins with respect to the rising edge of SSIClk.

Figure 14-12 on page 670 illustrates these setup and hold time requirements. With respect to the SSIClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSIFss must have a setup of at least two times the period of SSIClk on which the SSI operates. With respect to the SSIClk rising edge previous to this edge, SSIFss must have a hold of at least one SSIClk period.

**Figure 14-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements**




## 14.3.5 DMA Operation

The SSI peripheral provides an interface to the µDMA controller with separate channels for transmit and receive. The µDMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When µDMA operation is enabled, the SSI 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. To enable µDMA operation for the transmit channel, the TXDMAE bit of **SSIDMACTL** should be set. If µDMA is enabled, then the µDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and µDMA is enabled, the SSI interrupt handler must be designed to handle the µDMA completion interrupt.

See "Micro Direct Memory Access (µDMA)" on page 335 for more details about programming the µDMA controller.

## 14.4 Initialization and Configuration

To enable and initialize the SSI, the following steps are necessary:

1. Enable the SSI module by setting the SSI bit in the **RCGC1** register (see page 243).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 251). To find out which GPIO port to enable, refer to Table 17-5 on page 757.
3. Set the GPIO AFSEL bits for the appropriate pins (see page 418). To determine which GPIOs to configure, see Table 17-4 on page 753.
4. Configure the PMCn fields in the **GPIOPCTL** register to assign the SSI signals to the appropriate pins. See page 435 and Table 17-5 on page 757.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the SSE bit in the **SSICR1** register is clear before making any configuration changes.
2. Select whether the SSI 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 clock prescale divisor by writing the **SSICPSR** register.
4. 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, TI SSF, MICROWIRE (FRF)
   - The data size (DSS)
5. Optionally, configure the µDMA channel (see "Micro Direct Memory Access (µDMA)" on page 335) and enable the DMA option(s) in the **SSIDMACTL** register.
6. Enable the SSI by setting the SSE bit in the **SSICR1** register.

As an example, assume the SSI 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:

```
SSIClk = SysClk / (CPSDVSR * (1 + SCR))
1x106  = 20x106 / (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 SSI is then enabled by setting the SSE bit in the **SSICR1** register.

## 14.5 Register Map

Table 14-3 on page 672 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000
- SSI1: 0x4000.9000

Note that the SSI module clock must be enabled before the registers can be programmed (see page 243). There must be a delay of 3 system clocks after the SSI module clock is enabled before any SSI module registers are accessed.

**Note:** The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

**Table 14-3. SSI Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x000 | SSICR0 | R/W | 0x0000.0000 | SSI Control 0 | 674 |
| 0x004 | SSICR1 | R/W | 0x0000.0000 | SSI Control 1 | 676 |
| 0x008 | SSIDR | R/W | 0x0000.0000 | SSI Data | 678 |
| 0x00C | SSISR | RO | 0x0000.0003 | SSI Status | 679 |
| 0x010 | SSICPSR | R/W | 0x0000.0000 | SSI Clock Prescale | 681 |
| 0x014 | SSIIM | R/W | 0x0000.0000 | SSI Interrupt Mask | 682 |

**Table 14-3. SSI Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x018 | SSIRIS | RO | 0x0000.0008 | SSI Raw Interrupt Status | 683 |
| 0x01C | SSIMIS | RO | 0x0000.0000 | SSI Masked Interrupt Status | 685 |
| 0x020 | SSIICR | W1C | 0x0000.0000 | SSI Interrupt Clear | 687 |
| 0x024 | SSIDMACTL | R/W | 0x0000.0000 | SSI DMA Control | 688 |
| 0xFD0 | SSIPeriphID4 | RO | 0x0000.0000 | SSI Peripheral Identification 4 | 689 |
| 0xFD4 | SSIPeriphID5 | RO | 0x0000.0000 | SSI Peripheral Identification 5 | 690 |
| 0xFD8 | SSIPeriphID6 | RO | 0x0000.0000 | SSI Peripheral Identification 6 | 691 |
| 0xFDC | SSIPeriphID7 | RO | 0x0000.0000 | SSI Peripheral Identification 7 | 692 |
| 0xFE0 | SSIPeriphID0 | RO | 0x0000.0022 | SSI Peripheral Identification 0 | 693 |
| 0xFE4 | SSIPeriphID1 | RO | 0x0000.0000 | SSI Peripheral Identification 1 | 694 |
| 0xFE8 | SSIPeriphID2 | RO | 0x0000.0018 | SSI Peripheral Identification 2 | 695 |
| 0xFEC | SSIPeriphID3 | RO | 0x0000.0001 | SSI Peripheral Identification 3 | 696 |
| 0xFF0 | SSIPCellID0 | RO | 0x0000.000D | SSI PrimeCell Identification 0 | 697 |
| 0xFF4 | SSIPCellID1 | RO | 0x0000.00F0 | SSI PrimeCell Identification 1 | 698 |
| 0xFF8 | SSIPCellID2 | RO | 0x0000.0005 | SSI PrimeCell Identification 2 | 699 |
| 0xFFC | SSIPCellID3 | RO | 0x0000.00B1 | SSI PrimeCell Identification 3 | 700 |

## 14.6 Register Descriptions

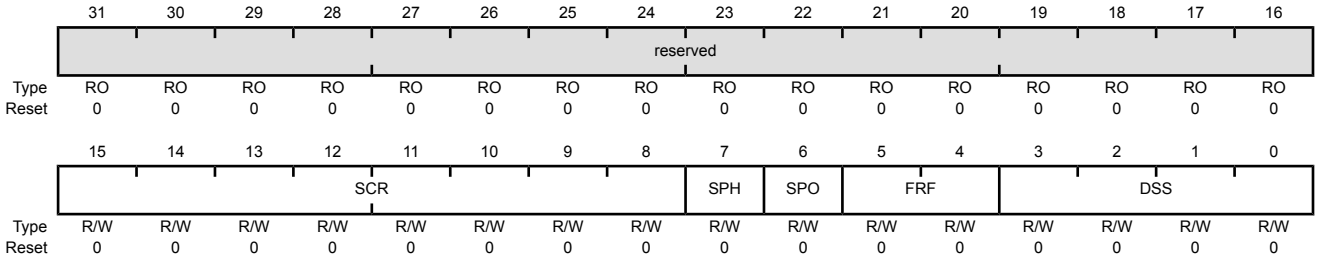
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

## Register 1: SSI Control 0 (SSICR0), offset 0x000

The **SSICR0** register contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x000
Type R/W, 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 | R/W | 0x00 | SSI Serial Clock Rate<br>This bit field is used to generate the transmit and receive bit rate of the SSI. 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 | R/W | 0 | SSI 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 | R/W | 0 | SSI Serial Clock Polarity<br>Value Description<br>0 A steady state Low value is placed on the `SSIClk` pin.<br>1 A steady state High value is placed on the `SSIClk` pin when data is not being transferred. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 5:4 | FRF | R/W | 0x0 | SSI Frame Format Select |

| Value | Frame Format |
|---|---|
| 0x0 | Freescale SPI Frame Format |
| 0x1 | Texas Instruments Synchronous Serial Frame Format |
| 0x2 | MICROWIRE Frame Format |
| 0x3 | Reserved |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 3:0 | DSS | R/W | 0x0 | SSI Data Size Select |

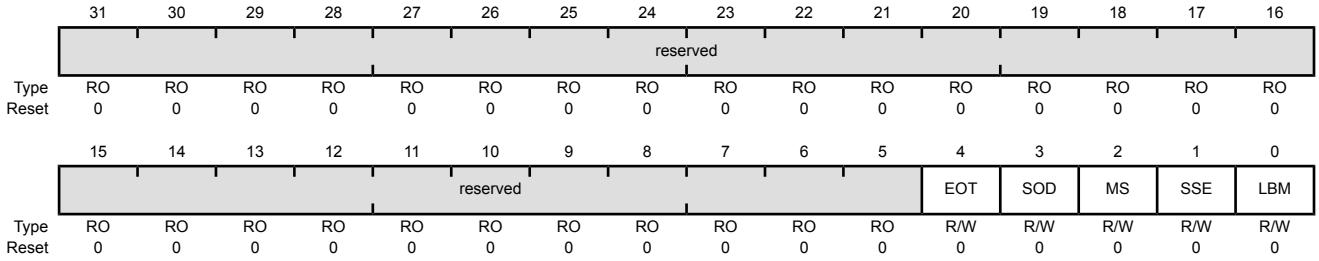
| Value | Data Size |
|---|---|
| 0x0-0x2 | Reserved |
| 0x3 | 4-bit data |
| 0x4 | 5-bit data |
| 0x5 | 6-bit data |
| 0x6 | 7-bit data |
| 0x7 | 8-bit data |
| 0x8 | 9-bit data |
| 0x9 | 10-bit data |
| 0xA | 11-bit data |
| 0xB | 12-bit data |
| 0xC | 13-bit data |
| 0xD | 14-bit data |
| 0xE | 15-bit data |
| 0xF | 16-bit data |

## Register 2: SSI Control 1 (SSICR1), offset 0x004

The **SSICR1** register contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x004
Type R/W, reset 0x0000.0000




| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:5 | 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. |
| 4 | EOT | R/W | 0 | End of Transmission<br><br>Value Description<br>0 The TXRIS interrupt indicates that the transmit FIFO is half full or less.<br>1 The End of Transmit interrupt mode for the TXRIS interrupt is enabled. |
| 3 | SOD | R/W | 0 | SSI Slave Mode Output Disable<br>This bit is relevant only in the Slave mode (MS=1). In multiple-slave systems, it is possible for the SSI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto the serial output line. In such systems, the TXD lines from multiple slaves could be tied together. To operate in such a system, the SOD bit can be configured so that the SSI slave does not drive the SSITx pin.<br><br>Value Description<br>0 SSI can drive the SSITx output in Slave mode.<br>1 SSI must not drive the SSITx output in Slave mode. |
| 2 | MS | R/W | 0 | SSI Master/Slave Select<br>This bit selects Master or Slave mode and can be modified only when the SSI is disabled (SSE=0).<br><br>Value Description<br>0 The SSI is configured as a master.<br>1 The SSI is configured as a slave. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SSE | R/W | 0 | SSI Synchronous Serial Port Enable<br><br>Value Description<br>0 SSI operation is disabled.<br>1 SSI operation is enabled.<br><br>**Note:** This bit must be cleared before any control registers are reprogrammed. |
| 0 | LBM | R/W | 0 | SSI Loopback Mode<br><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: SSI 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 SSI 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.

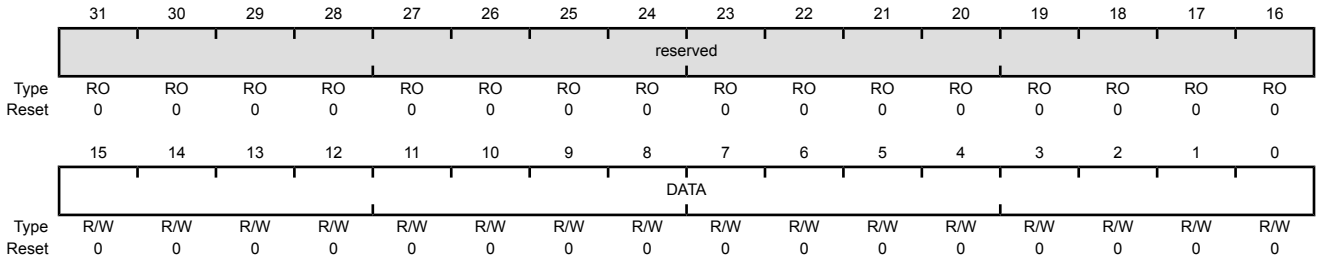
When the **SSIDR** register is written to, the entry in the transmit FIFO that is pointed to by the write pointer is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. Each data value is loaded into the transmit serial shifter, then serially shifted out onto the SSITx 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.

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the SSE bit in the **SSICR1** register is cleared, allowing the software to fill the transmit FIFO before enabling the SSI.

SSI Data (SSIDR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x008
Type R/W, reset 0x0000.0000



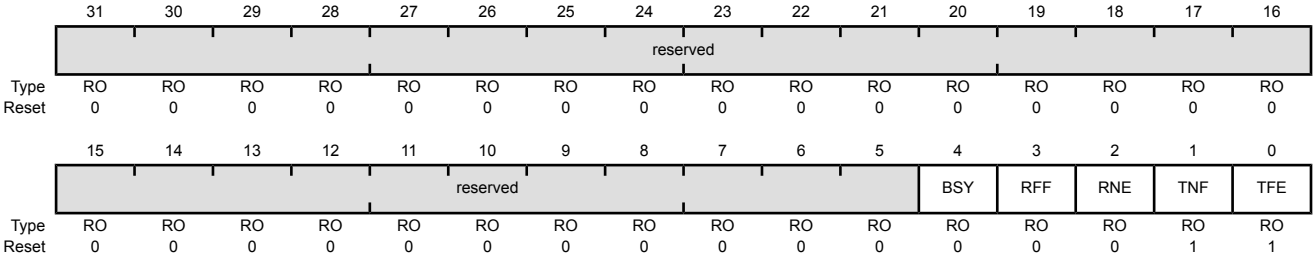

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | DATA | R/W | 0x0000 | SSI 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 SSI 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: SSI Status (SSISR), offset 0x00C

The **SSISR** register contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x00C
Type RO, reset 0x0000.0003




| 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 | SSI Busy Bit<br>Value Description<br>0 The SSI is idle.<br>1 The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty. |
| 3 | RFF | RO | 0 | SSI 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 | SSI 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 | SSI 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 | SSI Transmit FIFO Empty |

| Value | Description |
|---|---|
| 0 | The transmit FIFO is not empty. |
| 1 | The transmit FIFO is empty. |

## Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

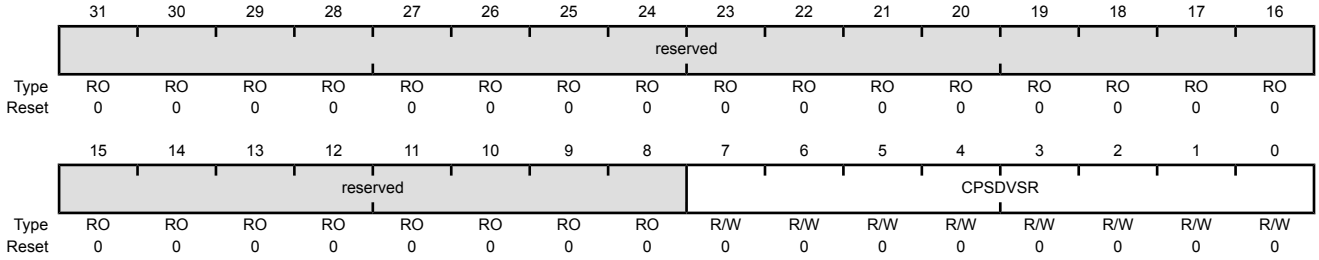
The **SSICPSR** register specifies the division factor which is used to derive the SSIClk from the system clock. The clock is further divided by a value from 1 to 256, which is 1 + SCR. SCR is programmed in the **SSICR0** register. The frequency of the SSIClk is defined by:

```
SSIClk = 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.

SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x010
Type R/W, 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 | R/W | 0x00 | SSI Clock Prescale Divisor<br>This value must be an even number from 2 to 254, depending on the frequency of SSIClk. The LSB always returns 0 on reads. |

## Register 6: SSI 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 sets the mask, preventing the interrupt from being signaled to the interrupt controller. Clearing a bit clears the corresponding mask, enabling the interrupt to be sent to the interrupt controller.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x014
Type R/W, 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 | | | | | | | | | | | | TXIM | RXIM | RTIM | RORIM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

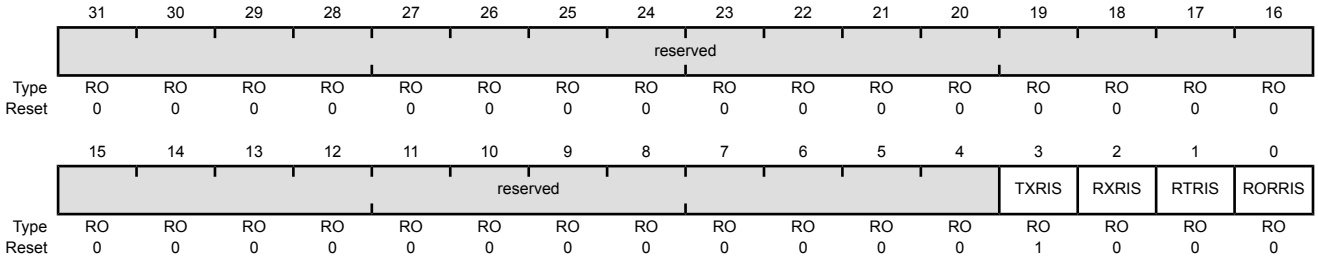
| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:4 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | TXIM | R/W | 0 | SSI Transmit FIFO Interrupt Mask<br>Value Description<br>0 The transmit FIFO interrupt is masked.<br>1 The transmit FIFO interrupt is not masked. |
| 2 | RXIM | R/W | 0 | SSI 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 | R/W | 0 | SSI 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 | R/W | 0 | SSI 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: SSI 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.

SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x018
Type RO, reset 0x0000.0008




| 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 | TXRIS | RO | 1 | SSI Transmit FIFO Raw Interrupt Status<br><br>Value Description<br>0 No interrupt.<br>1 If the EOT bit in the **SSICR1** register is clear, the transmit FIFO is half full or less.<br>If the EOT bit is set, the transmit FIFO is empty, and the last bit has been transmitted out of the serializer.<br><br>This bit is cleared when the transmit FIFO is more than half full (if the EOT bit is clear) or when it has any data in it (if the EOT bit is set). |
| 2 | RXRIS | RO | 0 | SSI 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 | SSI 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RORRIS | RO | 0 | SSI Receive Overrun Raw Interrupt Status |

| Value | Description |
|---|---|
| 0 | No interrupt. |
| 1 | The receive FIFO has overflowed |

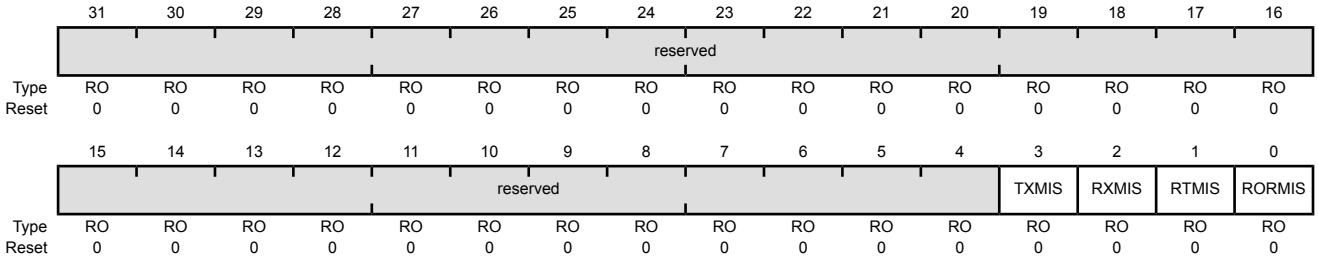
This bit is cleared when a 1 is written to the `RORIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

## Register 8: SSI 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.

SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x01C
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 | TXMIS | RO | 0 | SSI 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 full or less (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 when the transmit FIFO is more than half full (if the EOT bit is clear) or when it has any data in it (if the EOT bit is set). |
| 2 | RXMIS | RO | 0 | SSI 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 less.<br><br>This bit is cleared when the receive FIFO is less than half full. |
| 1 | RTMIS | RO | 0 | SSI 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. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RORMIS | RO | 0 | SSI Receive Overrun Masked Interrupt Status |

| Value | Description |
|---|---|
| 0 | An interrupt has not occurred or is masked. |
| 1 | An unmasked interrupt was signaled due to the receive FIFO overflowing. |

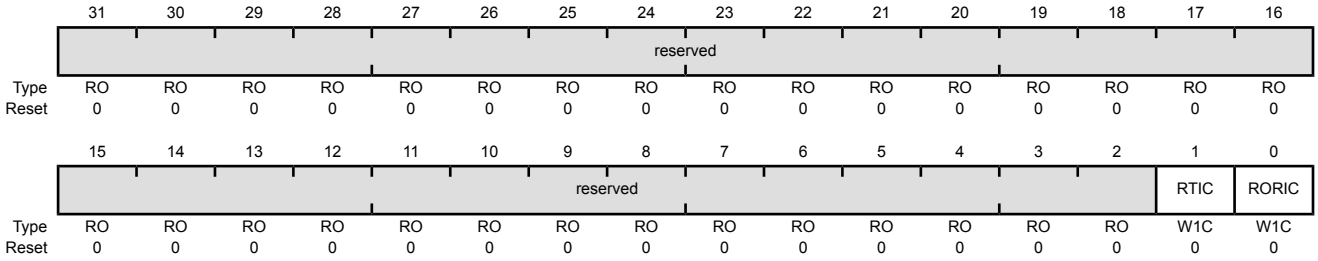
This bit is cleared when a 1 is written to the `RORIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

## Register 9: SSI 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.

SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x020
Type W1C, reset 0x0000.0000



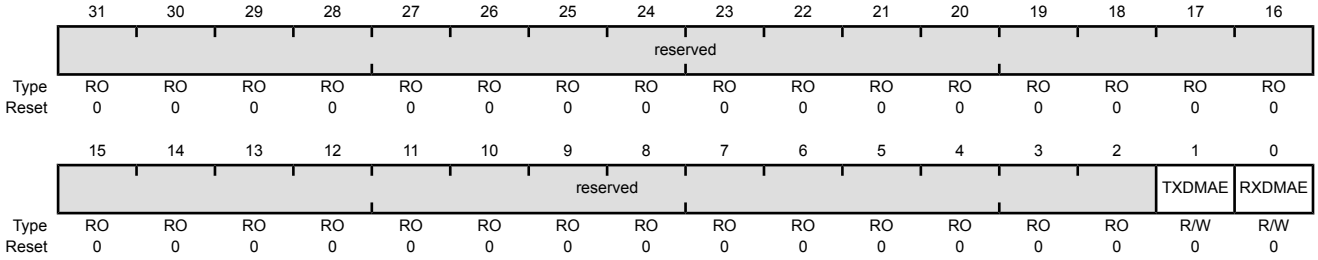

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | RTIC | W1C | 0 | SSI 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 | SSI 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: SSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the µDMA control register.

SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x024
Type R/W, reset 0x0000.0000



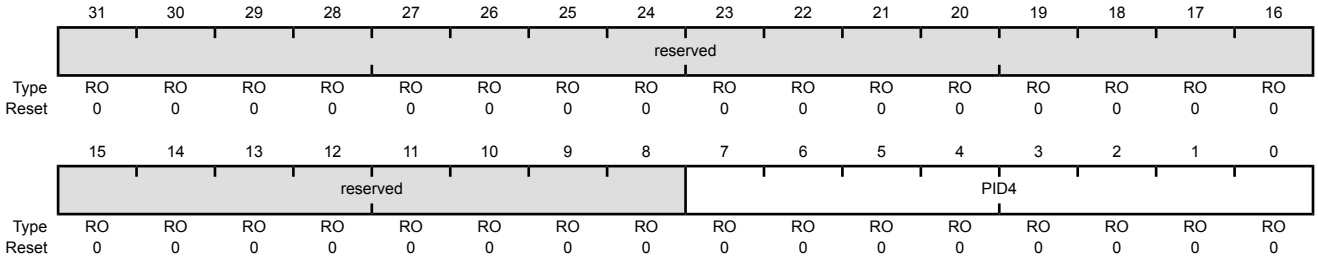

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | TXDMAE | R/W | 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 | R/W | 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: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD0
Type RO, reset 0x0000.0000



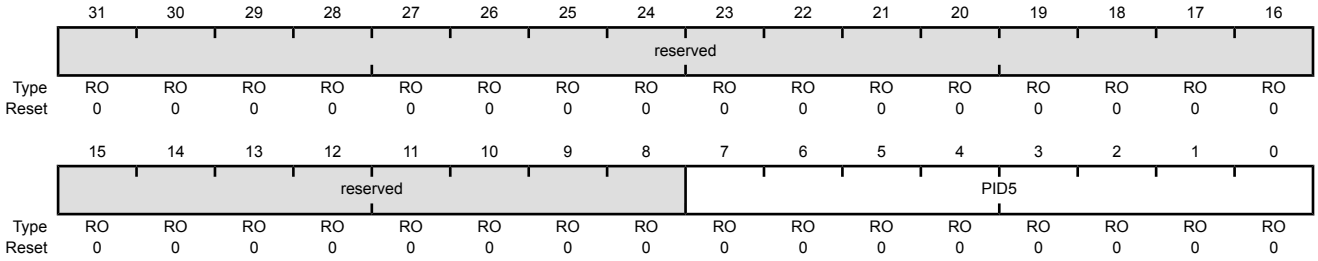

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | SSI Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD4
Type RO, reset 0x0000.0000



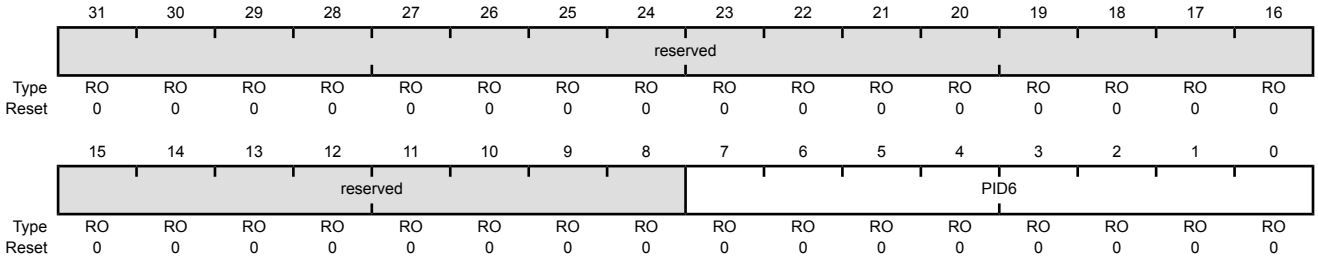

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | SSI Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD8
Type RO, reset 0x0000.0000



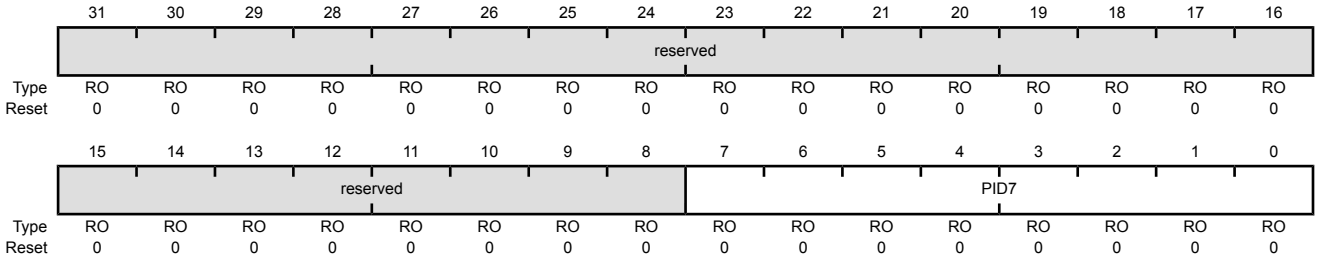

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | SSI Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFDC
Type RO, reset 0x0000.0000



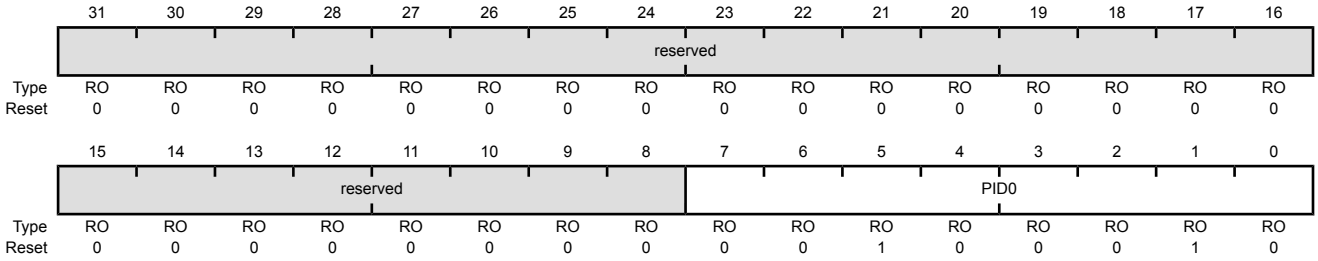

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | SSI Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE0
Type RO, reset 0x0000.0022



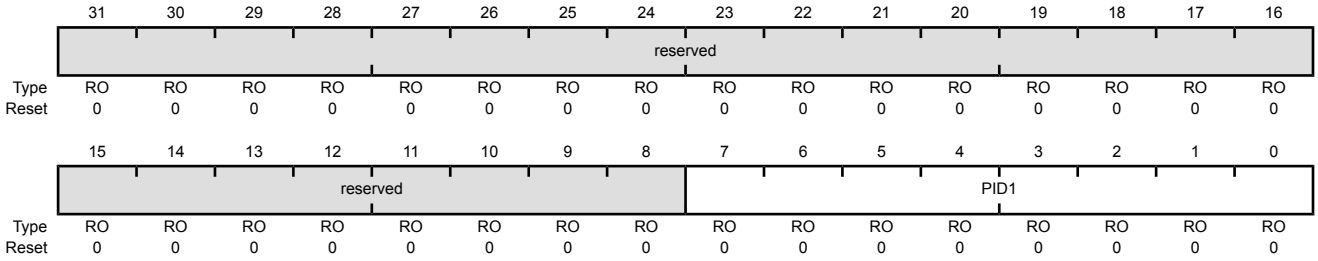

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x22 | SSI Peripheral ID Register [7:0]<br>Can be used by software to identify the presence of this peripheral. |

## Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE4
Type RO, reset 0x0000.0000



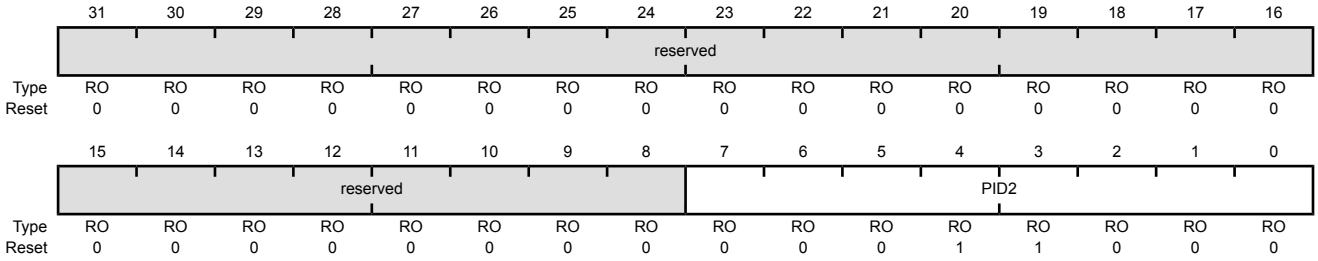

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x00 | SSI Peripheral ID Register [15:8]<br>Can be used by software to identify the presence of this peripheral. |

## Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE8
Type RO, reset 0x0000.0018



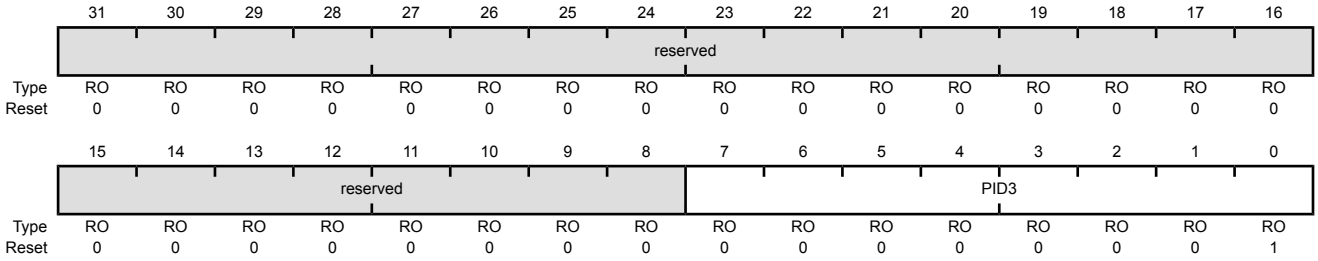

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | SSI Peripheral ID Register [23:16]<br>Can be used by software to identify the presence of this peripheral. |

## Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFEC
Type RO, reset 0x0000.0001



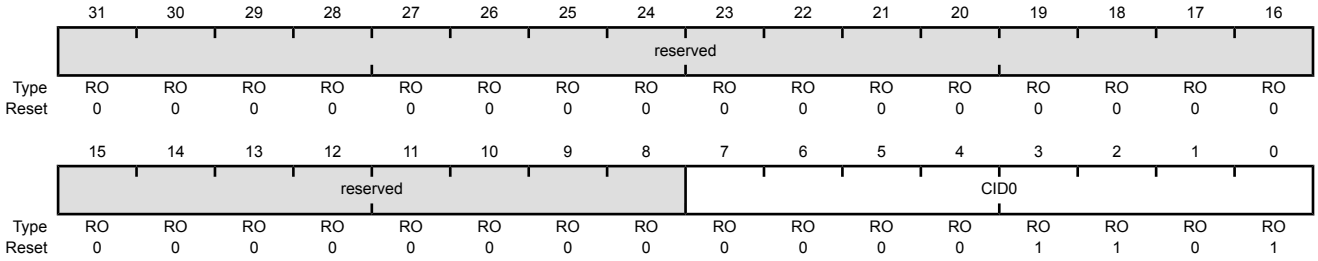

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | SSI Peripheral ID Register [31:24]<br>Can be used by software to identify the presence of this peripheral. |

## Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF0
Type RO, reset 0x0000.000D



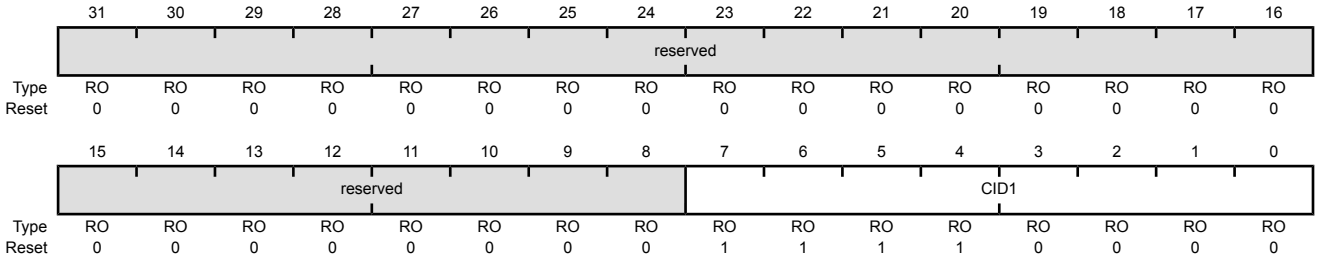

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | SSI PrimeCell ID Register [7:0]<br>Provides software a standard cross-peripheral identification system. |

## Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF4
Type RO, reset 0x0000.00F0



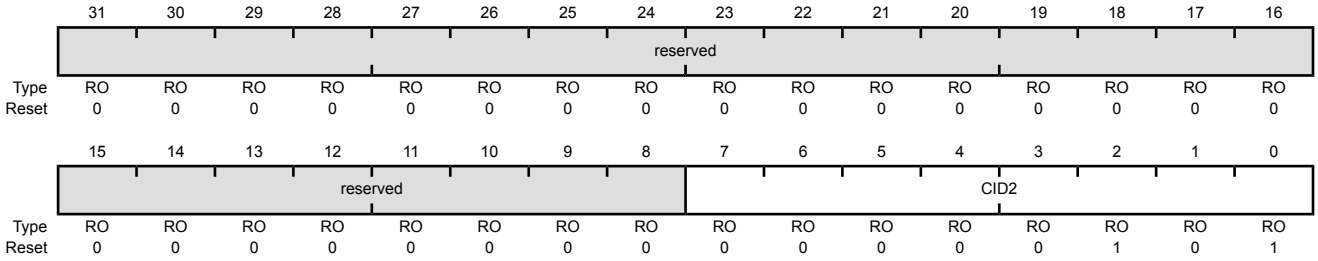

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | SSI PrimeCell ID Register [15:8]<br>Provides software a standard cross-peripheral identification system. |

## Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF8
Type RO, reset 0x0000.0005



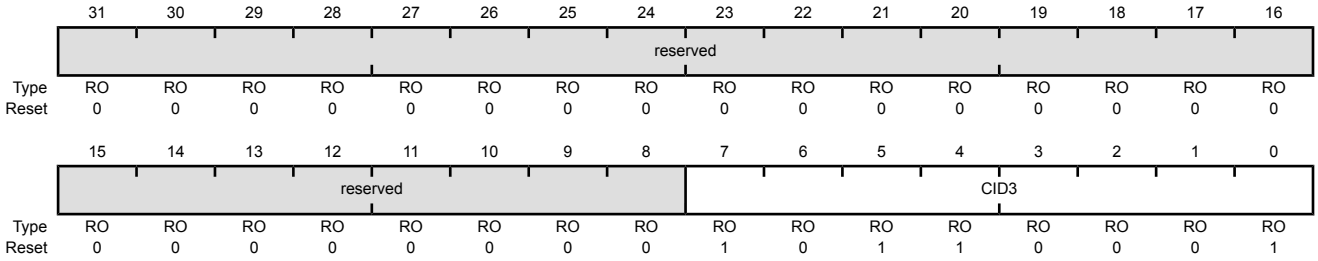

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | SSI PrimeCell ID Register [23:16]<br>Provides software a standard cross-peripheral identification system. |

## Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
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 | SSI PrimeCell ID Register [31:24]<br>Provides software a standard cross-peripheral identification system. |

# 15 Inter-Integrated Circuit ($I^2C$) Interface

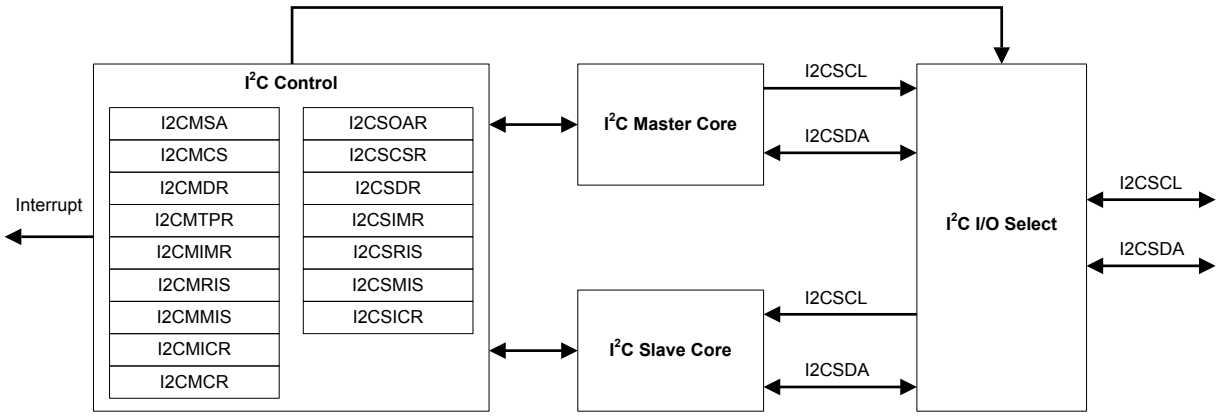
The Inter-Integrated Circuit ($I^2C$) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external $I^2C$ devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The $I^2C$ bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S1G58 microcontroller includes two $I^2C$ modules, providing the ability to interact (both transmit and receive) with other $I^2C$ devices on the bus.

The Stellaris® LM3S1G58 controller includes two $I^2C$ 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 transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
- Master and slave interrupt generation
  - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
  - Slave generates interrupts when data has been transferred or requested by a master or when a START or STOP condition is detected
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

## 15.1 Block Diagram

**Figure 15-1. I2C Block Diagram**




## 15.2 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., with the exception of the I2C0SCL and I2CSDA pins which default to the I2C function. 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 418) 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 435) to assign the I2C signal to the specified GPIO port pin. Note that the I2C pins 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 395.

**Table 15-1. Signals for I2C (100LQFP)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| I2C0SCL | 70 | PB2 (1) | I/O | OD | I2C module 0 clock. |
| I2C0SDA | 71 | PB3 (1) | I/O | OD | I2C module 0 data. |
| I2C1SCL | 19<br>26<br>34 | PG0 (3)<br>PA0 (8)<br>PA6 (1) | I/O | OD | I2C module 1 clock. |
| I2C1SDA | 18<br>27<br>35 | PG1 (3)<br>PA1 (8)<br>PA7 (1) | I/O | OD | I2C module 1 data. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 15-2. Signals for I2C (108BGA)**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| I2C0SCL | C11 | PB2 (1) | I/O | OD | I2C module 0 clock. |
| I2C0SDA | C12 | PB3 (1) | I/O | OD | I2C module 0 data. |

**Table 15-2. Signals for I2C (108BGA) *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| I2C1SCL | K1<br>L3<br>L6 | PG0 (3)<br>PA0 (8)<br>PA6 (1) | I/O | OD | I²C module 1 clock. |
| I2C1SDA | K2<br>M3<br>M6 | PG1 (3)<br>PA1 (8)<br>PA7 (1) | I/O | OD | I²C module 1 data. |

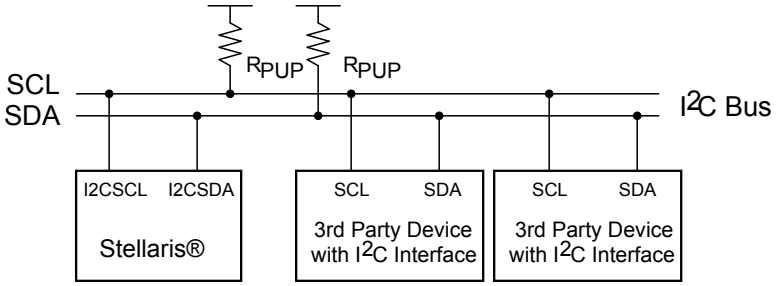
a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 15.3 Functional Description

Each I²C module is comprised of both master and slave functions. For proper operation, the SDA and SCL pins must be configured as open-drain signals. A typical I²C bus configuration is shown in Figure 15-2.

See "Inter-Integrated Circuit (I²C) Interface" on page 799 for I²C timing diagrams.

**Figure 15-2. I²C Bus Configuration**




### 15.3.1 I²C Bus Functional Overview

The I²C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on Stellaris 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 703) is unrestricted, but each 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.

#### 15.3.1.1 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 15-3.

**Figure 15-3. START and STOP Conditions**

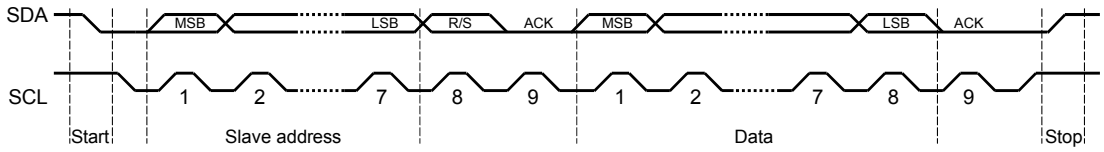
The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I2C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and the Control register is written 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, two bits in the **I2C Slave Raw Interrupt Status (I2CSRIS)** register indicate detection of start and stop conditions on the bus; while two bits in the **I2C Slave Masked Interrupt Status (I2CSMIS)** register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

### 15.3.1.2 Data Format with 7-Bit Address

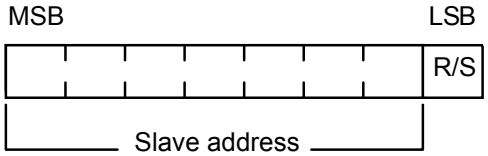
Data transfers follow the format shown in Figure 15-4. After the START condition, a slave address is transmitted. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the **I2CMSA** register). If the R/S bit is clear, it indicates a transmit operation (send), 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 15-4. Complete Data Transfer with a 7-Bit Address**




The first seven bits of the first byte make up the slave address (see Figure 15-5). 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.

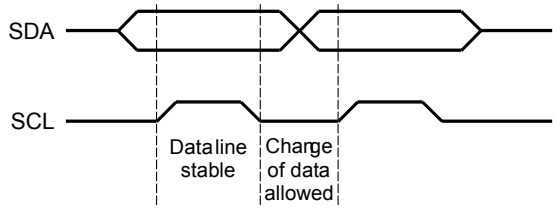
**Figure 15-5. R/S Bit in First Byte**




#### 15.3.1.3 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 15-6).

**Figure 15-6. Data Validity During Bit Transfer on the I²C Bus**




#### 15.3.1.4 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 705.

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.

#### 15.3.1.5 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.

### 15.3.2 Available Speed Modes

The I²C bus can run in either Standard mode (100 kbps) or Fast mode (400 kbps). The selected mode should match the speed of the other I²C devices on the bus.

#### 15.3.2.1 Standard and Fast Modes

Standard and Fast 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 or 400 kbps for Fast mode.

The I2C clock rate is determined by the parameters *CLK_PRD*, *TIMER_PRD*, *SCL_LP*, and *SCL_HP* where:

*CLK_PRD* is the system clock period

*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 724).

The I2C clock period is calculated as follows:

*SCL_PERIOD* = 2 × (1 + *TIMER_PRD*) × (*SCL_LP* + *SCL_HP*) × *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 15-3 gives examples of the timer periods that should be used to generate both Standard and Fast mode SCL frequencies based on various system clock frequencies.

**Table 15-3. Examples of I2C Master Timer Period versus Speed Mode**

| System Clock | Timer Period | Standard Mode | Timer Period | Fast Mode |
|---|---|---|---|---|
| 4 MHz | 0x01 | 100 Kbps | - | - |
| 6 MHz | 0x02 | 100 Kbps | - | - |
| 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 |
| 50 MHz | 0x18 | 100 Kbps | 0x06 | 357 Kbps |
| 80 MHz | 0x27 | 100 Kbps | 0x09 | 400 Kbps |

### 15.3.3 Interrupts

The I2C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master arbitration lost

- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I²C master and I²C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

#### 15.3.3.1 I²C Master Interrupts

The I²C master module generates an interrupt when a transaction completes (either transmit or receive), when arbitration is lost, or when an error occurs during a transaction. To enable the I²C master interrupt, software must set the IM bit in the **I²C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the ERROR and ARBLST bits in the **I²C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction and to ensure that arbitration has not been lost. An error condition is asserted if the last transaction wasn't acknowledged by the slave. If an error is not detected and the master has not lost arbitration, the application can proceed with the transfer. The interrupt is cleared by writing a 1 to the IC bit in the **I²C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Master Raw Interrupt Status (I2CMRIS)** register.

#### 15.3.3.2 I²C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by setting the DATAIM bit in the **I²C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I²C Slave Data (I2CSDR)** register, by checking the RREQ and TREQ bits of the **I²C Slave Control/Status (I2CSCSR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the FBR bit is set along with the RREQ bit. The interrupt is cleared by setting the DATAIC bit in the **I²C Slave Interrupt Clear (I2CSICR)** register.

In addition, the slave module can generate an interrupt when a start and stop condition is detected. These interrupts are enabled by setting the STARTIM and STOPIM bits of the **I²C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the STOPIC and STARTIC bits of the **I²C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Slave Raw Interrupt Status (I2CSRIS)** register.

### 15.3.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the LPBK bit in the **I²C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

### 15.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I2C transfer types in both master and slave mode.

#### 15.3.5.1 I2C Master Command Sequences

The figures that follow show the command sequences available for the I2C master.

**Figure 15-7. Master Single TRANSMIT**

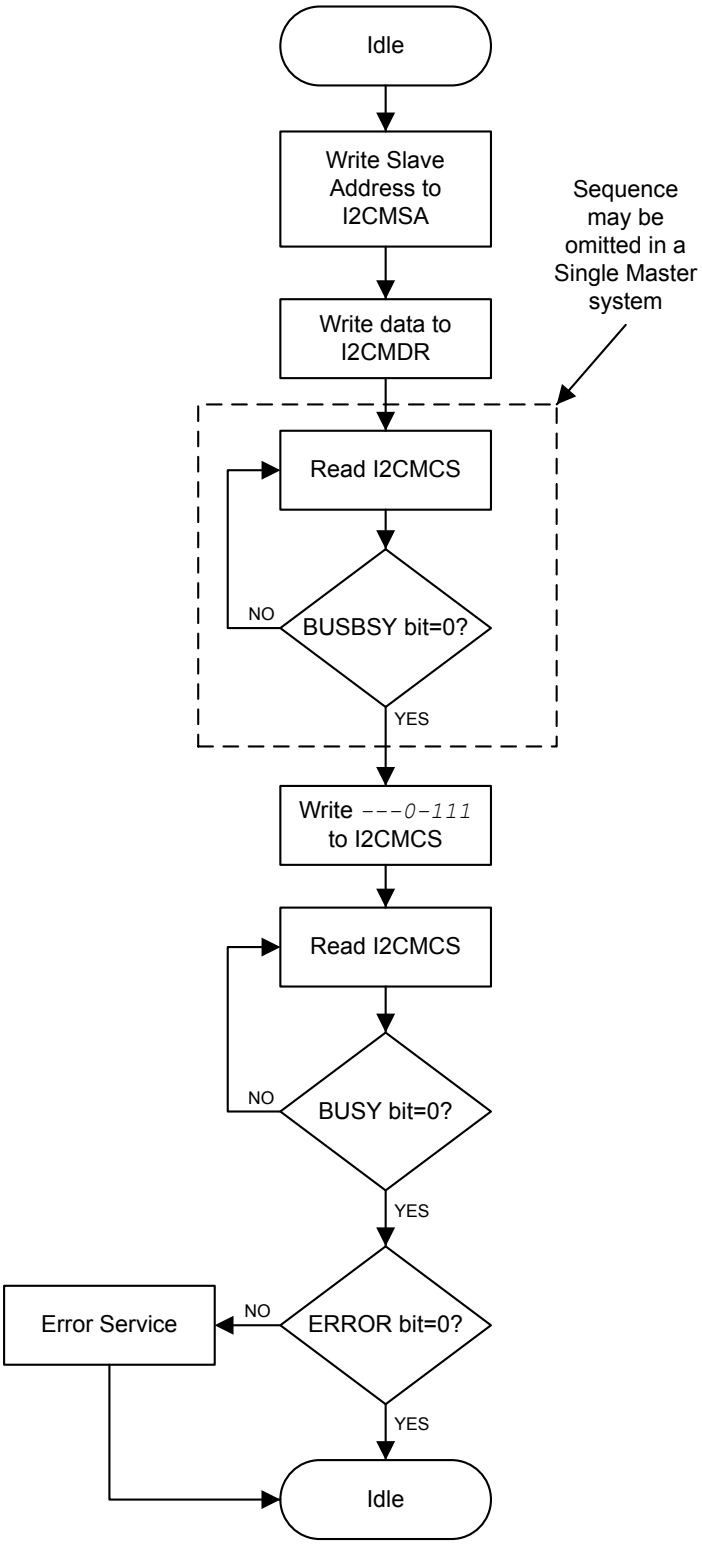


**Figure 15-8. Master Single RECEIVE**

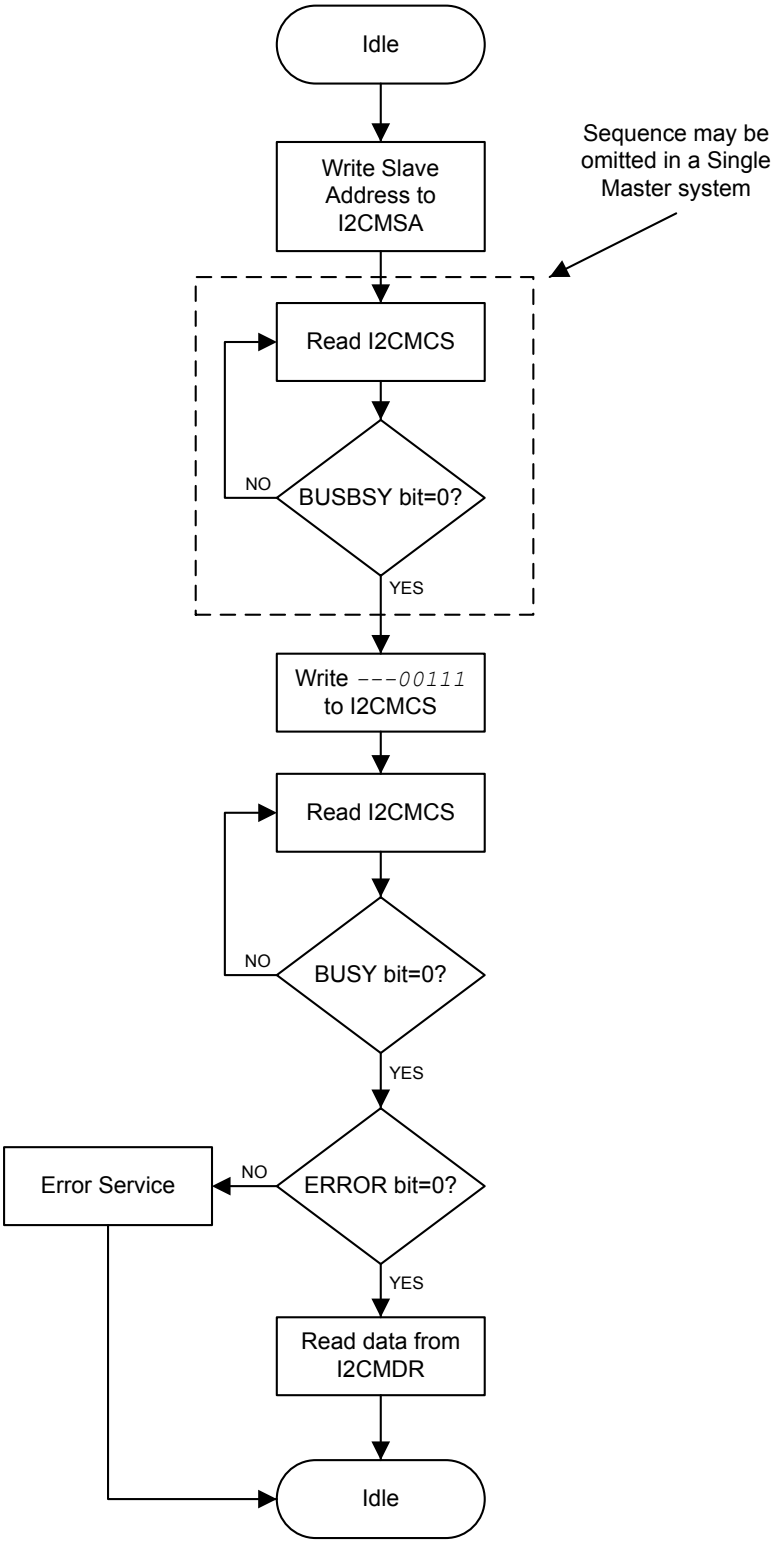


**Figure 15-9. Master TRANSMIT with Repeated START**

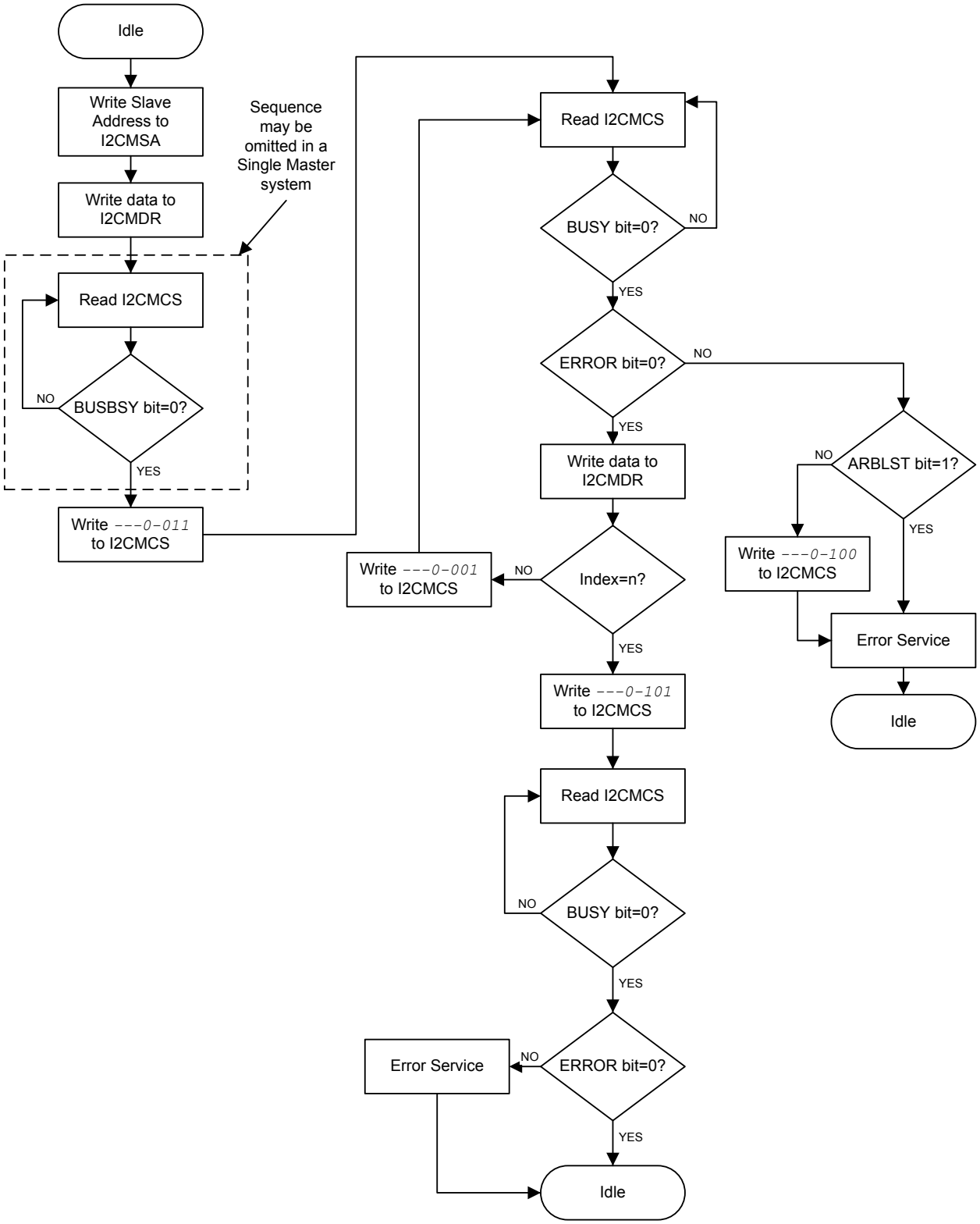


**Figure 15-10. Master RECEIVE with Repeated START**

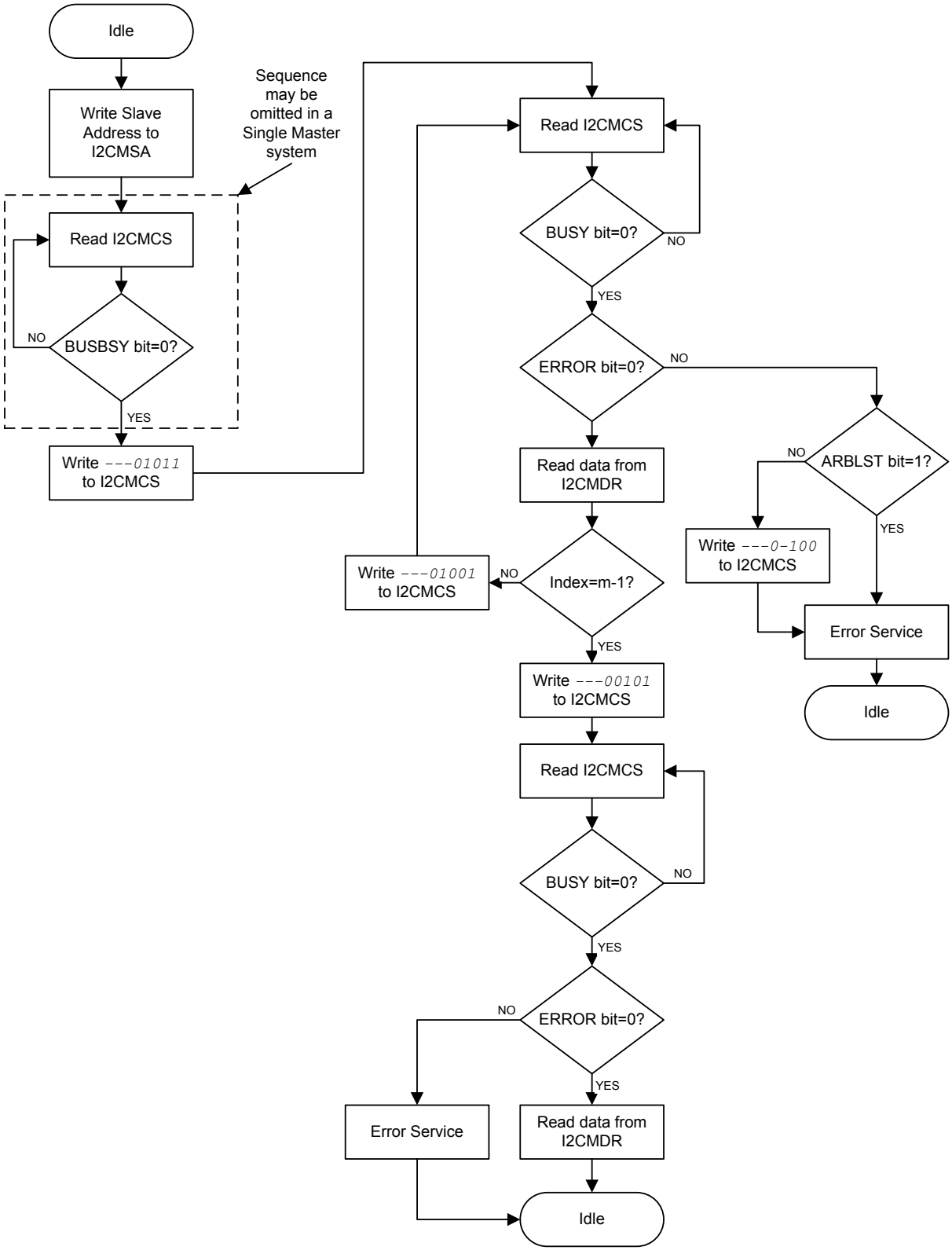


**Figure 15-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START**

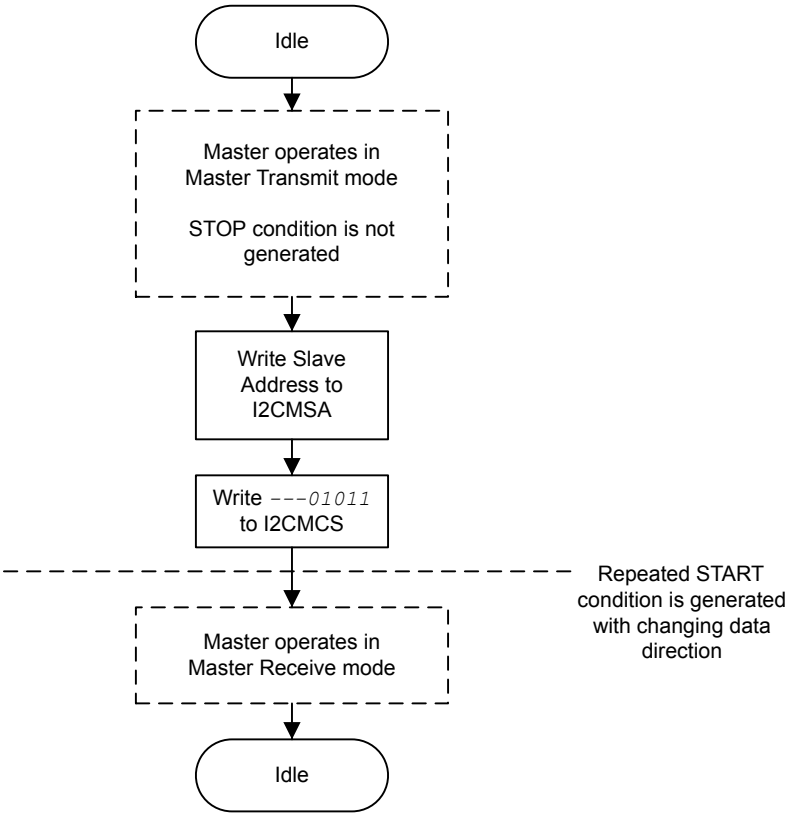
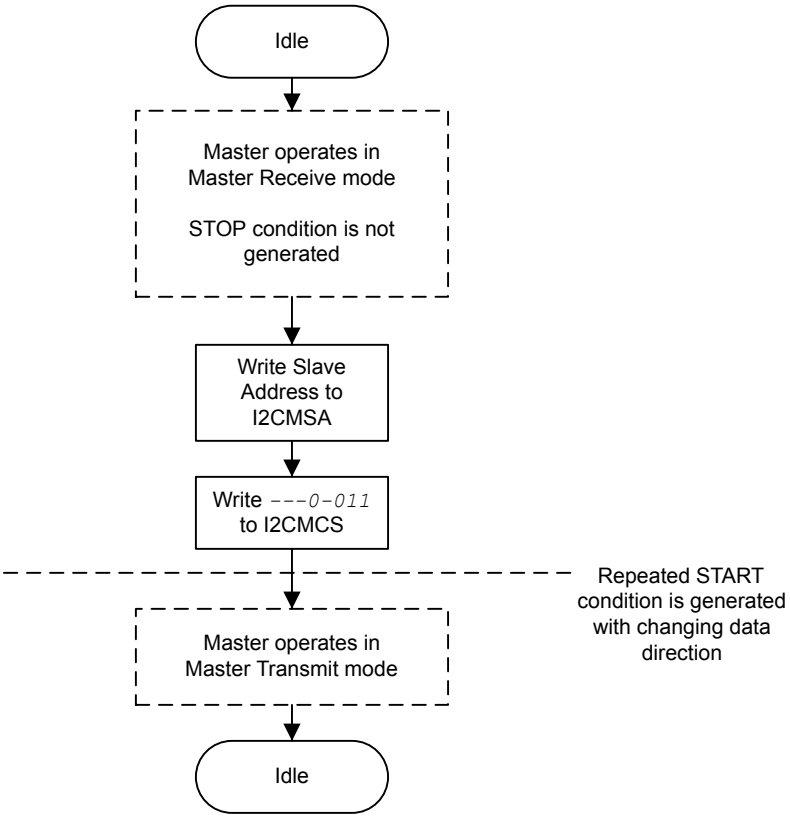


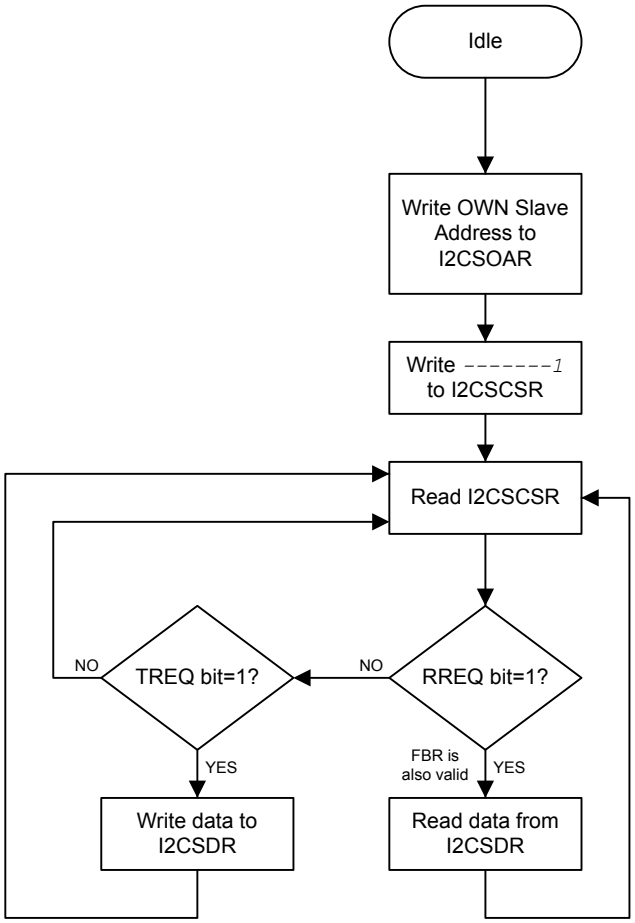
**Figure 15-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START**




### 15.3.5.2 I2C Slave Command Sequences

Figure 15-13 on page 715 presents the command sequence available for the I2C slave.

**Figure 15-13. Slave Command Sequence**




## 15.4 Initialization and Configuration

The following example shows how to configure the I$^2$C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I$^2$C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module (see page 243).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 251). To find out which GPIO port to enable, refer to Table 17-5 on page 757.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 418). To determine which GPIOs to configure, see Table 17-4 on page 753.
4. Enable the I$^2$C pins for Open Drain operation. See page 423.
5. Configure the PMCn fields in the **GPIOPCTL** register to assign the I$^2$C signals to the appropriate pins. See page 435 and Table 17-5 on page 757.
6. Initialize the I$^2$C 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.

## 15.5 Register Map

Table 15-4 on page 716 lists the I2C registers. All addresses given are relative to the I2C base address:

- I2C 0: 0x4002.0000
- I2C 1: 0x4002.1000

Note that the I2C module clock must be enabled before the registers can be programmed (see page 243). There must be a delay of 3 system clocks after the I2C module clock is enabled before any I2C module registers are accessed.

The hw_i2c.h file in the StellarisWare® Driver Library uses a base address of 0x800 for the I2C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that StellarisWare uses an offset between 0x000 and 0x018 with the slave base address.

**Table 15-4. Inter-Integrated Circuit (I2C) Interface Register Map**

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| **I2C Master** | | | | | |
| 0x000 | I2CMSA | R/W | 0x0000.0000 | I2C Master Slave Address | 718 |
| 0x004 | I2CMCS | R/W | 0x0000.0000 | I2C Master Control/Status | 719 |
| 0x008 | I2CMDR | R/W | 0x0000.0000 | I2C Master Data | 723 |
| 0x00C | I2CMTPR | R/W | 0x0000.0001 | I2C Master Timer Period | 724 |
| 0x010 | I2CMIMR | R/W | 0x0000.0000 | I2C Master Interrupt Mask | 725 |
| 0x014 | I2CMRIS | RO | 0x0000.0000 | I2C Master Raw Interrupt Status | 726 |

**Table 15-4. Inter-Integrated Circuit ($I^2C$) Interface Register Map *(continued)***

| Offset | Name | Type | Reset | Description | See page |
|---|---|---|---|---|---|
| 0x018 | I2CMMIS | RO | 0x0000.0000 | I2C Master Masked Interrupt Status | 727 |
| 0x01C | I2CMICR | WO | 0x0000.0000 | I2C Master Interrupt Clear | 728 |
| 0x020 | I2CMCR | R/W | 0x0000.0000 | I2C Master Configuration | 729 |
| **$I^2C$ Slave** | | | | | |
| 0x800 | I2CSOAR | R/W | 0x0000.0000 | I2C Slave Own Address | 730 |
| 0x804 | I2CSCSR | RO | 0x0000.0000 | I2C Slave Control/Status | 731 |
| 0x808 | I2CSDR | R/W | 0x0000.0000 | I2C Slave Data | 733 |
| 0x80C | I2CSIMR | R/W | 0x0000.0000 | I2C Slave Interrupt Mask | 734 |
| 0x810 | I2CSRIS | RO | 0x0000.0000 | I2C Slave Raw Interrupt Status | 735 |
| 0x814 | I2CSMIS | RO | 0x0000.0000 | I2C Slave Masked Interrupt Status | 736 |
| 0x818 | I2CSICR | WO | 0x0000.0000 | I2C Slave Interrupt Clear | 737 |

## 15.6 Register Descriptions ($I^2C$ Master)

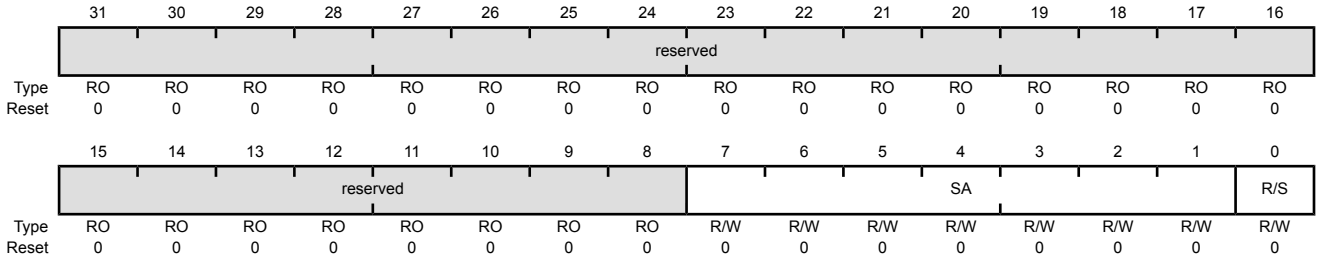
The remainder of this section lists and describes the $I^2C$ master registers, in numerical order by address offset.

## 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
Offset 0x000
Type R/W, 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 | R/W | 0x00 | I²C Slave Address<br>This field specifies bits A6 through A0 of the slave address. |
| 0 | R/S | R/W | 0 | Receive/Send<br>The R/S bit specifies if the next 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

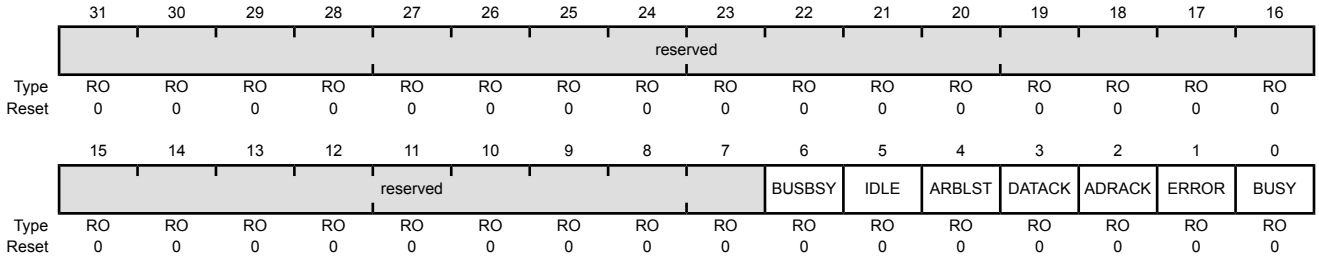
This register accesses status bits when read and control bits when written. When read, the status register indicates the state of the I²C bus controller. When written, the control register configures the I²C controller operation.

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 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 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
Offset 0x004
Type RO, reset 0x0000.0000



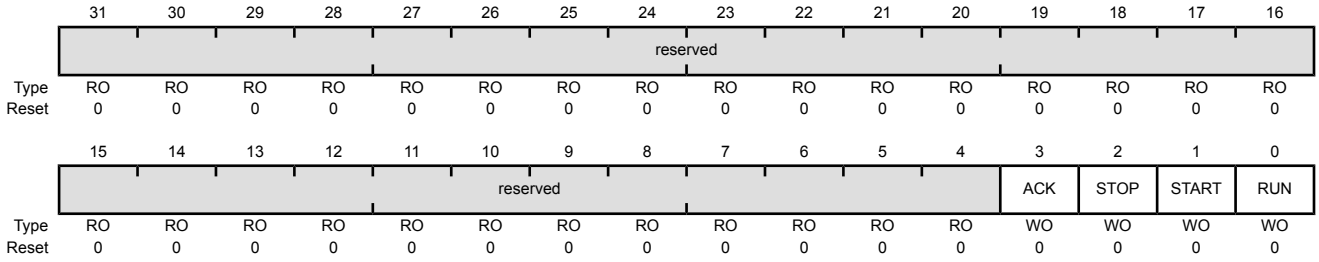

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6 | BUSBSY | RO | 0 | Bus Busy<br><br>Value Description<br>0 The I²C bus is idle.<br>1 The I²C bus is busy.<br><br>The bit changes based on the START and STOP conditions. |
| 5 | IDLE | RO | 0 | I²C Idle<br><br>Value Description<br>0 The I²C controller is not idle.<br>1 The I²C controller is idle. |

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 4 | ARBLST | RO | 0 | Arbitration Lost<br><br>Value Description<br>0 The I²C controller won arbitration.<br>1 The I²C 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. |
| 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 | I²C 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
Offset 0x004
Type WO, 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 | 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 15-5 on page 721. |
| 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 15-5 on page 721. |
| 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 15-5 on page 721. |
| 0 | RUN | WO | 0 | $I^2C$ Master Enable<br>Value Description<br>0 The master is disabled.<br>1 The master is enabled to transmit or receive data. See field decoding in Table 15-5 on page 721. |

**Table 15-5. Write Field Decoding for I2CMCS[3:0] Field**

| Current State | I2CMSA[0] | I2CMCS[3:0] | | | | Description |
|---|---|---|---|---|---|---|
| | R/S | ACK | STOP | START | RUN | |
| Idle | 0 | X[a] | 0 | 1 | 1 | START condition followed by TRANSMIT (master goes to the Master Transmit state). |
| | 0 | X | 1 | 1 | 1 | START condition followed by a TRANSMIT and STOP condition (master remains in Idle state). |
| | 1 | 0 | 0 | 1 | 1 | START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state). |
| | 1 | 0 | 1 | 1 | 1 | START condition followed by RECEIVE and STOP condition (master remains in Idle state). |
| | 1 | 1 | 0 | 1 | 1 | START condition followed by RECEIVE (master goes to the Master Receive state). |
| | 1 | 1 | 1 | 1 | 1 | Illegal |
| | All other combinations not listed are non-operations. | | | | | NOP |

**Table 15-5. Write Field Decoding for I2CMCS[3:0] Field *(continued)***

| Current State | I2CMSA[0] | I2CMCS[3:0] | | | | Description |
|---|---|---|---|---|---|---|
| | R/S | ACK | STOP | START | RUN | |
| Master Transmit | X | X | 0 | 0 | 1 | TRANSMIT operation (master remains in Master Transmit state). |
| | X | X | 1 | 0 | 0 | STOP condition (master goes to Idle state). |
| | X | X | 1 | 0 | 1 | TRANSMIT followed by STOP condition (master goes to Idle state). |
| | 0 | X | 0 | 1 | 1 | Repeated START condition followed by a TRANSMIT (master remains in Master Transmit state). |
| | 0 | X | 1 | 1 | 1 | Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state). |
| | 1 | 0 | 0 | 1 | 1 | Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state). |
| | 1 | 0 | 1 | 1 | 1 | Repeated START condition followed by a TRANSMIT and STOP condition (master goes to Idle state). |
| | 1 | 1 | 0 | 1 | 1 | Repeated START condition followed by RECEIVE (master goes to Master Receive state). |
| | 1 | 1 | 1 | 1 | 1 | Illegal. |
| | All other combinations not listed are non-operations. | | | | | NOP. |
| Master Receive | X | 0 | 0 | 0 | 1 | RECEIVE operation with negative ACK (master remains in Master Receive state). |
| | X | X | 1 | 0 | 0 | STOP condition (master goes to Idle state).[b] |
| | X | 0 | 1 | 0 | 1 | RECEIVE followed by STOP condition (master goes to Idle state). |
| | X | 1 | 0 | 0 | 1 | RECEIVE operation (master remains in Master Receive state). |
| | X | 1 | 1 | 0 | 1 | Illegal. |
| | 1 | 0 | 0 | 1 | 1 | Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state). |
| | 1 | 0 | 1 | 1 | 1 | Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state). |
| | 1 | 1 | 0 | 1 | 1 | Repeated START condition followed by RECEIVE (master remains in Master Receive state). |
| | 0 | X | 0 | 1 | 1 | Repeated START condition followed by TRANSMIT (master goes to Master Transmit state). |
| | 0 | X | 1 | 1 | 1 | Repeated START condition followed by TRANSMIT 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.

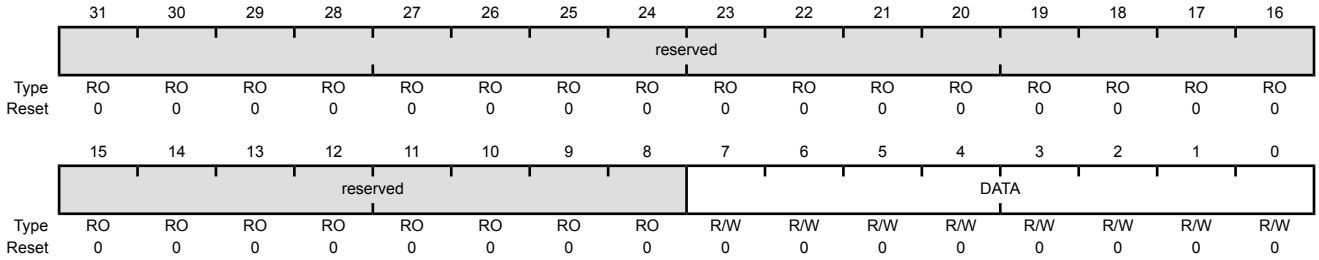
## Register 3: I²C Master Data (I2CMDR), offset 0x008

**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.

I2C Master Data (I2CMDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x008
Type R/W, 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 | R/W | 0x00 | Data Transferred<br>Data transferred during transaction. |

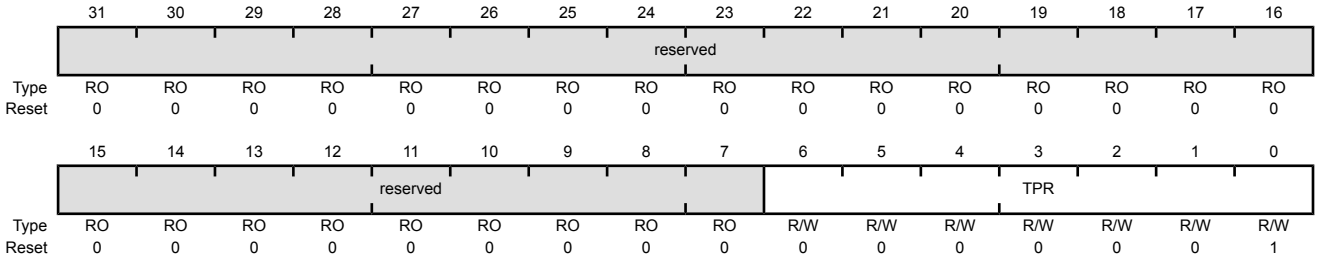
## Register 4: I2C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

**Caution – Take care not to set bit 7 when accessing this register as unpredictable behavior can occur.**

I2C Master Timer Period (I2CMTPR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x00C
Type R/W, reset 0x0000.0001



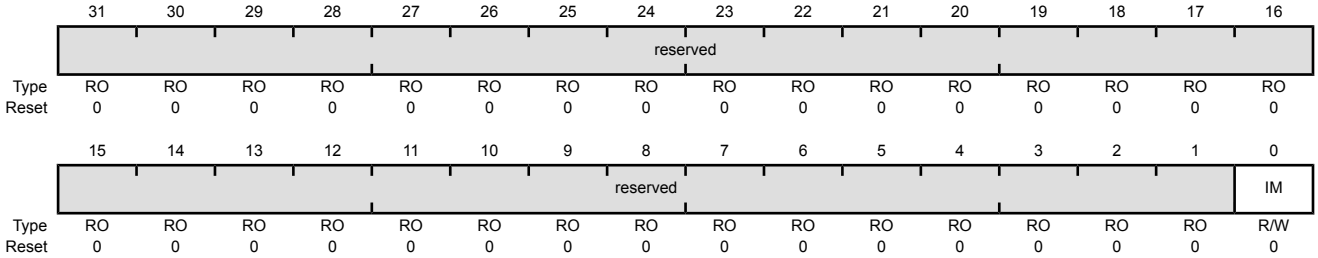

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:7 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | TPR | R/W | 0x1 | SCL Clock Period<br>This field specifies the period of the SCL clock.<br>$SCL_PRD = 2\times(1 + TPR)\times(SCL_LP + SCL_HP)\times CLK_PRD$<br>where:<br>*SCL_PRD* is the SCL line period (I2C 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
Offset 0x010
Type R/W, 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 | IM | R/W | 0 | Interrupt Mask |

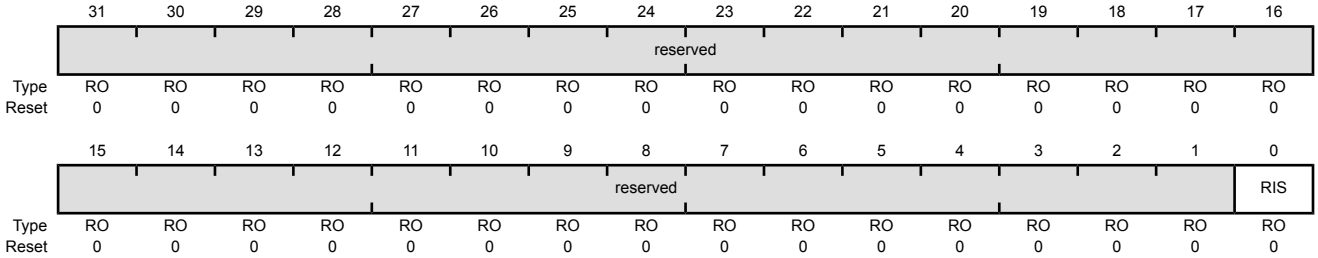
| Value | Description |
|---|---|
| 1 | The master interrupt is sent to the interrupt controller when the RIS bit in the **I2CMRIS** register is set. |
| 0 | The RIS interrupt is suppressed and not sent to the interrupt controller. |

## 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
Offset 0x014
Type RO, reset 0x0000.0000



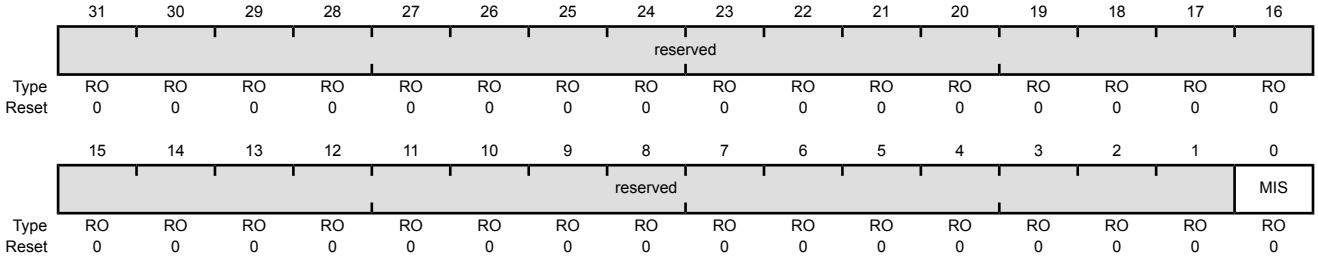

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | RIS | RO | 0 | Raw Interrupt Status<br><br>Value Description<br>1 A master interrupt is pending.<br>0 No interrupt.<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
Offset 0x018
Type RO, reset 0x0000.0000



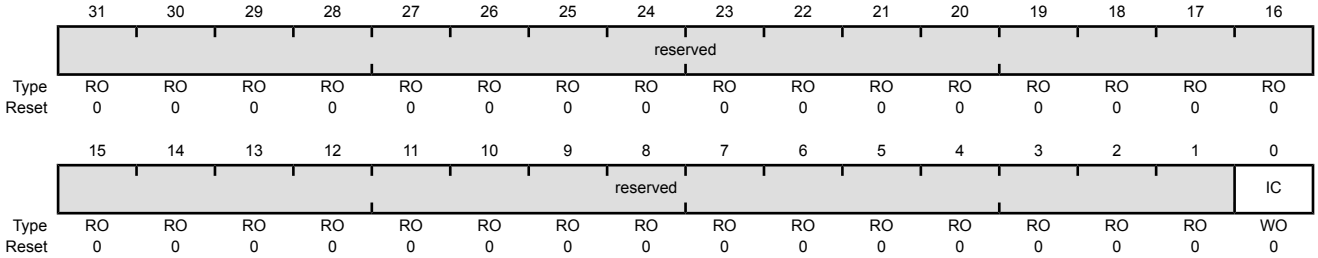

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | MIS | RO | 0 | Masked Interrupt Status<br><br>Value Description<br>1 An unmasked master interrupt was signaled and is pending.<br>0 An interrupt has not occurred or is masked.<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
Offset 0x01C
Type WO, reset 0x0000.0000

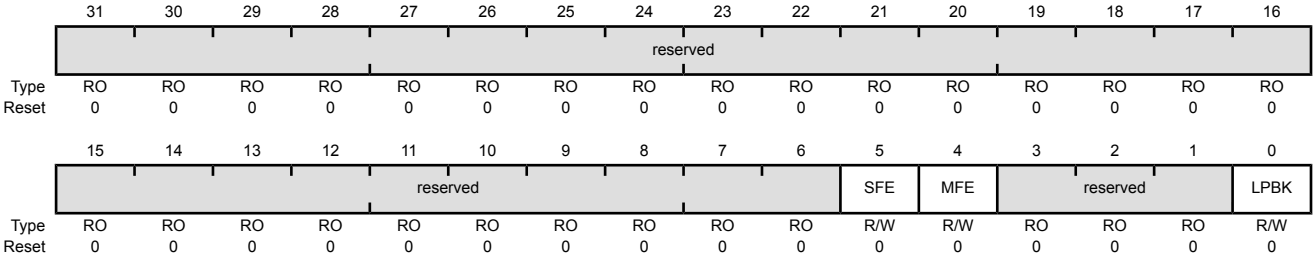



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | IC | WO | 0 | 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
Offset 0x020
Type R/W, 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 | R/W | 0 | I²C Slave Function Enable<br>Value Description<br>1 Slave mode is enabled.<br>0 Slave mode is disabled. |
| 4 | MFE | R/W | 0 | I²C Master Function Enable<br>Value Description<br>1 Master mode is enabled.<br>0 Master mode is disabled. |
| 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 | R/W | 0 | I²C Loopback<br>Value Description<br>1 The controller in a test mode loopback configuration.<br>0 Normal operation. |

## 15.7 Register Descriptions (I²C Slave)

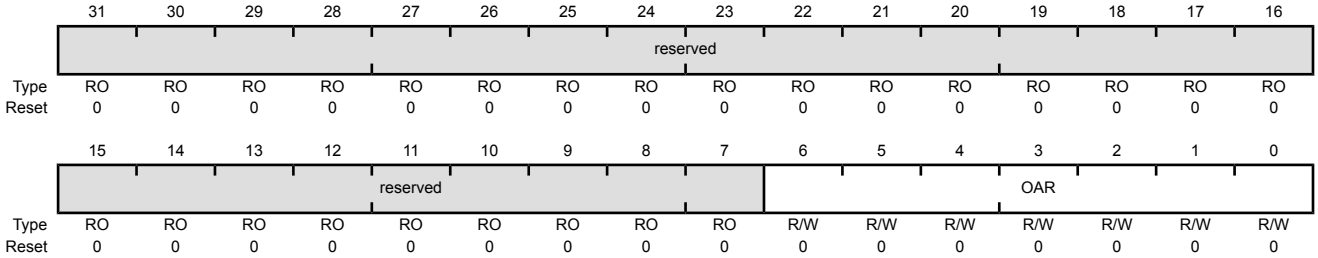
The remainder of this section lists and describes the I²C slave registers, in numerical order by address offset.

## Register 10: I2C Slave Own Address (I2CSOAR), offset 0x800

This register consists of seven address bits that identify the Stellaris I2C device on the I2C bus.

I2C Slave Own Address (I2CSOAR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x800
Type R/W, 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 | R/W | 0x00 | I2C Slave Own Address<br>This field specifies bits A6 through A0 of the slave address. |

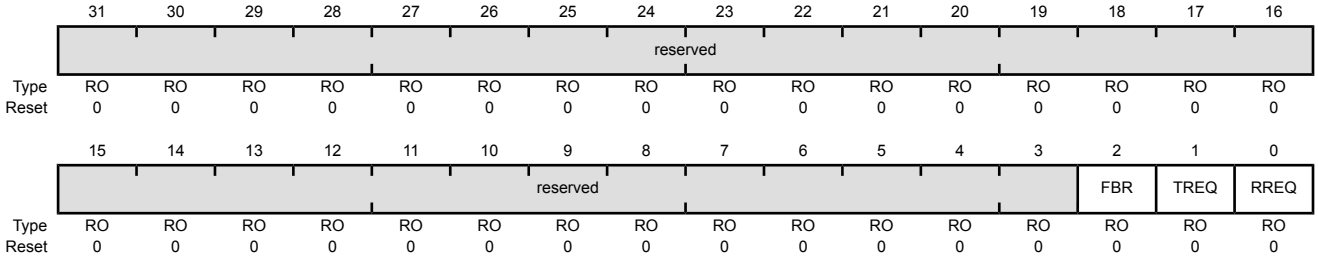
## Register 11: I²C Slave Control/Status (I2CSCSR), offset 0x804

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
Offset 0x804
Type RO, reset 0x0000.0000

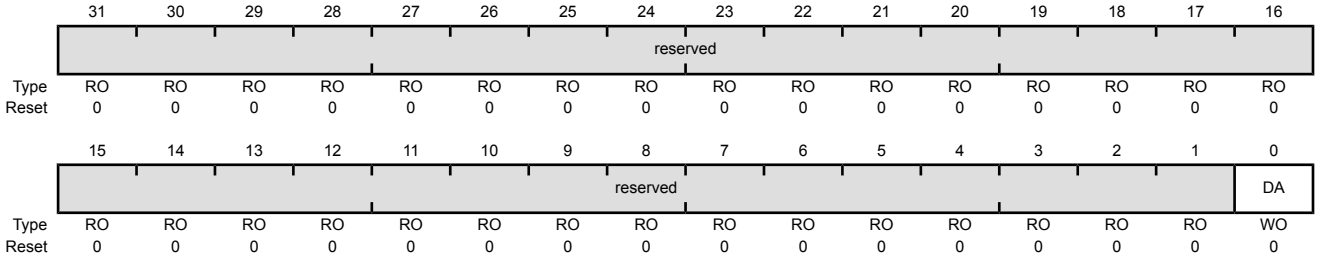



| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | FBR | RO | 0 | First Byte Received<br><br>Value Description<br>1 The first byte following the slave's own address has been received.<br>0 The first byte has not 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>1 The I²C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the **I2CSDR** register.<br>0 No outstanding transmit request. |
| 0 | RREQ | RO | 0 | Receive Request<br><br>Value Description<br>1 The I²C controller has outstanding receive data from the I²C master and is using clock stretching to delay the master until the data has been read from the **I2CSDR** register.<br>0 No outstanding receive data. |

## Write-Only Control Register

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x804
Type WO, 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 | DA | WO | 0 | Device Active<br>Value Description<br>0 Disables the I²C slave operation.<br>1 Enables the I²C slave operation. |

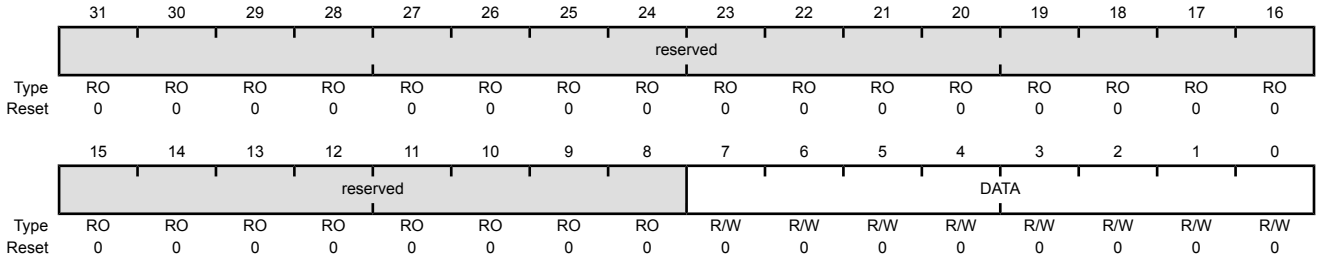
## Register 12: I²C Slave Data (I2CSDR), offset 0x808

**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.

I2C Slave Data (I2CSDR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x808
Type R/W, reset 0x0000.0000



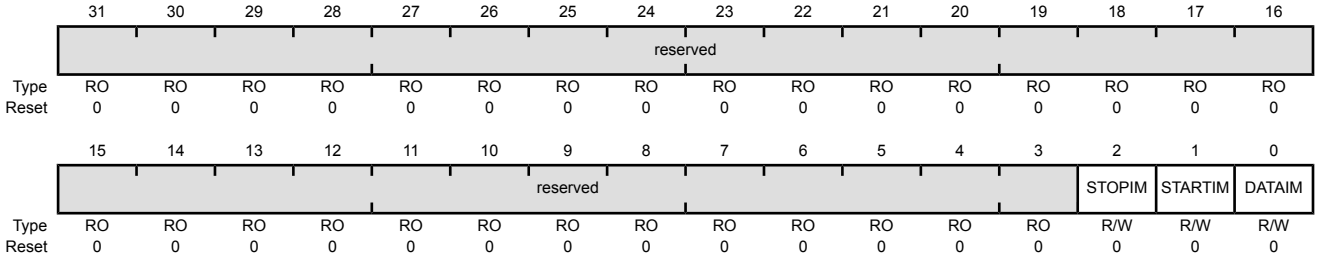

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | R/W | 0x00 | Data for Transfer<br>This field contains the data for transfer during a slave receive or transmit operation. |

## Register 13: 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
Offset 0x80C
Type R/W, reset 0x0000.0000



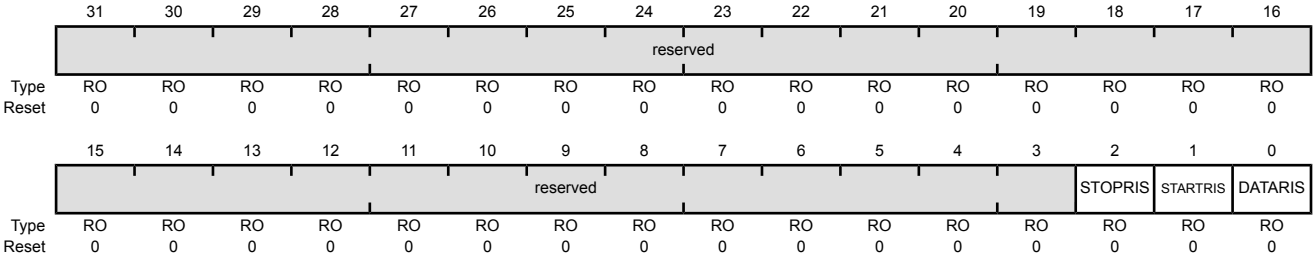

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | STOPIM | R/W | 0 | Stop Condition Interrupt Mask<br>Value Description<br>1 The STOP condition interrupt is sent to the interrupt controller when the STOPRIS bit in the **I2CSRIS** register is set.<br>0 The STOPRIS interrupt is suppressed and not sent to the interrupt controller. |
| 1 | STARTIM | R/W | 0 | Start Condition Interrupt Mask<br>Value Description<br>1 The START condition interrupt is sent to the interrupt controller when the STARTRIS bit in the **I2CSRIS** register is set.<br>0 The STARTRIS interrupt is suppressed and not sent to the interrupt controller. |
| 0 | DATAIM | R/W | 0 | Data Interrupt Mask<br>Value Description<br>1 The data received or data requested interrupt is sent to the interrupt controller when the DATARIS bit in the **I2CSRIS** register is set.<br>0 The DATARIS interrupt is suppressed and not sent to the interrupt controller. |

## Register 14: 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
Offset 0x810
Type RO, reset 0x0000.0000



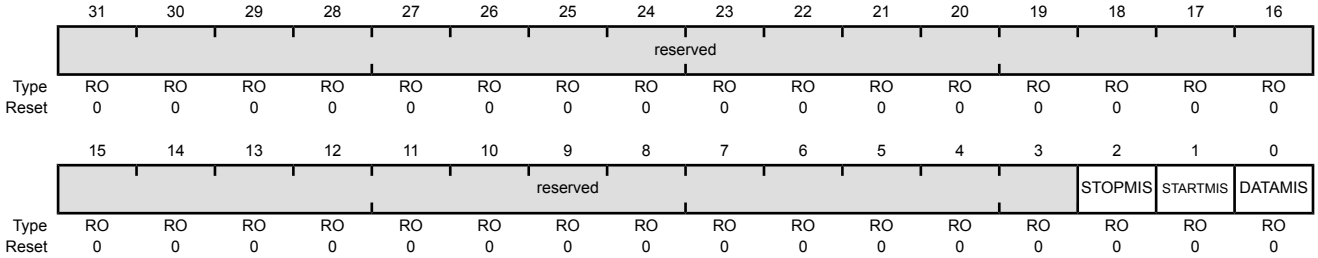

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | STOPRIS | RO | 0 | Stop Condition Raw Interrupt Status<br><br>Value Description<br>1 A STOP condition interrupt is pending.<br>0 No interrupt.<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>1 A START condition interrupt is pending.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1 to the `STARTIC` bit in the **I2CSICR** register. |
| 0 | DATARIS | RO | 0 | Data Raw Interrupt Status<br><br>Value Description<br>1 A data received or data requested interrupt is pending.<br>0 No interrupt.<br><br>This bit is cleared by writing a 1 to the `DATAIC` bit in the **I2CSICR** register. |

## Register 15: 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
Offset 0x814
Type RO, reset 0x0000.0000



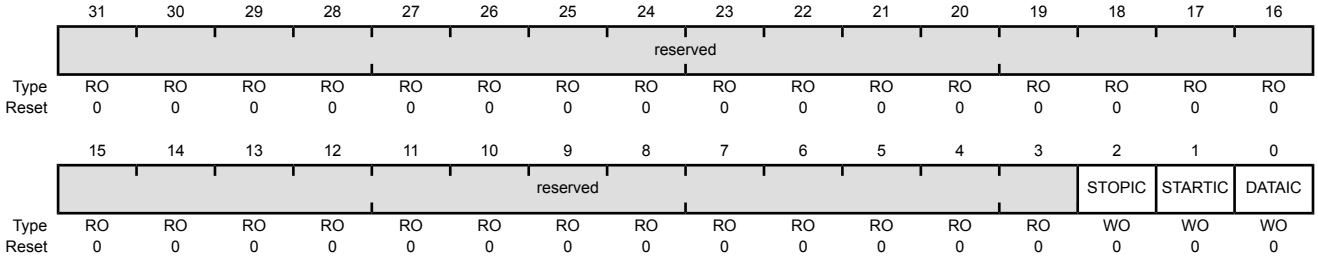

| Bit/Field | Name | Type | Reset | Description |
|---|---|---|---|---|
| 31:3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | STOPMIS | RO | 0 | Stop Condition Masked Interrupt Status<br>Value Description<br>1 An unmasked STOP condition interrupt was signaled is pending.<br>0 An interrupt has not occurred or is masked.<br>This bit is cleared by writing a 1 to the STOPIC bit in the **I2CSICR** register. |
| 1 | STARTMIS | RO | 0 | Start Condition Masked Interrupt Status<br>Value Description<br>1 An unmasked START condition interrupt was signaled is pending.<br>0 An interrupt has not occurred or is masked.<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>Value Description<br>1 An unmasked data received or data requested interrupt was signaled is pending.<br>0 An interrupt has not occurred or is masked.<br>This bit is cleared by writing a 1 to the DATAIC bit in the **I2CSICR** register. |

## Register 16: 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
Offset 0x818
Type WO, 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 | 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 STOPRIS bit in the **I2CSRIS** register and the STOPMIS 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 STOPRIS bit in the **I2CSRIS** register and the STOPMIS bit in the **I2CSMIS** register.<br>A read of this register returns no meaningful data. |

# 16 Pin Diagram

The LM3S1G58 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 17-5 on page 757.

**Figure 16-1. 100-Pin LQFP Package Pin Diagram**

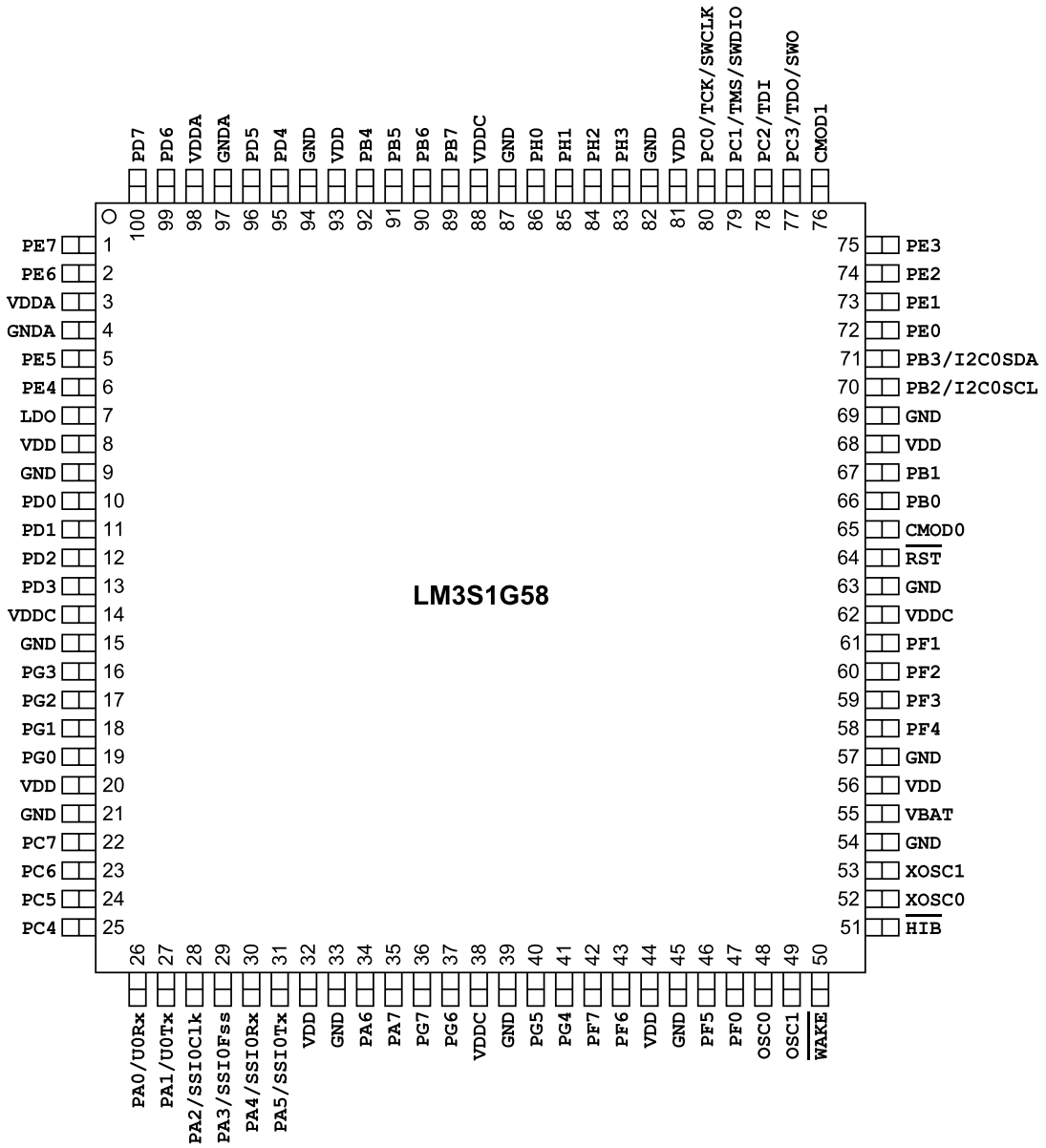
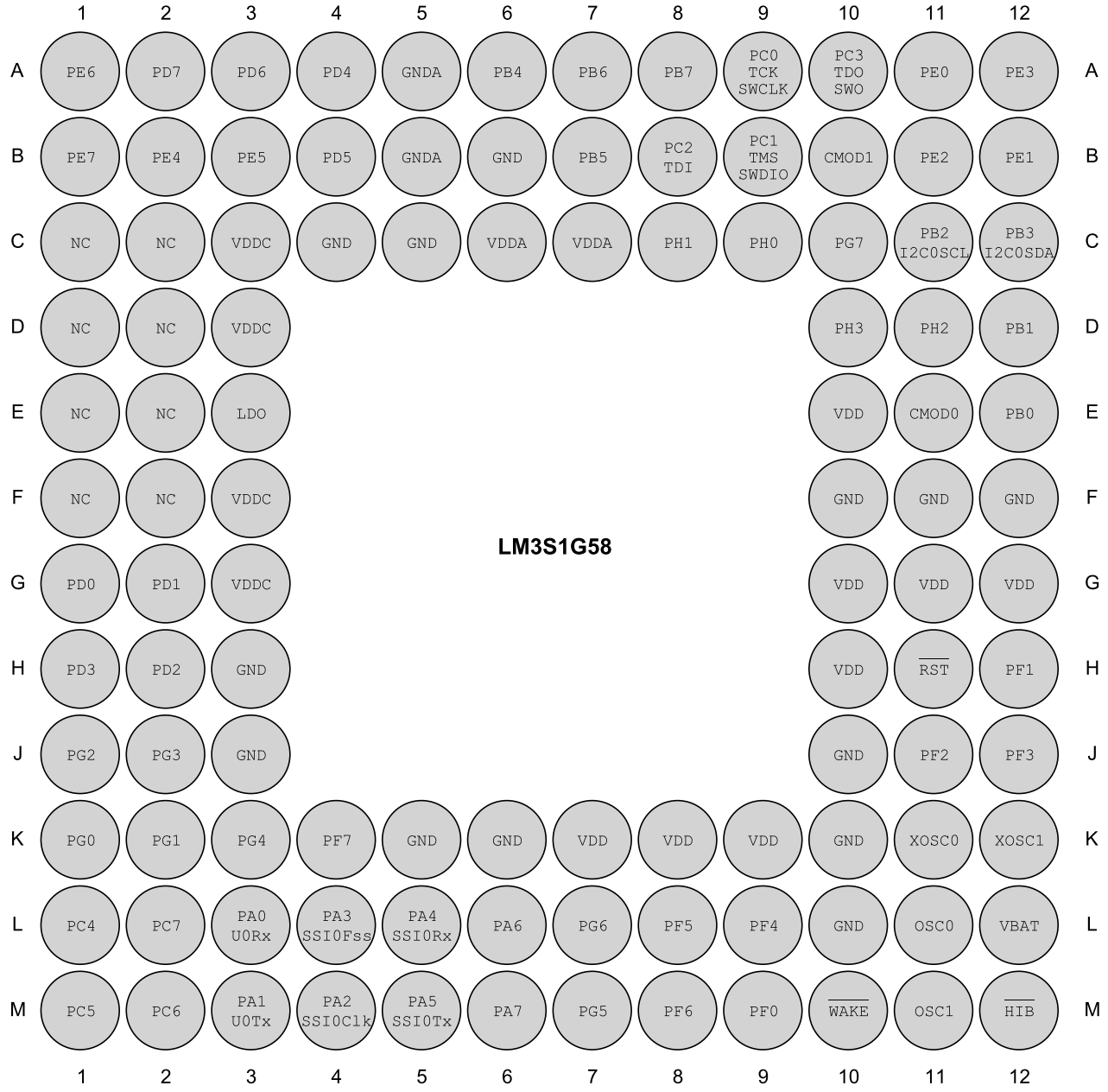


**Figure 16-2. 108-Ball BGA Package Pin Diagram (Top View)**



# 17 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 434) 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 418) must be set. Further pin muxing options are provided through the PMCx bit field in the **GPIOPCTL** register (see page 435), which selects one of several available peripheral functions for that GPIO.

**Important:** All GPIO pins are configured as GPIOs by default with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

**Table 17-1. GPIO Pins With Default Alternate Functions**

| GPIO Pin | Default State | GPIOAFSEL Bit | GPIOPCTL PMCx Bit Field |
|---|---|---|---|
| PA[1:0] | UART0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 0 | 0x1 |
| PB[3:2] | $I^2C0$ | 0 | 0x1 |
| PC[3:0] | JTAG/SWD | 1 | 0x3 |

Table 17-2 on page 741 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 17-3 on page 747 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 17-4 on page 753 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 17-5 on page 757 lists the GPIO pins and their analog and digital alternate functions. The AINx and VREFA analog signals are not 5-V tolerant and 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. 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 17-6 on page 760 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 Stellaris® 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.

## 17.1 100-Pin LQFP Package Pin Tables

**Table 17-2. Signals by Pin Number**

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| 1 | PE7 | I/O | TTL | GPIO port E bit 7. |
| | AIN0 | I | Analog | Analog-to-digital converter input 0. |
| 2 | PE6 | I/O | TTL | GPIO port E bit 6. |
| | AIN1 | I | Analog | Analog-to-digital converter input 1. |
| 3 | VDDA | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |
| 4 | GNDA | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| 5 | PE5 | I/O | TTL | GPIO port E bit 5. |
| | AIN2 | I | Analog | Analog-to-digital converter input 2. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| 6 | PE4 | I/O | TTL | GPIO port E bit 4. |
| | AIN3 | I | Analog | Analog-to-digital converter input 3. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 7 | LDO | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| 8 | VDD | - | Power | Positive supply for I/O and some logic. |
| 9 | GND | - | Power | Ground reference for logic and I/O pins. |
| 10 | PD0 | I/O | TTL | GPIO port D bit 0. |
| | AIN15 | I | Analog | Analog-to-digital converter input 15. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| 11 | PD1 | I/O | TTL | GPIO port D bit 1. |
| | AIN14 | I | Analog | Analog-to-digital converter input 14. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |

**Table 17-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| 12 | PD2 | I/O | TTL | GPIO port D bit 2. |
| | AIN13 | I | Analog | Analog-to-digital converter input 13. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| 13 | PD3 | I/O | TTL | GPIO port D bit 3. |
| | AIN12 | I | Analog | Analog-to-digital converter input 12. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 14 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| 15 | GND | - | Power | Ground reference for logic and I/O pins. |
| 16 | PG3 | I/O | TTL | GPIO port G bit 3. |
| 17 | PG2 | I/O | TTL | GPIO port G bit 2. |
| 18 | PG1 | I/O | TTL | GPIO port G bit 1. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 19 | PG0 | I/O | TTL | GPIO port G bit 0. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| 20 | VDD | - | Power | Positive supply for I/O and some logic. |
| 21 | GND | - | Power | Ground reference for logic and I/O pins. |
| 22 | PC7 | I/O | TTL | GPIO port C bit 7. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 23 | PC6 | I/O | TTL | GPIO port C bit 6. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| 24 | PC5 | I/O | TTL | GPIO port C bit 5. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |

**Table 17-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| 25 | PC4 | I/O | TTL | GPIO port C bit 4. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| 26 | PA0 | I/O | TTL | GPIO port A bit 0. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. |
| | U0Rx | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| 27 | PA1 | I/O | TTL | GPIO port A bit 1. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| | U0Tx | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 28 | PA2 | I/O | TTL | GPIO port A bit 2. |
| | SSI0Clk | I/O | TTL | SSI module 0 clock. |
| 29 | PA3 | I/O | TTL | GPIO port A bit 3. |
| | SSI0Fss | I/O | TTL | SSI module 0 frame. |
| 30 | PA4 | I/O | TTL | GPIO port A bit 4. |
| | SSI0Rx | I | TTL | SSI module 0 receive. |
| 31 | PA5 | I/O | TTL | GPIO port A bit 5. |
| | SSI0Tx | O | TTL | SSI module 0 transmit. |
| 32 | VDD | - | Power | Positive supply for I/O and some logic. |
| 33 | GND | - | Power | Ground reference for logic and I/O pins. |
| 34 | PA6 | I/O | TTL | GPIO port A bit 6. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. |
| 35 | PA7 | I/O | TTL | GPIO port A bit 7. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| 36 | PG7 | I/O | TTL | GPIO port G bit 7. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| 37 | PG6 | I/O | TTL | GPIO port G bit 6. |
| 38 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| 39 | GND | - | Power | Ground reference for logic and I/O pins. |
| 40 | PG5 | I/O | TTL | GPIO port G bit 5. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |

**Table 17-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| 41 | PG4 | I/O | TTL | GPIO port G bit 4. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| 42 | PF7 | I/O | TTL | GPIO port F bit 7. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| 43 | PF6 | I/O | TTL | GPIO port F bit 6. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| 44 | VDD | - | Power | Positive supply for I/O and some logic. |
| 45 | GND | - | Power | Ground reference for logic and I/O pins. |
| 46 | PF5 | I/O | TTL | GPIO port F bit 5. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | SSI1Tx | O | TTL | SSI module 1 transmit. |
| 47 | PF0 | I/O | TTL | GPIO port F bit 0. |
| | U1DSR | I | TTL | UART module 1 Data Set Ready modem output control line. |
| 48 | OSC0 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| 49 | OSC1 | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| 50 | $\overline{\text{WAKE}}$ | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| 51 | $\overline{\text{HIB}}$ | O | OD | An output that indicates the processor is in Hibernate mode. |
| 52 | XOSC0 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| 53 | XOSC1 | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| 54 | GND | - | Power | Ground reference for logic and I/O pins. |
| 55 | 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. |
| 56 | VDD | - | Power | Positive supply for I/O and some logic. |
| 57 | GND | - | Power | Ground reference for logic and I/O pins. |
| 58 | PF4 | I/O | TTL | GPIO port F bit 4. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | SSI1Rx | I | TTL | SSI module 1 receive. |
| 59 | PF3 | I/O | TTL | GPIO port F bit 3. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame. |
| 60 | PF2 | I/O | TTL | GPIO port F bit 2. |
| | SSI1Clk | I/O | TTL | SSI module 1 clock. |
| 61 | PF1 | I/O | TTL | GPIO port F bit 1. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| 62 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| 63 | GND | - | Power | Ground reference for logic and I/O pins. |
| 64 | $\overline{\text{RST}}$ | I | TTL | System reset input. |

**Table 17-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| 65 | CMOD0 | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| 66 | PB0 | I/O | TTL | GPIO port B bit 0. This pin is not 5-V tolerant. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| 67 | PB1 | I/O | TTL | GPIO port B bit 1. This pin is not 5-V tolerant. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 68 | VDD | - | Power | Positive supply for I/O and some logic. |
| 69 | GND | - | Power | Ground reference for logic and I/O pins. |
| 70 | PB2 | I/O | TTL | GPIO port B bit 2. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | I2C0SCL | I/O | OD | $I^2C$ module 0 clock. |
| 71 | PB3 | I/O | TTL | GPIO port B bit 3. |
| | I2C0SDA | I/O | OD | $I^2C$ module 0 data. |
| 72 | PE0 | I/O | TTL | GPIO port E bit 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | SSI1Clk | I/O | TTL | SSI module 1 clock. |
| 73 | PE1 | I/O | TTL | GPIO port E bit 1. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame. |
| 74 | PE2 | I/O | TTL | GPIO port E bit 2. |
| | AIN9 | I | Analog | Analog-to-digital converter input 9. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | SSI1Rx | I | TTL | SSI module 1 receive. |
| 75 | PE3 | I/O | TTL | GPIO port E bit 3. |
| | AIN8 | I | Analog | Analog-to-digital converter input 8. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | SSI1Tx | O | TTL | SSI module 1 transmit. |
| 76 | CMOD1 | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |
| 77 | PC3 | I/O | TTL | GPIO port C bit 3. |
| | SWO | O | TTL | JTAG TDO and SWO. |
| | TDO | O | TTL | JTAG TDO and SWO. |

**Table 17-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| 78 | PC2 | I/O | TTL | GPIO port C bit 2. |
| | TDI | I | TTL | JTAG TDI. |
| 79 | PC1 | I/O | TTL | GPIO port C bit 1. |
| | SWDIO | I/O | TTL | JTAG TMS and SWDIO. |
| | TMS | I | TTL | JTAG TMS and SWDIO. |
| 80 | PC0 | I/O | TTL | GPIO port C bit 0. |
| | SWCLK | I | TTL | JTAG/SWD CLK. |
| | TCK | I | TTL | JTAG/SWD CLK. |
| 81 | VDD | - | Power | Positive supply for I/O and some logic. |
| 82 | GND | - | Power | Ground reference for logic and I/O pins. |
| 83 | PH3 | I/O | TTL | GPIO port H bit 3. |
| 84 | PH2 | I/O | TTL | GPIO port H bit 2. |
| 85 | PH1 | I/O | TTL | GPIO port H bit 1. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| 86 | PH0 | I/O | TTL | GPIO port H bit 0. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| 87 | GND | - | Power | Ground reference for logic and I/O pins. |
| 88 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| 89 | PB7 | I/O | TTL | GPIO port B bit 7. |
| | NMI | I | TTL | Non-maskable interrupt. |
| 90 | PB6 | I/O | TTL | GPIO port B bit 6. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | VREFA | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |
| 91 | PB5 | I/O | TTL | GPIO port B bit 5. |
| | AIN11 | I | Analog | Analog-to-digital converter input 11. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 92 | PB4 | I/O | TTL | GPIO port B bit 4. |
| | AIN10 | I | Analog | Analog-to-digital converter input 10. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |

**Table 17-2. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| 93 | VDD | - | Power | Positive supply for I/O and some logic. |
| 94 | GND | - | Power | Ground reference for logic and I/O pins. |
| 95 | PD4 | I/O | TTL | GPIO port D bit 4. |
| | AIN7 | I | Analog | Analog-to-digital converter input 7. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| 96 | PD5 | I/O | TTL | GPIO port D bit 5. |
| | AIN6 | I | Analog | Analog-to-digital converter input 6. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| 97 | GNDA | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| 98 | VDDA | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |
| 99 | PD6 | I/O | TTL | GPIO port D bit 6. |
| | AIN5 | I | Analog | Analog-to-digital converter input 5. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 100 | PD7 | I/O | TTL | GPIO port D bit 7. |
| | AIN4 | I | Analog | Analog-to-digital converter input 4. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 17-3. Signals by Signal Name**

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| AIN0 | 1 | PE7 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | 2 | PE6 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | 5 | PE5 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | 6 | PE4 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | 100 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | 99 | PD6 | I | Analog | Analog-to-digital converter input 5. |
| AIN6 | 96 | PD5 | I | Analog | Analog-to-digital converter input 6. |
| AIN7 | 95 | PD4 | I | Analog | Analog-to-digital converter input 7. |
| AIN8 | 75 | PE3 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | 74 | PE2 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | 92 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | 91 | PB5 | I | Analog | Analog-to-digital converter input 11. |

**Table 17-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| AIN12 | 13 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | 12 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | 11 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | 10 | PD0 | I | Analog | Analog-to-digital converter input 15. |
| CCP0 | 13<br>22<br>23<br>58<br>66<br>70<br>91<br>95 | PD3 (4)<br>PC7 (4)<br>PC6 (6)<br>PF4 (1)<br>PB0 (1)<br>PB2 (5)<br>PB5 (4)<br>PD4 (1) | I/O | TTL | Capture/Compare/PWM 0. |
| CCP1 | 24<br>25<br>34<br>43<br>67<br>75<br>90<br>100 | PC5 (1)<br>PC4 (9)<br>PA6 (2)<br>PF6 (1)<br>PB1 (4)<br>PE3 (1)<br>PB6 (1)<br>PD7 (3) | I/O | TTL | Capture/Compare/PWM 1. |
| CCP2 | 6<br>11<br>25<br>46<br>67<br>73<br>74<br>91<br>96 | PE4 (6)<br>PD1 (10)<br>PC4 (5)<br>PF5 (1)<br>PB1 (1)<br>PE1 (4)<br>PE2 (5)<br>PB5 (6)<br>PD5 (1) | I/O | TTL | Capture/Compare/PWM 2. |
| CCP3 | 6<br>23<br>24<br>35<br>41<br>61<br>70<br>72<br>95 | PE4 (1)<br>PC6 (1)<br>PC5 (5)<br>PA7 (7)<br>PG4 (1)<br>PF1 (10)<br>PB2 (4)<br>PE0 (3)<br>PD4 (2) | I/O | TTL | Capture/Compare/PWM 3. |
| CCP4 | 22<br>25<br>35<br>42<br>74<br>96 | PC7 (1)<br>PC4 (6)<br>PA7 (2)<br>PF7 (1)<br>PE2 (1)<br>PD5 (2) | I/O | TTL | Capture/Compare/PWM 4. |
| CCP5 | 5<br>12<br>25<br>36<br>40<br>90<br>91 | PE5 (1)<br>PD2 (4)<br>PC4 (1)<br>PG7 (8)<br>PG5 (1)<br>PB6 (6)<br>PB5 (2) | I/O | TTL | Capture/Compare/PWM 5. |

**Table 17-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CCP6 | 10<br>12<br>73<br>86<br>91 | PD0 (6)<br>PD2 (2)<br>PE1 (5)<br>PH0 (1)<br>PB5 (3) | I/O | TTL | Capture/Compare/PWM 6. |
| CCP7 | 11<br>13<br>75<br>85<br>90 | PD1 (6)<br>PD3 (2)<br>PE3 (5)<br>PH1 (1)<br>PB6 (2) | I/O | TTL | Capture/Compare/PWM 7. |
| CMOD0 | 65 | fixed | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| CMOD1 | 76 | fixed | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |
| GND | 9<br>15<br>21<br>33<br>39<br>45<br>54<br>57<br>63<br>69<br>82<br>87<br>94 | fixed | - | Power | Ground reference for logic and I/O pins. |
| GNDA | 4<br>97 | fixed | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| HIB | 51 | fixed | O | OD | An output that indicates the processor is in Hibernate mode. |
| I2C0SCL | 70 | PB2 (1) | I/O | OD | $I^2C$ module 0 clock. |
| I2C0SDA | 71 | PB3 (1) | I/O | OD | $I^2C$ module 0 data. |
| I2C1SCL | 19<br>26<br>34 | PG0 (3)<br>PA0 (8)<br>PA6 (1) | I/O | OD | $I^2C$ module 1 clock. |
| I2C1SDA | 18<br>27<br>35 | PG1 (3)<br>PA1 (8)<br>PA7 (1) | I/O | OD | $I^2C$ module 1 data. |
| LDO | 7 | fixed | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| NMI | 89 | PB7 (4) | I | TTL | Non-maskable interrupt. |
| OSC0 | 48 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | 49 | fixed | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |

**Table 17-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| PA0 | 26 | - | I/O | TTL | GPIO port A bit 0. |
| PA1 | 27 | - | I/O | TTL | GPIO port A bit 1. |
| PA2 | 28 | - | I/O | TTL | GPIO port A bit 2. |
| PA3 | 29 | - | I/O | TTL | GPIO port A bit 3. |
| PA4 | 30 | - | I/O | TTL | GPIO port A bit 4. |
| PA5 | 31 | - | I/O | TTL | GPIO port A bit 5. |
| PA6 | 34 | - | I/O | TTL | GPIO port A bit 6. |
| PA7 | 35 | - | I/O | TTL | GPIO port A bit 7. |
| PB0 | 66 | - | I/O | TTL | GPIO port B bit 0. This pin is not 5-V tolerant. |
| PB1 | 67 | - | I/O | TTL | GPIO port B bit 1. This pin is not 5-V tolerant. |
| PB2 | 70 | - | I/O | TTL | GPIO port B bit 2. |
| PB3 | 71 | - | I/O | TTL | GPIO port B bit 3. |
| PB4 | 92 | - | I/O | TTL | GPIO port B bit 4. |
| PB5 | 91 | - | I/O | TTL | GPIO port B bit 5. |
| PB6 | 90 | - | I/O | TTL | GPIO port B bit 6. |
| PB7 | 89 | - | I/O | TTL | GPIO port B bit 7. |
| PC0 | 80 | - | I/O | TTL | GPIO port C bit 0. |
| PC1 | 79 | - | I/O | TTL | GPIO port C bit 1. |
| PC2 | 78 | - | I/O | TTL | GPIO port C bit 2. |
| PC3 | 77 | - | I/O | TTL | GPIO port C bit 3. |
| PC4 | 25 | - | I/O | TTL | GPIO port C bit 4. |
| PC5 | 24 | - | I/O | TTL | GPIO port C bit 5. |
| PC6 | 23 | - | I/O | TTL | GPIO port C bit 6. |
| PC7 | 22 | - | I/O | TTL | GPIO port C bit 7. |
| PD0 | 10 | - | I/O | TTL | GPIO port D bit 0. |
| PD1 | 11 | - | I/O | TTL | GPIO port D bit 1. |
| PD2 | 12 | - | I/O | TTL | GPIO port D bit 2. |
| PD3 | 13 | - | I/O | TTL | GPIO port D bit 3. |
| PD4 | 95 | - | I/O | TTL | GPIO port D bit 4. |
| PD5 | 96 | - | I/O | TTL | GPIO port D bit 5. |
| PD6 | 99 | - | I/O | TTL | GPIO port D bit 6. |
| PD7 | 100 | - | I/O | TTL | GPIO port D bit 7. |
| PE0 | 72 | - | I/O | TTL | GPIO port E bit 0. |
| PE1 | 73 | - | I/O | TTL | GPIO port E bit 1. |
| PE2 | 74 | - | I/O | TTL | GPIO port E bit 2. |
| PE3 | 75 | - | I/O | TTL | GPIO port E bit 3. |
| PE4 | 6 | - | I/O | TTL | GPIO port E bit 4. |
| PE5 | 5 | - | I/O | TTL | GPIO port E bit 5. |
| PE6 | 2 | - | I/O | TTL | GPIO port E bit 6. |
| PE7 | 1 | - | I/O | TTL | GPIO port E bit 7. |
| PF0 | 47 | - | I/O | TTL | GPIO port F bit 0. |

**Table 17-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| PF1 | 61 | - | I/O | TTL | GPIO port F bit 1. |
| PF2 | 60 | - | I/O | TTL | GPIO port F bit 2. |
| PF3 | 59 | - | I/O | TTL | GPIO port F bit 3. |
| PF4 | 58 | - | I/O | TTL | GPIO port F bit 4. |
| PF5 | 46 | - | I/O | TTL | GPIO port F bit 5. |
| PF6 | 43 | - | I/O | TTL | GPIO port F bit 6. |
| PF7 | 42 | - | I/O | TTL | GPIO port F bit 7. |
| PG0 | 19 | - | I/O | TTL | GPIO port G bit 0. |
| PG1 | 18 | - | I/O | TTL | GPIO port G bit 1. |
| PG2 | 17 | - | I/O | TTL | GPIO port G bit 2. |
| PG3 | 16 | - | I/O | TTL | GPIO port G bit 3. |
| PG4 | 41 | - | I/O | TTL | GPIO port G bit 4. |
| PG5 | 40 | - | I/O | TTL | GPIO port G bit 5. |
| PG6 | 37 | - | I/O | TTL | GPIO port G bit 6. |
| PG7 | 36 | - | I/O | TTL | GPIO port G bit 7. |
| PH0 | 86 | - | I/O | TTL | GPIO port H bit 0. |
| PH1 | 85 | - | I/O | TTL | GPIO port H bit 1. |
| PH2 | 84 | - | I/O | TTL | GPIO port H bit 2. |
| PH3 | 83 | - | I/O | TTL | GPIO port H bit 3. |
| $\overline{\text{RST}}$ | 64 | fixed | I | TTL | System reset input. |
| SSI0Clk | 28 | PA2 (1) | I/O | TTL | SSI module 0 clock. |
| SSI0Fss | 29 | PA3 (1) | I/O | TTL | SSI module 0 frame. |
| SSI0Rx | 30 | PA4 (1) | I | TTL | SSI module 0 receive. |
| SSI0Tx | 31 | PA5 (1) | O | TTL | SSI module 0 transmit. |
| SSI1Clk | 60<br>72 | PF2 (9)<br>PE0 (2) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | 59<br>73 | PF3 (9)<br>PE1 (2) | I/O | TTL | SSI module 1 frame. |
| SSI1Rx | 58<br>74 | PF4 (9)<br>PE2 (2) | I | TTL | SSI module 1 receive. |
| SSI1Tx | 46<br>75 | PF5 (9)<br>PE3 (2) | O | TTL | SSI module 1 transmit. |
| SWCLK | 80 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| SWDIO | 79 | PC1 (3) | I/O | TTL | JTAG TMS and SWDIO. |
| SWO | 77 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TCK | 80 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| TDI | 78 | PC2 (3) | I | TTL | JTAG TDI. |
| TDO | 77 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TMS | 79 | PC1 (3) | I | TTL | JTAG TMS and SWDIO. |
| U0Rx | 26 | PA0 (1) | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |

**Table 17-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| U0Tx | 27 | PA1 (1) | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U1DSR | 47 | PF0 (9) | I | TTL | UART module 1 Data Set Ready modem output control line. |
| U1Rx | 10<br>12<br>23<br>26<br>66<br>92 | PD0 (5)<br>PD2 (1)<br>PC6 (5)<br>PA0 (9)<br>PB0 (5)<br>PB4 (7) | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| U1Tx | 11<br>13<br>22<br>27<br>67<br>91 | PD1 (5)<br>PD3 (1)<br>PC7 (5)<br>PA1 (9)<br>PB1 (5)<br>PB5 (7) | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U2Rx | 10<br>19<br>92<br>96 | PD0 (4)<br>PG0 (1)<br>PB4 (4)<br>PD5 (9) | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| U2Tx | 6<br>11<br>18<br>99 | PE4 (5)<br>PD1 (4)<br>PG1 (1)<br>PD6 (9) | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| VBAT | 55 | 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 | 8<br>20<br>32<br>44<br>56<br>68<br>81<br>93 | fixed | - | Power | Positive supply for I/O and some logic. |
| VDDA | 3<br>98 | fixed | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |
| VDDC | 14<br>38<br>62<br>88 | fixed | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| VREFA | 90 | PB6 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |

**Table 17-3. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| WAKE | 50 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| XOSC0 | 52 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| XOSC1 | 53 | fixed | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 17-4. Signals by Function, Except for GPIO**

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| ADC | AIN0 | 1 | I | Analog | Analog-to-digital converter input 0. |
| | AIN1 | 2 | I | Analog | Analog-to-digital converter input 1. |
| | AIN2 | 5 | I | Analog | Analog-to-digital converter input 2. |
| | AIN3 | 6 | I | Analog | Analog-to-digital converter input 3. |
| | AIN4 | 100 | I | Analog | Analog-to-digital converter input 4. |
| | AIN5 | 99 | I | Analog | Analog-to-digital converter input 5. |
| | AIN6 | 96 | I | Analog | Analog-to-digital converter input 6. |
| | AIN7 | 95 | I | Analog | Analog-to-digital converter input 7. |
| | AIN8 | 75 | I | Analog | Analog-to-digital converter input 8. |
| | AIN9 | 74 | I | Analog | Analog-to-digital converter input 9. |
| | AIN10 | 92 | I | Analog | Analog-to-digital converter input 10. |
| | AIN11 | 91 | I | Analog | Analog-to-digital converter input 11. |
| | AIN12 | 13 | I | Analog | Analog-to-digital converter input 12. |
| | AIN13 | 12 | I | Analog | Analog-to-digital converter input 13. |
| | AIN14 | 11 | I | Analog | Analog-to-digital converter input 14. |
| | AIN15 | 10 | I | Analog | Analog-to-digital converter input 15. |
| | VREFA | 90 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |

**Table 17-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| General-Purpose Timers | CCP0 | 13<br>22<br>23<br>58<br>66<br>70<br>91<br>95 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP1 | 24<br>25<br>34<br>43<br>67<br>75<br>90<br>100 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP2 | 6<br>11<br>25<br>46<br>67<br>73<br>74<br>91<br>96 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP3 | 6<br>23<br>24<br>35<br>41<br>61<br>70<br>72<br>95 | I/O | TTL | Capture/Compare/PWM 3. |
| | CCP4 | 22<br>25<br>35<br>42<br>74<br>96 | I/O | TTL | Capture/Compare/PWM 4. |
| | CCP5 | 5<br>12<br>25<br>36<br>40<br>90<br>91 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | 10<br>12<br>73<br>86<br>91 | I/O | TTL | Capture/Compare/PWM 6. |
| | CCP7 | | I/O | TTL | Capture/Compare/PWM 7. |

**Table 17-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| | | 11<br>13<br>75<br>85<br>90 | | | |
| Hibernate | HIB | 51 | O | OD | An output that indicates the processor is in Hibernate mode. |
| | VBAT | 55 | - | 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 | 50 | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| | XOSC0 | 52 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| | XOSC1 | 53 | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| I2C | I2C0SCL | 70 | I/O | OD | $I^2C$ module 0 clock. |
| | I2C0SDA | 71 | I/O | OD | $I^2C$ module 0 data. |
| | I2C1SCL | 19<br>26<br>34 | I/O | OD | $I^2C$ module 1 clock. |
| | I2C1SDA | 18<br>27<br>35 | I/O | OD | $I^2C$ module 1 data. |
| JTAG/SWD/SWO | SWCLK | 80 | I | TTL | JTAG/SWD CLK. |
| | SWDIO | 79 | I/O | TTL | JTAG TMS and SWDIO. |
| | SWO | 77 | O | TTL | JTAG TDO and SWO. |
| | TCK | 80 | I | TTL | JTAG/SWD CLK. |
| | TDI | 78 | I | TTL | JTAG TDI. |
| | TDO | 77 | O | TTL | JTAG TDO and SWO. |
| | TMS | 79 | I | TTL | JTAG TMS and SWDIO. |

**Table 17-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| Power | GND | 9<br>15<br>21<br>33<br>39<br>45<br>54<br>57<br>63<br>69<br>82<br>87<br>94 | - | Power | Ground reference for logic and I/O pins. |
| | GNDA | 4<br>97 | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| | LDO | 7 | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| | VDD | 8<br>20<br>32<br>44<br>56<br>68<br>81<br>93 | - | Power | Positive supply for I/O and some logic. |
| | VDDA | 3<br>98 | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |
| | VDDC | 14<br>38<br>62<br>88 | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| SSI | SSI0Clk | 28 | I/O | TTL | SSI module 0 clock. |
| | SSI0Fss | 29 | I/O | TTL | SSI module 0 frame. |
| | SSI0Rx | 30 | I | TTL | SSI module 0 receive. |
| | SSI0Tx | 31 | O | TTL | SSI module 0 transmit. |
| | SSI1Clk | 60<br>72 | I/O | TTL | SSI module 1 clock. |
| | SSI1Fss | 59<br>73 | I/O | TTL | SSI module 1 frame. |
| | SSI1Rx | 58<br>74 | I | TTL | SSI module 1 receive. |
| | SSI1Tx | 46<br>75 | O | TTL | SSI module 1 transmit. |

**Table 17-4. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| System Control & Clocks | CMOD0 | 65 | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| | CMOD1 | 76 | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |
| | NMI | 89 | I | TTL | Non-maskable interrupt. |
| | OSC0 | 48 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| | OSC1 | 49 | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| | $\overline{\text{RST}}$ | 64 | I | TTL | System reset input. |
| UART | U0Rx | 26 | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U0Tx | 27 | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U1DSR | 47 | I | TTL | UART module 1 Data Set Ready modem output control line. |
| | U1Rx | 10<br>12<br>23<br>26<br>66<br>92 | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U1Tx | 11<br>13<br>22<br>27<br>67<br>91 | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U2Rx | 10<br>19<br>92<br>96 | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U2Tx | 6<br>11<br>18<br>99 | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 17-5. GPIO Pins and Alternate Functions**

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PA0 | 26 | - | U0Rx | - | - | - | - | - | - | I2C1SCL | U1Rx | - | - |
| PA1 | 27 | - | U0Tx | - | - | - | - | - | - | I2C1SDA | U1Tx | - | - |
| PA2 | 28 | - | SSI0Clk | - | - | - | - | - | - | - | - | - | - |
| PA3 | 29 | - | SSI0Fss | - | - | - | - | - | - | - | - | - | - |
| PA4 | 30 | - | SSI0Rx | - | - | - | - | - | - | - | - | - | - |
| PA5 | 31 | - | SSI0Tx | - | - | - | - | - | - | - | - | - | - |
| PA6 | 34 | - | I2C1SCL | CCP1 | - | - | - | - | - | - | - | - | - |
| PA7 | 35 | - | I2C1SDA | CCP4 | - | - | - | - | CCP3 | - | - | - | - |

**Table 17-5. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PB0 | 66 | - | CCP0 | - | - | - | U1Rx | - | - | - | - | - | - |
| PB1 | 67 | - | CCP2 | - | - | CCP1 | U1Tx | - | - | - | - | - | - |
| PB2 | 70 | - | I2C0SCL | - | - | CCP3 | CCP0 | - | - | - | - | - | - |
| PB3 | 71 | - | I2C0SDA | - | - | - | - | - | - | - | - | - | - |
| PB4 | 92 | AIN10 | - | - | - | U2Rx | - | - | U1Rx | - | - | - | - |
| PB5 | 91 | AIN11 | - | CCP5 | CCP6 | CCP0 | - | CCP2 | U1Tx | - | - | - | - |
| PB6 | 90 | VREFA | CCP1 | CCP7 | - | - | - | CCP5 | - | - | - | - | - |
| PB7 | 89 | - | - | - | - | NMI | - | - | - | - | - | - | - |
| PC0 | 80 | - | - | - | TCK SWCLK | - | - | - | - | - | - | - | - |
| PC1 | 79 | - | - | - | TMS SWDIO | - | - | - | - | - | - | - | - |
| PC2 | 78 | - | - | - | TDI | - | - | - | - | - | - | - | - |
| PC3 | 77 | - | - | - | TDO SWO | - | - | - | - | - | - | - | - |
| PC4 | 25 | - | CCP5 | - | - | - | CCP2 | CCP4 | - | - | CCP1 | - | - |
| PC5 | 24 | - | CCP1 | - | - | - | CCP3 | - | - | - | - | - | - |
| PC6 | 23 | - | CCP3 | - | - | - | U1Rx | CCP0 | - | - | - | - | - |
| PC7 | 22 | - | CCP4 | - | - | CCP0 | U1Tx | - | - | - | - | - | - |
| PD0 | 10 | AIN15 | - | - | - | U2Rx | U1Rx | CCP6 | - | - | - | - | - |
| PD1 | 11 | AIN14 | - | - | - | U2Tx | U1Tx | CCP7 | - | - | - | CCP2 | - |
| PD2 | 12 | AIN13 | U1Rx | CCP6 | - | CCP5 | - | - | - | - | - | - | - |
| PD3 | 13 | AIN12 | U1Tx | CCP7 | - | CCP0 | - | - | - | - | - | - | - |
| PD4 | 95 | AIN7 | CCP0 | CCP3 | - | - | - | - | - | - | - | - | - |
| PD5 | 96 | AIN6 | CCP2 | CCP4 | - | - | - | - | - | - | U2Rx | - | - |
| PD6 | 99 | AIN5 | - | - | - | - | - | - | - | - | U2Tx | - | - |
| PD7 | 100 | AIN4 | - | - | CCP1 | - | - | - | - | - | - | - | - |
| PE0 | 72 | - | - | SSI1Clk | CCP3 | - | - | - | - | - | - | - | - |
| PE1 | 73 | - | - | SSI1Fss | - | CCP2 | CCP6 | - | - | - | - | - | - |
| PE2 | 74 | AIN9 | CCP4 | SSI1Rx | - | - | CCP2 | - | - | - | - | - | - |
| PE3 | 75 | AIN8 | CCP1 | SSI1Tx | - | - | CCP7 | - | - | - | - | - | - |
| PE4 | 6 | AIN3 | CCP3 | - | - | - | U2Tx | CCP2 | - | - | - | - | - |
| PE5 | 5 | AIN2 | CCP5 | - | - | - | - | - | - | - | - | - | - |
| PE6 | 2 | AIN1 | - | - | - | - | - | - | - | - | - | - | - |
| PE7 | 1 | AIN0 | - | - | - | - | - | - | - | - | - | - | - |
| PF0 | 47 | - | - | - | - | - | - | - | - | - | U1DSR | - | - |
| PF1 | 61 | - | - | - | - | - | - | - | - | - | - | CCP3 | - |
| PF2 | 60 | - | - | - | - | - | - | - | - | - | SSI1Clk | - | - |
| PF3 | 59 | - | - | - | - | - | - | - | - | - | SSI1Fss | - | - |
| PF4 | 58 | - | CCP0 | - | - | - | - | - | - | - | SSI1Rx | - | - |
| PF5 | 46 | - | CCP2 | - | - | - | - | - | - | - | SSI1Tx | - | - |

**Table 17-5. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PF6 | 43 | - | CCP1 | - | - | - | - | - | - | - | - | - | - |
| PF7 | 42 | - | CCP4 | - | - | - | - | - | - | - | - | - | - |
| PG0 | 19 | - | U2Rx | - | I2C1SCL | - | - | - | - | - | - | - | - |
| PG1 | 18 | - | U2Tx | - | I2C1SDA | - | - | - | - | - | - | - | - |
| PG2 | 17 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG3 | 16 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG4 | 41 | - | CCP3 | - | - | - | - | - | - | - | - | - | - |
| PG5 | 40 | - | CCP5 | - | - | - | - | - | - | - | - | - | - |
| PG6 | 37 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG7 | 36 | - | - | - | - | - | - | - | - | CCP5 | - | - | - |
| PH0 | 86 | - | CCP6 | - | - | - | - | - | - | - | - | - | - |
| PH1 | 85 | - | CCP7 | - | - | - | - | - | - | - | - | - | - |
| PH2 | 84 | - | - | - | - | - | - | - | - | - | - | - | - |
| PH3 | 83 | - | - | - | - | - | - | - | - | - | - | - | - |

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

**Table 17-6. Possible Pin Assignments for Alternate Functions**

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| one | AIN0 | PE7 |
| | AIN1 | PE6 |
| | AIN10 | PB4 |
| | AIN11 | PB5 |
| | AIN12 | PD3 |
| | AIN13 | PD2 |
| | AIN14 | PD1 |
| | AIN15 | PD0 |
| | AIN2 | PE5 |
| | AIN3 | PE4 |
| | AIN4 | PD7 |
| | AIN5 | PD6 |
| | AIN6 | PD5 |
| | AIN7 | PD4 |
| | AIN8 | PE3 |
| | AIN9 | PE2 |
| | I2C0SCL | PB2 |
| | I2C0SDA | PB3 |
| | NMI | PB7 |
| | SSI0Clk | PA2 |
| | SSI0Fss | PA3 |
| | SSI0Rx | PA4 |
| | SSI0Tx | PA5 |
| | SWCLK | PC0 |
| | SWDIO | PC1 |
| | SWO | PC3 |
| | TCK | PC0 |
| | TDI | PC2 |
| | TDO | PC3 |
| | TMS | PC1 |
| | U0Rx | PA0 |
| | U0Tx | PA1 |
| | U1DSR | PF0 |
| | VREFA | PB6 |
| two | SSI1Clk | PE0 PF2 |
| | SSI1Fss | PE1 PF3 |
| | SSI1Rx | PE2 PF4 |
| | SSI1Tx | PE3 PF5 |
| three | I2C1SCL | PA0 PA6 PG0 |
| | I2C1SDA | PA1 PA7 PG1 |

**Table 17-6. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| four | U2Rx | PB4 PD0 PD5 PG0 |
| | U2Tx | PD1 PD6 PE4 PG1 |
| five | CCP6 | PB5 PD0 PD2 PE1 PH0 |
| | CCP7 | PB6 PD1 PD3 PE3 PH1 |
| six | CCP4 | PA7 PC4 PC7 PD5 PE2 PF7 |
| | U1Rx | PA0 PB0 PB4 PC6 PD0 PD2 |
| | U1Tx | PA1 PB1 PB5 PC7 PD1 PD3 |
| seven | CCP5 | PB5 PB6 PC4 PD2 PE5 PG5 PG7 |
| eight | CCP0 | PB0 PB2 PB5 PC6 PC7 PD3 PD4 PF4 |
| | CCP1 | PA6 PB1 PB6 PC4 PC5 PD7 PE3 PF6 |
| nine | CCP2 | PB1 PB5 PC4 PD1 PD5 PE1 PE2 PE4 PF5 |
| | CCP3 | PA7 PB2 PC5 PC6 PD4 PE0 PE4 PF1 PG4 |

## 17.2 108-Ball BGA Package Pin Tables

**Table 17-7. Signals by Pin Number**

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| A1 | PE6 | I/O | TTL | GPIO port E bit 6. |
| | AIN1 | I | Analog | Analog-to-digital converter input 1. |
| A2 | PD7 | I/O | TTL | GPIO port D bit 7. |
| | AIN4 | I | Analog | Analog-to-digital converter input 4. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| A3 | PD6 | I/O | TTL | GPIO port D bit 6. |
| | AIN5 | I | Analog | Analog-to-digital converter input 5. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| A4 | PD4 | I/O | TTL | GPIO port D bit 4. |
| | AIN7 | I | Analog | Analog-to-digital converter input 7. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| A5 | GNDA | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| A6 | PB4 | I/O | TTL | GPIO port B bit 4. |
| | AIN10 | I | Analog | Analog-to-digital converter input 10. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |

**Table 17-7. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| A7 | PB6 | I/O | TTL | GPIO port B bit 6. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | VREFA | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |
| A8 | PB7 | I/O | TTL | GPIO port B bit 7. |
| | NMI | I | TTL | Non-maskable interrupt. |
| A9 | PC0 | I/O | TTL | GPIO port C bit 0. |
| | SWCLK | I | TTL | JTAG/SWD CLK. |
| | TCK | I | TTL | JTAG/SWD CLK. |
| A10 | PC3 | I/O | TTL | GPIO port C bit 3. |
| | SWO | O | TTL | JTAG TDO and SWO. |
| | TDO | O | TTL | JTAG TDO and SWO. |
| A11 | PE0 | I/O | TTL | GPIO port E bit 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | SSI1Clk | I/O | TTL | SSI module 1 clock. |
| A12 | PE3 | I/O | TTL | GPIO port E bit 3. |
| | AIN8 | I | Analog | Analog-to-digital converter input 8. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | SSI1Tx | O | TTL | SSI module 1 transmit. |
| B1 | PE7 | I/O | TTL | GPIO port E bit 7. |
| | AIN0 | I | Analog | Analog-to-digital converter input 0. |
| B2 | PE4 | I/O | TTL | GPIO port E bit 4. |
| | AIN3 | I | Analog | Analog-to-digital converter input 3. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| B3 | PE5 | I/O | TTL | GPIO port E bit 5. |
| | AIN2 | I | Analog | Analog-to-digital converter input 2. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| B4 | PD5 | I/O | TTL | GPIO port D bit 5. |
| | AIN6 | I | Analog | Analog-to-digital converter input 6. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |

**Table 17-7. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| B5 | GNDA | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| B6 | GND | - | Power | Ground reference for logic and I/O pins. |
| B7 | PB5 | I/O | TTL | GPIO port B bit 5. |
| | AIN11 | I | Analog | Analog-to-digital converter input 11. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| B8 | PC2 | I/O | TTL | GPIO port C bit 2. |
| | TDI | I | TTL | JTAG TDI. |
| B9 | PC1 | I/O | TTL | GPIO port C bit 1. |
| | SWDIO | I/O | TTL | JTAG TMS and SWDIO. |
| | TMS | I | TTL | JTAG TMS and SWDIO. |
| B10 | CMOD1 | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |
| B11 | PE2 | I/O | TTL | GPIO port E bit 2. |
| | AIN9 | I | Analog | Analog-to-digital converter input 9. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | SSI1Rx | I | TTL | SSI module 1 receive. |
| B12 | PE1 | I/O | TTL | GPIO port E bit 1. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame. |
| C1 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| C2 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| C3 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| C4 | GND | - | Power | Ground reference for logic and I/O pins. |
| C5 | GND | - | Power | Ground reference for logic and I/O pins. |
| C6 | VDDA | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |
| C7 | VDDA | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |

**Table 17-7. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| C8 | PH1 | I/O | TTL | GPIO port H bit 1. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| C9 | PH0 | I/O | TTL | GPIO port H bit 0. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| C10 | PG7 | I/O | TTL | GPIO port G bit 7. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| C11 | PB2 | I/O | TTL | GPIO port B bit 2. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | I2C0SCL | I/O | OD | $I^2C$ module 0 clock. |
| C12 | PB3 | I/O | TTL | GPIO port B bit 3. |
| | I2C0SDA | I/O | OD | $I^2C$ module 0 data. |
| D1 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| D2 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| D3 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| D10 | PH3 | I/O | TTL | GPIO port H bit 3. |
| D11 | PH2 | I/O | TTL | GPIO port H bit 2. |
| D12 | PB1 | I/O | TTL | GPIO port B bit 1. This pin is not 5-V tolerant. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| E1 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| E2 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| E3 | LDO | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| E10 | VDD | - | Power | Positive supply for I/O and some logic. |
| E11 | CMOD0 | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| E12 | PB0 | I/O | TTL | GPIO port B bit 0. This pin is not 5-V tolerant. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| F1 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| F2 | NC | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| F3 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| F10 | GND | - | Power | Ground reference for logic and I/O pins. |
| F11 | GND | - | Power | Ground reference for logic and I/O pins. |
| F12 | GND | - | Power | Ground reference for logic and I/O pins. |

**Table 17-7. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| G1 | PD0 | I/O | TTL | GPIO port D bit 0. |
| | AIN15 | I | Analog | Analog-to-digital converter input 15. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| G2 | PD1 | I/O | TTL | GPIO port D bit 1. |
| | AIN14 | I | Analog | Analog-to-digital converter input 14. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| G3 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| G10 | VDD | - | Power | Positive supply for I/O and some logic. |
| G11 | VDD | - | Power | Positive supply for I/O and some logic. |
| G12 | VDD | - | Power | Positive supply for I/O and some logic. |
| H1 | PD3 | I/O | TTL | GPIO port D bit 3. |
| | AIN12 | I | Analog | Analog-to-digital converter input 12. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| H2 | PD2 | I/O | TTL | GPIO port D bit 2. |
| | AIN13 | I | Analog | Analog-to-digital converter input 13. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| H3 | GND | - | Power | Ground reference for logic and I/O pins. |
| H10 | VDD | - | Power | Positive supply for I/O and some logic. |
| H11 | $\overline{\text{RST}}$ | I | TTL | System reset input. |
| H12 | PF1 | I/O | TTL | GPIO port F bit 1. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| J1 | PG2 | I/O | TTL | GPIO port G bit 2. |
| J2 | PG3 | I/O | TTL | GPIO port G bit 3. |
| J3 | GND | - | Power | Ground reference for logic and I/O pins. |
| J10 | GND | - | Power | Ground reference for logic and I/O pins. |
| J11 | PF2 | I/O | TTL | GPIO port F bit 2. |
| | SSI1Clk | I/O | TTL | SSI module 1 clock. |

**Table 17-7. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| J12 | PF3 | I/O | TTL | GPIO port F bit 3. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame. |
| K1 | PG0 | I/O | TTL | GPIO port G bit 0. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| K2 | PG1 | I/O | TTL | GPIO port G bit 1. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| K3 | PG4 | I/O | TTL | GPIO port G bit 4. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| K4 | PF7 | I/O | TTL | GPIO port F bit 7. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| K5 | GND | - | Power | Ground reference for logic and I/O pins. |
| 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 | VDD | - | Power | Positive supply for I/O and some logic. |
| K10 | GND | - | Power | Ground reference for logic and I/O pins. |
| K11 | XOSC0 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| K12 | XOSC1 | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| L1 | PC4 | I/O | TTL | GPIO port C bit 4. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| L2 | PC7 | I/O | TTL | GPIO port C bit 7. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| L3 | PA0 | I/O | TTL | GPIO port A bit 0. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. |
| | U0Rx | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| L4 | PA3 | I/O | TTL | GPIO port A bit 3. |
| | SSI0Fss | I/O | TTL | SSI module 0 frame. |

**Table 17-7. Signals by Pin Number *(continued)***

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| L5 | PA4 | I/O | TTL | GPIO port A bit 4. |
| | SSI0Rx | I | TTL | SSI module 0 receive. |
| L6 | PA6 | I/O | TTL | GPIO port A bit 6. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | I2C1SCL | I/O | OD | $I^2C$ module 1 clock. |
| L7 | PG6 | I/O | TTL | GPIO port G bit 6. |
| L8 | PF5 | I/O | TTL | GPIO port F bit 5. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | SSI1Tx | O | TTL | SSI module 1 transmit. |
| L9 | PF4 | I/O | TTL | GPIO port F bit 4. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | SSI1Rx | I | TTL | SSI module 1 receive. |
| L10 | GND | - | Power | Ground reference for logic and I/O pins. |
| L11 | OSC0 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| L12 | 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. |
| M1 | PC5 | I/O | TTL | GPIO port C bit 5. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| M2 | PC6 | I/O | TTL | GPIO port C bit 6. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| M3 | PA1 | I/O | TTL | GPIO port A bit 1. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| | U0Tx | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| M4 | PA2 | I/O | TTL | GPIO port A bit 2. |
| | SSI0Clk | I/O | TTL | SSI module 0 clock. |
| M5 | PA5 | I/O | TTL | GPIO port A bit 5. |
| | SSI0Tx | O | TTL | SSI module 0 transmit. |
| M6 | PA7 | I/O | TTL | GPIO port A bit 7. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | I2C1SDA | I/O | OD | $I^2C$ module 1 data. |
| M7 | PG5 | I/O | TTL | GPIO port G bit 5. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| M8 | PF6 | I/O | TTL | GPIO port F bit 6. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |

### Table 17-7. Signals by Pin Number *(continued)*

| Pin Number | Pin Name | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|
| M9 | PF0 | I/O | TTL | GPIO port F bit 0. |
| | U1DSR | I | TTL | UART module 1 Data Set Ready modem output control line. |
| M10 | WAKE | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| M11 | OSC1 | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| M12 | HIB | O | OD | An output that indicates the processor is in Hibernate mode. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

### Table 17-8. Signals by Signal Name

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| AIN0 | B1 | PE7 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | A1 | PE6 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | B3 | PE5 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | B2 | PE4 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | A2 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | A3 | 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 | A12 | PE3 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | B11 | PE2 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | A6 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | B7 | PB5 | I | Analog | Analog-to-digital converter input 11. |
| AIN12 | H1 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | H2 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | G2 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | G1 | PD0 | I | Analog | Analog-to-digital converter input 15. |
| CCP0 | H1<br>L2<br>M2<br>L9<br>E12<br>C11<br>B7<br>A4 | PD3 (4)<br>PC7 (4)<br>PC6 (6)<br>PF4 (1)<br>PB0 (1)<br>PB2 (5)<br>PB5 (4)<br>PD4 (1) | I/O | TTL | Capture/Compare/PWM 0. |
| CCP1 | M1<br>L1<br>L6<br>M8<br>D12<br>A12<br>A7<br>A2 | PC5 (1)<br>PC4 (9)<br>PA6 (2)<br>PF6 (1)<br>PB1 (4)<br>PE3 (1)<br>PB6 (1)<br>PD7 (3) | I/O | TTL | Capture/Compare/PWM 1. |

**Table 17-8. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| CCP2 | B2<br>G2<br>L1<br>L8<br>D12<br>B12<br>B11<br>B7<br>B4 | PE4 (6)<br>PD1 (10)<br>PC4 (5)<br>PF5 (1)<br>PB1 (1)<br>PE1 (4)<br>PE2 (5)<br>PB5 (6)<br>PD5 (1) | I/O | TTL | Capture/Compare/PWM 2. |
| CCP3 | B2<br>M2<br>M1<br>M6<br>K3<br>H12<br>C11<br>A11<br>A4 | PE4 (1)<br>PC6 (1)<br>PC5 (5)<br>PA7 (7)<br>PG4 (1)<br>PF1 (10)<br>PB2 (4)<br>PE0 (3)<br>PD4 (2) | I/O | TTL | Capture/Compare/PWM 3. |
| CCP4 | L2<br>L1<br>M6<br>K4<br>B11<br>B4 | PC7 (1)<br>PC4 (6)<br>PA7 (2)<br>PF7 (1)<br>PE2 (1)<br>PD5 (2) | I/O | TTL | Capture/Compare/PWM 4. |
| CCP5 | B3<br>H2<br>L1<br>C10<br>M7<br>A7<br>B7 | PE5 (1)<br>PD2 (4)<br>PC4 (1)<br>PG7 (8)<br>PG5 (1)<br>PB6 (6)<br>PB5 (2) | I/O | TTL | Capture/Compare/PWM 5. |
| CCP6 | G1<br>H2<br>B12<br>C9<br>B7 | PD0 (6)<br>PD2 (2)<br>PE1 (5)<br>PH0 (1)<br>PB5 (3) | I/O | TTL | Capture/Compare/PWM 6. |
| CCP7 | G2<br>H1<br>A12<br>C8<br>A7 | PD1 (6)<br>PD3 (2)<br>PE3 (5)<br>PH1 (1)<br>PB6 (2) | I/O | TTL | Capture/Compare/PWM 7. |
| CMOD0 | E11 | fixed | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| CMOD1 | B10 | fixed | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |

**Table 17-8. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| GND | C4<br>H3<br>C5<br>J3<br>K6<br>K5<br>L10<br>K10<br>F10<br>J10<br>F11<br>B6<br>F12 | fixed | - | Power | Ground reference for logic and I/O pins. |
| GNDA | A5<br>B5 | fixed | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| HIB | M12 | fixed | O | OD | An output that indicates the processor is in Hibernate mode. |
| I2C0SCL | C11 | PB2 (1) | I/O | OD | $I^2C$ module 0 clock. |
| I2C0SDA | C12 | PB3 (1) | I/O | OD | $I^2C$ module 0 data. |
| I2C1SCL | K1<br>L3<br>L6 | PG0 (3)<br>PA0 (8)<br>PA6 (1) | I/O | OD | $I^2C$ module 1 clock. |
| I2C1SDA | K2<br>M3<br>M6 | PG1 (3)<br>PA1 (8)<br>PA7 (1) | I/O | OD | $I^2C$ module 1 data. |
| LDO | E3 | fixed | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| NC | C1<br>C2<br>D2<br>D1<br>E1<br>E2<br>F1<br>F2 | fixed | - | - | No connect. Leave the pin electrically unconnected/isolated. |
| NMI | A8 | PB7 (4) | I | TTL | Non-maskable interrupt. |
| OSC0 | L11 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | M11 | fixed | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| PA0 | L3 | - | I/O | TTL | GPIO port A bit 0. |
| PA1 | M3 | - | I/O | TTL | GPIO port A bit 1. |
| PA2 | M4 | - | I/O | TTL | GPIO port A bit 2. |
| PA3 | L4 | - | I/O | TTL | GPIO port A bit 3. |
| PA4 | L5 | - | I/O | TTL | GPIO port A bit 4. |

**Table 17-8. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| PA5 | M5 | - | I/O | TTL | GPIO port A bit 5. |
| PA6 | L6 | - | I/O | TTL | GPIO port A bit 6. |
| PA7 | M6 | - | I/O | TTL | GPIO port A bit 7. |
| PB0 | E12 | - | I/O | TTL | GPIO port B bit 0. This pin is not 5-V tolerant. |
| PB1 | D12 | - | I/O | TTL | GPIO port B bit 1. This pin is not 5-V tolerant. |
| PB2 | C11 | - | I/O | TTL | GPIO port B bit 2. |
| PB3 | C12 | - | I/O | TTL | GPIO port B bit 3. |
| PB4 | A6 | - | I/O | TTL | GPIO port B bit 4. |
| PB5 | B7 | - | I/O | TTL | GPIO port B bit 5. |
| PB6 | A7 | - | I/O | TTL | GPIO port B bit 6. |
| PB7 | A8 | - | I/O | TTL | GPIO port B bit 7. |
| PC0 | A9 | - | I/O | TTL | GPIO port C bit 0. |
| PC1 | B9 | - | I/O | TTL | GPIO port C bit 1. |
| PC2 | B8 | - | I/O | TTL | GPIO port C bit 2. |
| PC3 | A10 | - | I/O | TTL | GPIO port C bit 3. |
| PC4 | L1 | - | I/O | TTL | GPIO port C bit 4. |
| PC5 | M1 | - | I/O | TTL | GPIO port C bit 5. |
| PC6 | M2 | - | I/O | TTL | GPIO port C bit 6. |
| PC7 | L2 | - | I/O | TTL | GPIO port C bit 7. |
| PD0 | G1 | - | I/O | TTL | GPIO port D bit 0. |
| PD1 | G2 | - | I/O | TTL | GPIO port D bit 1. |
| PD2 | H2 | - | I/O | TTL | GPIO port D bit 2. |
| PD3 | H1 | - | 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 | A3 | - | I/O | TTL | GPIO port D bit 6. |
| PD7 | A2 | - | I/O | TTL | GPIO port D bit 7. |
| PE0 | A11 | - | I/O | TTL | GPIO port E bit 0. |
| PE1 | B12 | - | I/O | TTL | GPIO port E bit 1. |
| PE2 | B11 | - | I/O | TTL | GPIO port E bit 2. |
| PE3 | A12 | - | I/O | TTL | GPIO port E bit 3. |
| PE4 | B2 | - | I/O | TTL | GPIO port E bit 4. |
| PE5 | B3 | - | I/O | TTL | GPIO port E bit 5. |
| PE6 | A1 | - | I/O | TTL | GPIO port E bit 6. |
| PE7 | B1 | - | I/O | TTL | GPIO port E bit 7. |
| PF0 | M9 | - | I/O | TTL | GPIO port F bit 0. |
| PF1 | H12 | - | I/O | TTL | GPIO port F bit 1. |
| PF2 | J11 | - | I/O | TTL | GPIO port F bit 2. |
| PF3 | J12 | - | I/O | TTL | GPIO port F bit 3. |
| PF4 | L9 | - | I/O | TTL | GPIO port F bit 4. |
| PF5 | L8 | - | I/O | TTL | GPIO port F bit 5. |

**Table 17-8. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| PF6 | M8 | - | I/O | TTL | GPIO port F bit 6. |
| PF7 | K4 | - | I/O | TTL | GPIO port F bit 7. |
| PG0 | K1 | - | I/O | TTL | GPIO port G bit 0. |
| PG1 | K2 | - | I/O | TTL | GPIO port G bit 1. |
| PG2 | J1 | - | I/O | TTL | GPIO port G bit 2. |
| PG3 | J2 | - | I/O | TTL | GPIO port G bit 3. |
| PG4 | K3 | - | I/O | TTL | GPIO port G bit 4. |
| PG5 | M7 | - | I/O | TTL | GPIO port G bit 5. |
| PG6 | L7 | - | I/O | TTL | GPIO port G bit 6. |
| PG7 | C10 | - | I/O | TTL | GPIO port G bit 7. |
| PH0 | C9 | - | I/O | TTL | GPIO port H bit 0. |
| PH1 | C8 | - | I/O | TTL | GPIO port H bit 1. |
| PH2 | D11 | - | I/O | TTL | GPIO port H bit 2. |
| PH3 | D10 | - | I/O | TTL | GPIO port H bit 3. |
| $\overline{\text{RST}}$ | H11 | fixed | I | TTL | System reset input. |
| SSI0Clk | M4 | PA2 (1) | I/O | TTL | SSI module 0 clock. |
| SSI0Fss | L4 | PA3 (1) | I/O | TTL | SSI module 0 frame. |
| SSI0Rx | L5 | PA4 (1) | I | TTL | SSI module 0 receive. |
| SSI0Tx | M5 | PA5 (1) | O | TTL | SSI module 0 transmit. |
| SSI1Clk | J11<br>A11 | PF2 (9)<br>PE0 (2) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | J12<br>B12 | PF3 (9)<br>PE1 (2) | I/O | TTL | SSI module 1 frame. |
| SSI1Rx | L9<br>B11 | PF4 (9)<br>PE2 (2) | I | TTL | SSI module 1 receive. |
| SSI1Tx | L8<br>A12 | PF5 (9)<br>PE3 (2) | O | TTL | SSI module 1 transmit. |
| SWCLK | A9 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| SWDIO | B9 | PC1 (3) | I/O | TTL | JTAG TMS and SWDIO. |
| SWO | A10 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TCK | A9 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| TDI | B8 | PC2 (3) | I | TTL | JTAG TDI. |
| TDO | A10 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TMS | B9 | PC1 (3) | I | TTL | JTAG TMS and SWDIO. |
| U0Rx | L3 | PA0 (1) | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| U0Tx | M3 | PA1 (1) | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U1DSR | M9 | PF0 (9) | I | TTL | UART module 1 Data Set Ready modem output control line. |

**Table 17-8. Signals by Signal Name *(continued)***

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| U1Rx | G1<br>H2<br>M2<br>L3<br>E12<br>A6 | PD0 (5)<br>PD2 (1)<br>PC6 (5)<br>PA0 (9)<br>PB0 (5)<br>PB4 (7) | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| U1Tx | G2<br>H1<br>L2<br>M3<br>D12<br>B7 | PD1 (5)<br>PD3 (1)<br>PC7 (5)<br>PA1 (9)<br>PB1 (5)<br>PB5 (7) | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U2Rx | G1<br>K1<br>A6<br>B4 | PD0 (4)<br>PG0 (1)<br>PB4 (4)<br>PD5 (9) | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| U2Tx | B2<br>G2<br>K2<br>A3 | PE4 (5)<br>PD1 (4)<br>PG1 (1)<br>PD6 (9) | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| VBAT | L12 | 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 | K7<br>G12<br>K8<br>K9<br>H10<br>G10<br>E10<br>G11 | fixed | - | Power | Positive supply for I/O and some logic. |
| VDDA | C7<br>C6 | fixed | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |
| VDDC | F3<br>D3<br>G3<br>C3 | fixed | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| VREFA | A7 | PB6 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |
| WAKE | M10 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |

### Table 17-8. Signals by Signal Name *(continued)*

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| XOSC0 | K11 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| XOSC1 | K12 | fixed | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

### Table 17-9. Signals by Function, Except for GPIO

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| ADC | AIN0 | B1 | I | Analog | Analog-to-digital converter input 0. |
| | AIN1 | A1 | I | Analog | Analog-to-digital converter input 1. |
| | AIN2 | B3 | I | Analog | Analog-to-digital converter input 2. |
| | AIN3 | B2 | I | Analog | Analog-to-digital converter input 3. |
| | AIN4 | A2 | I | Analog | Analog-to-digital converter input 4. |
| | AIN5 | A3 | 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 | A12 | I | Analog | Analog-to-digital converter input 8. |
| | AIN9 | B11 | I | Analog | Analog-to-digital converter input 9. |
| | AIN10 | A6 | I | Analog | Analog-to-digital converter input 10. |
| | AIN11 | B7 | I | Analog | Analog-to-digital converter input 11. |
| | AIN12 | H1 | I | Analog | Analog-to-digital converter input 12. |
| | AIN13 | H2 | I | Analog | Analog-to-digital converter input 13. |
| | AIN14 | G2 | I | Analog | Analog-to-digital converter input 14. |
| | AIN15 | G1 | I | Analog | Analog-to-digital converter input 15. |
| | VREFA | A7 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 4095. The VREFA input is limited to the range specified in Table 19-22 on page 797. |

**Table 17-9. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| General-Purpose Timers | CCP0 | H1<br>L2<br>M2<br>L9<br>E12<br>C11<br>B7<br>A4 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP1 | M1<br>L1<br>L6<br>M8<br>D12<br>A12<br>A7<br>A2 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP2 | B2<br>G2<br>L1<br>L8<br>D12<br>B12<br>B11<br>B7<br>B4 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP3 | B2<br>M2<br>M1<br>M6<br>K3<br>H12<br>C11<br>A11<br>A4 | I/O | TTL | Capture/Compare/PWM 3. |
| | CCP4 | L2<br>L1<br>M6<br>K4<br>B11<br>B4 | I/O | TTL | Capture/Compare/PWM 4. |
| | CCP5 | B3<br>H2<br>L1<br>C10<br>M7<br>A7<br>B7 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | G1<br>H2<br>B12<br>C9<br>B7 | I/O | TTL | Capture/Compare/PWM 6. |
| | CCP7 | | I/O | TTL | Capture/Compare/PWM 7. |

**Table 17-9. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| | | G2<br>H1<br>A12<br>C8<br>A7 | | | |
| Hibernate | HIB | M12 | O | OD | An output that indicates the processor is in Hibernate mode. |
| | VBAT | L12 | - | 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 | M10 | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| | XOSC0 | K11 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the **HIBCTL** register. |
| | XOSC1 | K12 | O | Analog | Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| I2C | I2C0SCL | C11 | I/O | OD | $I^2C$ module 0 clock. |
| | I2C0SDA | C12 | I/O | OD | $I^2C$ module 0 data. |
| | I2C1SCL | K1<br>L3<br>L6 | I/O | OD | $I^2C$ module 1 clock. |
| | I2C1SDA | K2<br>M3<br>M6 | I/O | OD | $I^2C$ module 1 data. |
| JTAG/SWD/SWO | SWCLK | A9 | I | TTL | JTAG/SWD CLK. |
| | SWDIO | B9 | I/O | TTL | JTAG TMS and SWDIO. |
| | SWO | A10 | O | TTL | JTAG TDO and SWO. |
| | TCK | A9 | I | TTL | JTAG/SWD CLK. |
| | TDI | B8 | I | TTL | JTAG TDI. |
| | TDO | A10 | O | TTL | JTAG TDO and SWO. |
| | TMS | B9 | I | TTL | JTAG TMS and SWDIO. |

**Table 17-9. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| Power | GND | C4<br>H3<br>C5<br>J3<br>K6<br>K5<br>L10<br>K10<br>F10<br>J10<br>F11<br>B6<br>F12 | - | Power | Ground reference for logic and I/O pins. |
| | GNDA | A5<br>B5 | - | Power | The ground reference for the analog circuits (ADC, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| | LDO | E3 | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| | VDD | K7<br>G12<br>K8<br>K9<br>H10<br>G10<br>E10<br>G11 | - | Power | Positive supply for I/O and some logic. |
| | VDDA | C7<br>C6 | - | Power | The positive supply (3.3 V) 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 connected to 3.3 V, regardless of system implementation. |
| | VDDC | F3<br>D3<br>G3<br>C3 | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| SSI | SSI0Clk | M4 | I/O | TTL | SSI module 0 clock. |
| | SSI0Fss | L4 | I/O | TTL | SSI module 0 frame. |
| | SSI0Rx | L5 | I | TTL | SSI module 0 receive. |
| | SSI0Tx | M5 | O | TTL | SSI module 0 transmit. |
| | SSI1Clk | J11<br>A11 | I/O | TTL | SSI module 1 clock. |
| | SSI1Fss | J12<br>B12 | I/O | TTL | SSI module 1 frame. |
| | SSI1Rx | L9<br>B11 | I | TTL | SSI module 1 receive. |
| | SSI1Tx | L8<br>A12 | O | TTL | SSI module 1 transmit. |

**Table 17-9. Signals by Function, Except for GPIO *(continued)***

| Function | Pin Name | Pin Number | Pin Type | Buffer Type[a] | Description |
|---|---|---|---|---|---|
| System Control & Clocks | CMOD0 | E11 | I | TTL | CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved. |
| | CMOD1 | B10 | I | TTL | CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved. |
| | NMI | A8 | I | TTL | Non-maskable interrupt. |
| | OSC0 | L11 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| | OSC1 | M11 | O | Analog | Main oscillator crystal output. Leave unconnected when using a single-ended clock source. |
| | $\overline{\text{RST}}$ | H11 | I | TTL | System reset input. |
| UART | U0Rx | L3 | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U0Tx | M3 | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U1DSR | M9 | I | TTL | UART module 1 Data Set Ready modem output control line. |
| | U1Rx | G1<br>H2<br>M2<br>L3<br>E12<br>A6 | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U1Tx | G2<br>H1<br>L2<br>M3<br>D12<br>B7 | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U2Rx | G1<br>K1<br>A6<br>B4 | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U2Tx | B2<br>G2<br>K2<br>A3 | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 17-10. GPIO Pins and Alternate Functions**

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PA0 | L3 | - | U0Rx | - | - | - | - | - | - | I2C1SCL | U1Rx | - | - |
| PA1 | M3 | - | U0Tx | - | - | - | - | - | - | I2C1SDA | U1Tx | - | - |
| PA2 | M4 | - | SSI0Clk | - | - | - | - | - | - | - | - | - | - |
| PA3 | L4 | - | SSI0Fss | - | - | - | - | - | - | - | - | - | - |
| PA4 | L5 | - | SSI0Rx | - | - | - | - | - | - | - | - | - | - |
| PA5 | M5 | - | SSI0Tx | - | - | - | - | - | - | - | - | - | - |
| PA6 | L6 | - | I2C1SCL | CCP1 | - | - | - | - | - | - | - | - | - |
| PA7 | M6 | - | I2C1SDA | CCP4 | - | - | - | - | CCP3 | - | - | - | - |

**Table 17-10. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PB0 | E12 | - | CCP0 | - | - | - | U1Rx | - | - | - | - | - | - |
| PB1 | D12 | - | CCP2 | - | - | CCP1 | U1Tx | - | - | - | - | - | - |
| PB2 | C11 | - | I2C0SCL | - | - | CCP3 | CCP0 | - | - | - | - | - | - |
| PB3 | C12 | - | I2C0SDA | - | - | - | - | - | - | - | - | - | - |
| PB4 | A6 | AIN10 | - | - | - | U2Rx | - | - | U1Rx | - | - | - | - |
| PB5 | B7 | AIN11 | - | CCP5 | CCP6 | CCP0 | - | CCP2 | U1Tx | - | - | - | - |
| PB6 | A7 | VREFA | CCP1 | CCP7 | - | - | - | CCP5 | - | - | - | - | - |
| PB7 | A8 | - | - | - | - | NMI | - | - | - | - | - | - | - |
| PC0 | A9 | - | - | - | TCK SWCLK | - | - | - | - | - | - | - | - |
| PC1 | B9 | - | - | - | TMS SWDIO | - | - | - | - | - | - | - | - |
| PC2 | B8 | - | - | - | TDI | - | - | - | - | - | - | - | - |
| PC3 | A10 | - | - | - | TDO SWO | - | - | - | - | - | - | - | - |
| PC4 | L1 | - | CCP5 | - | - | - | CCP2 | CCP4 | - | - | CCP1 | - | - |
| PC5 | M1 | - | CCP1 | - | - | - | CCP3 | - | - | - | - | - | - |
| PC6 | M2 | - | CCP3 | - | - | - | U1Rx | CCP0 | - | - | - | - | - |
| PC7 | L2 | - | CCP4 | - | - | CCP0 | U1Tx | - | - | - | - | - | - |
| PD0 | G1 | AIN15 | - | - | - | U2Rx | U1Rx | CCP6 | - | - | - | - | - |
| PD1 | G2 | AIN14 | - | - | - | U2Tx | U1Tx | CCP7 | - | - | - | CCP2 | - |
| PD2 | H2 | AIN13 | U1Rx | CCP6 | - | CCP5 | - | - | - | - | - | - | - |
| PD3 | H1 | AIN12 | U1Tx | CCP7 | - | CCP0 | - | - | - | - | - | - | - |
| PD4 | A4 | AIN7 | CCP0 | CCP3 | - | - | - | - | - | - | - | - | - |
| PD5 | B4 | AIN6 | CCP2 | CCP4 | - | - | - | - | - | - | U2Rx | - | - |
| PD6 | A3 | AIN5 | - | - | - | - | - | - | - | - | U2Tx | - | - |
| PD7 | A2 | AIN4 | - | - | CCP1 | - | - | - | - | - | - | - | - |
| PE0 | A11 | - | - | SSI1Clk | CCP3 | - | - | - | - | - | - | - | - |
| PE1 | B12 | - | - | SSI1Fss | - | CCP2 | CCP6 | - | - | - | - | - | - |
| PE2 | B11 | AIN9 | CCP4 | SSI1Rx | - | - | CCP2 | - | - | - | - | - | - |
| PE3 | A12 | AIN8 | CCP1 | SSI1Tx | - | - | CCP7 | - | - | - | - | - | - |
| PE4 | B2 | AIN3 | CCP3 | - | - | - | U2Tx | CCP2 | - | - | - | - | - |
| PE5 | B3 | AIN2 | CCP5 | - | - | - | - | - | - | - | - | - | - |
| PE6 | A1 | AIN1 | - | - | - | - | - | - | - | - | - | - | - |
| PE7 | B1 | AIN0 | - | - | - | - | - | - | - | - | - | - | - |
| PF0 | M9 | - | - | - | - | - | - | - | - | - | U1DSR | - | - |
| PF1 | H12 | - | - | - | - | - | - | - | - | - | - | CCP3 | - |
| PF2 | J11 | - | - | - | - | - | - | - | - | - | SSI1Clk | - | - |
| PF3 | J12 | - | - | - | - | - | - | - | - | - | SSI1Fss | - | - |
| PF4 | L9 | - | CCP0 | - | - | - | - | - | - | - | SSI1Rx | - | - |
| PF5 | L8 | - | CCP2 | - | - | - | - | - | - | - | SSI1Tx | - | - |

**Table 17-10. GPIO Pins and Alternate Functions *(continued)***

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a] | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PF6 | M8 | - | CCP1 | - | - | - | - | - | - | - | - | - | - |
| PF7 | K4 | - | CCP4 | - | - | - | - | - | - | - | - | - | - |
| PG0 | K1 | - | U2Rx | - | I2C1SCL | - | - | - | - | - | - | - | - |
| PG1 | K2 | - | U2Tx | - | I2C1SDA | - | - | - | - | - | - | - | - |
| PG2 | J1 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG3 | J2 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG4 | K3 | - | CCP3 | - | - | - | - | - | - | - | - | - | - |
| PG5 | M7 | - | CCP5 | - | - | - | - | - | - | - | - | - | - |
| PG6 | L7 | - | - | - | - | - | - | - | - | - | - | - | - |
| PG7 | C10 | - | - | - | - | - | - | - | - | CCP5 | - | - | - |
| PH0 | C9 | - | CCP6 | - | - | - | - | - | - | - | - | - | - |
| PH1 | C8 | - | CCP7 | - | - | - | - | - | - | - | - | - | - |
| PH2 | D11 | - | - | - | - | - | - | - | - | - | - | - | - |
| PH3 | D10 | - | - | - | - | - | - | - | - | - | - | - | - |

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

**Table 17-11. Possible Pin Assignments for Alternate Functions**

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| one | AIN0 | PE7 |
| | AIN1 | PE6 |
| | AIN10 | PB4 |
| | AIN11 | PB5 |
| | AIN12 | PD3 |
| | AIN13 | PD2 |
| | AIN14 | PD1 |
| | AIN15 | PD0 |
| | AIN2 | PE5 |
| | AIN3 | PE4 |
| | AIN4 | PD7 |
| | AIN5 | PD6 |
| | AIN6 | PD5 |
| | AIN7 | PD4 |
| | AIN8 | PE3 |
| | AIN9 | PE2 |
| | I2C0SCL | PB2 |
| | I2C0SDA | PB3 |
| | NMI | PB7 |
| | SSI0Clk | PA2 |
| | SSI0Fss | PA3 |
| | SSI0Rx | PA4 |
| | SSI0Tx | PA5 |
| | SWCLK | PC0 |
| | SWDIO | PC1 |
| | SWO | PC3 |
| | TCK | PC0 |
| | TDI | PC2 |
| | TDO | PC3 |
| | TMS | PC1 |
| | U0Rx | PA0 |
| | U0Tx | PA1 |
| | U1DSR | PF0 |
| | VREFA | PB6 |
| two | SSI1Clk | PF2 PE0 |
| | SSI1Fss | PF3 PE1 |
| | SSI1Rx | PF4 PE2 |
| | SSI1Tx | PF5 PE3 |
| three | I2C1SCL | PG0 PA0 PA6 |
| | I2C1SDA | PG1 PA1 PA7 |

**Table 17-11. Possible Pin Assignments for Alternate Functions *(continued)***

| # of Possible Assignments | Alternate Function | GPIO Function |
|---|---|---|
| four | U2Rx | PD0 PG0 PB4 PD5 |
| | U2Tx | PE4 PD1 PG1 PD6 |
| five | CCP6 | PD0 PD2 PE1 PH0 PB5 |
| | CCP7 | PD1 PD3 PE3 PH1 PB6 |
| six | CCP4 | PC7 PC4 PA7 PF7 PE2 PD5 |
| | U1Rx | PD0 PD2 PC6 PA0 PB0 PB4 |
| | U1Tx | PD1 PD3 PC7 PA1 PB1 PB5 |
| seven | CCP5 | PE5 PD2 PC4 PG7 PG5 PB6 PB5 |
| eight | CCP0 | PD3 PC7 PC6 PF4 PB0 PB2 PB5 PD4 |
| | CCP1 | PC5 PC4 PA6 PF6 PB1 PE3 PB6 PD7 |
| nine | CCP2 | PE4 PD1 PC4 PF5 PB1 PE1 PE2 PB5 PD5 |
| | CCP3 | PE4 PC6 PC5 PA7 PG4 PF1 PB2 PE0 PD4 |

## 17.3 Connections for Unused Signals

Table 17-12 on page 782 show how to handle signals for functions that are not used in a particular system implementation for devices that are in a 100-pin LQFP 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 17-12. Connections for Unused Signals (100-Pin LQFP)**

| Function | Signal Name | Pin Number | Acceptable Practice | Preferred Practice |
|---|---|---|---|---|
| GPIO | All unused GPIOs | - | NC | GND |
| Hibernate | $\overline{\text{HIB}}$ | 51 | NC | NC |
| | VBAT | 55 | NC | GND |
| | $\overline{\text{WAKE}}$ | 50 | NC | GND |
| | XOSC0 | 52 | NC | GND |
| | XOSC1 | 53 | NC | NC |
| No Connects | NC | - | NC | NC |
| System Control | OSC0 | 48 | NC | GND |
| | OSC1 | 49 | NC | NC |
| | $\overline{\text{RST}}$ | 64 | Pull up as shown in Figure 5-1 on page 171 | Connect through a capacitor to GND as close to pin as possible |
| USB0RBIAS | 73 | Connect to GND through 10-kΩ resistor. | Connect to GND through 10-kΩ resistor. | |

Table 17-13 on page 783 show how to handle signals for functions that are not used in a particular system implementation for devices that are in a 108-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 17-13. Connections for Unused Signals (108-Ball BGA)**

| Function | Signal Name | Pin Number | Acceptable Practice | Preferred Practice |
|---|---|---|---|---|
| GPIO | All unused GPIOs | - | NC | GND |
| Hibernate | HIB | M12 | NC | NC |
| | VBAT | L12 | NC | GND |
| | WAKE | M10 | NC | GND |
| | XOSC0 | K11 | NC | GND |
| | XOSC1 | K12 | NC | NC |
| No Connects | NC | - | NC | NC |
| System Control | OSC0 | L11 | NC | GND |
| | OSC1 | M11 | NC | NC |
| | RST | H11 | Pull up as shown in Figure 5-1 on page 171 | Connect through a capacitor to GND as close to pin as possible |

# 18 Operating Characteristics

### Table 18-1. Temperature Characteristics

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Industrial operating temperature range | $T_A$ | -40 to +85 | °C |
| Unpowered storage temperature range | $T_S$ | -65 to +150 | °C |

### Table 18-2. Thermal Characteristics

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Thermal resistance (junction to ambient)[a] | $\Theta_{JA}$ | 33 (100LQFP)<br>31 (108BGA) | °C/W |
| Junction temperature, -40 to +125[b] | $T_J$ | $T_A + (P \bullet \Theta_{JA})$ | °C |

a. Junction to ambient thermal resistance $\theta_{JA}$ numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

### Table 18-3. ESD Absolute Maximum Ratings[a]

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| $V_{ESDHBM}$ | - | - | 2.0 | kV |
| $V_{ESDCDM}$ | - | - | 500 | V |

a. All Stellaris® parts are ESD tested following the JEDEC standard.

# 19 Electrical Characteristics

## 19.1 Maximum Ratings

The maximum ratings are the limits to which the device can be subjected without permanently damaging the device.

**Note:** The device is not guaranteed to operate properly at the maximum ratings.

**Table 19-1. 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 |
| $V_{IN}$ | Input voltage | -0.3 | 5.5 | V |
| | Input voltage for a GPIO configured as an analog input | -0.3 | $V_{DD}$ + 0.3 | V |
| | Input voltage for PB0 and PB1 when configured as GPIO | -0.3 | $V_{DD}$ + 0.3 | V |
| I | Maximum current per output pin | - | 25 | mA |
| $V_{NON}$ | Maximum input voltage on a non-power pin when the microcontroller is unpowered | - | 300 | mV |

a. Voltages are measured with respect to GND.

**Important:** This device contains circuitry to protect the inputs against damage due to high-static voltages or electric fields; 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 782).

## 19.2 Recommended Operating Conditions

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the $V_{OL}$ value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package or BGA pin group with the total number of high-current GPIO outputs not exceeding four for the entire package.

**Table 19-2. Recommended DC Operating Conditions**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{DD}$ | $V_{DD}$ supply voltage | 3.0 | 3.3 | 3.6 | V |
| $V_{DDA}$ | $V_{DDA}$ supply voltage | 3.0 | 3.3 | 3.6 | V |
| $V_{DDC}$ | $V_{DDC}$ supply voltage | 1.235 | 1.3 | 1.365 | V |
| $V_{IH}$ | High-level input voltage | 2.1 | - | 5.0 | V |
| $V_{IL}$ | Low-level input voltage | -0.3 | - | 1.2 | V |
| $V_{OH}$[a] | High-level output voltage | 2.4 | - | - | V |

**Table 19-2. Recommended DC Operating Conditions *(continued)***

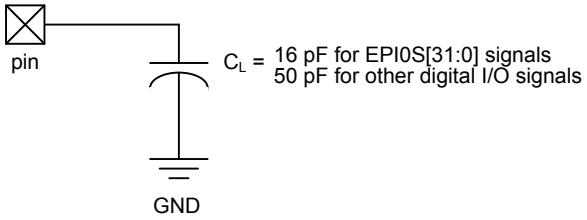
| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{OL}$[a] | Low-level output voltage | - | - | 0.4 | V |
| $I_{OH}$ | High-level source current, $V_{OH}$=2.4 V | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |
| | 4-mA Drive | 4.0 | - | - | mA |
| | 8-mA Drive | 8.0 | - | - | mA |
| $I_{OL}$ | Low-level sink current, $V_{OL}$=0.4 V | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |
| | 4-mA Drive | 4.0 | - | - | mA |
| | 8-mA Drive | 8.0 | - | - | mA |
| $I_{OHC}$ | High-current sink, $V_{OL}$=1.2 V | 18 | - | - | mA |

a. $V_{OL}$ and $V_{OH}$ shift to 1.2 V when using high-current GPIOs.

## 19.3 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements.

**Figure 19-1. Load Conditions**




## 19.4 JTAG and Boundary Scan

**Table 19-3. 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 | - | $t_{TCK}$ | - | ns |
| J4 | $T_{TCK_HIGH}$ | TCK clock High time | - | $t_{TCK}$ | - | 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 | 20 | - | - | ns |
| J8 | $T_{TMS_HLD}$ | TMS hold time from TCK rise | 20 | - | - | ns |
| J9 | $T_{TDI_SU}$ | TDI setup time to TCK rise | 25 | - | - | ns |
| J10 | $T_{TDI_HLD}$ | TDI hold time from TCK rise | 25 | - | - | ns |

**Table 19-3. JTAG Characteristics *(continued)***

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| J11 | $T_{TDO_ZDV}$ | TCK fall to Data Valid from High-Z, 2-mA drive | - | 23 | 35 | ns |
| | | TCK fall to Data Valid from High-Z, 4-mA drive | | 15 | 26 | ns |
| | | TCK fall to Data Valid from High-Z, 8-mA drive | | 14 | 25 | ns |
| | | TCK fall to Data Valid from High-Z, 8-mA drive with slew rate control | | 18 | 29 | ns |
| J12 | $T_{TDO_DV}$ | TCK fall to Data Valid from Data Valid, 2-mA drive | - | 21 | 35 | ns |
| | | TCK fall to Data Valid from Data Valid, 4-mA drive | | 14 | 25 | ns |
| | | TCK fall to Data Valid from Data Valid, 8-mA drive | | 13 | 24 | ns |
| | | TCK fall to Data Valid from Data Valid, 8-mA drive with slew rate control | | 18 | 28 | ns |
| J13 | $T_{TDO_DVZ}$ | TCK fall to High-Z from Data Valid, 2-mA drive | - | 9 | 11 | ns |
| | | TCK fall to High-Z from Data Valid, 4-mA drive | | 7 | 9 | ns |
| | | TCK fall to High-Z from Data Valid, 8-mA drive | | 6 | 8 | ns |
| | | TCK fall to High-Z from Data Valid, 8-mA drive with slew rate control | | 7 | 9 | ns |

a. A ratio of at least 8:1 must be kept between the system clock and TCK.

**Figure 19-2. JTAG Test Clock Input Timing**

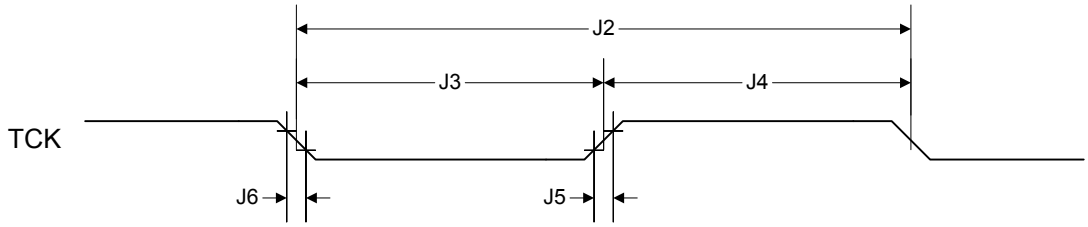
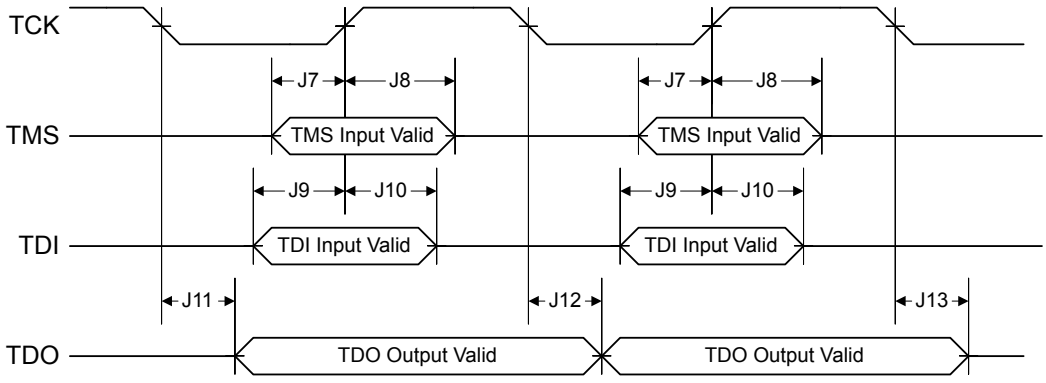



**Figure 19-3. JTAG Test Access Port (TAP) Timing**



## 19.5 Power and Brown-out

**Table 19-4. Power Characteristics**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| P1 | $V_{TH}$ | Power-On Reset threshold | - | 2 | - | V |
| P2 | $V_{BTH}$ | Brown-Out Reset threshold | 2.85 | 2.9 | 2.95 | V |
| P3 | $T_{POR}$ | Power-On Reset timeout | 6 | - | 18 | ms |
| P4 | $T_{BOR}$ | Brown-Out timeout | - | 500 | - | µs |
| P5 | $T_{IRPOR}$ | Internal reset timeout after POR | - | - | 2 | ms |
| P6 | $T_{IRBOR}$ | Internal reset timeout after BOR | - | - | 2 | ms |
| P7 | $T_{VDDRISE}$ | Supply voltage ($V_{DD}$) rise time (0V-3.0V) | - | - | 10 | ms |
| P8 | $T_{VDD2_3}$ | Supply voltage ($V_{DD}$) rise time (2.0V-3.0V) | - | - | 6 | ms |

**Figure 19-4. Power-On Reset Timing**

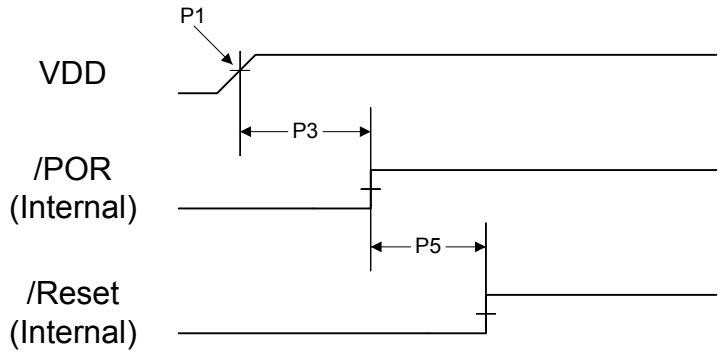



**Figure 19-5. Brown-Out Reset Timing**

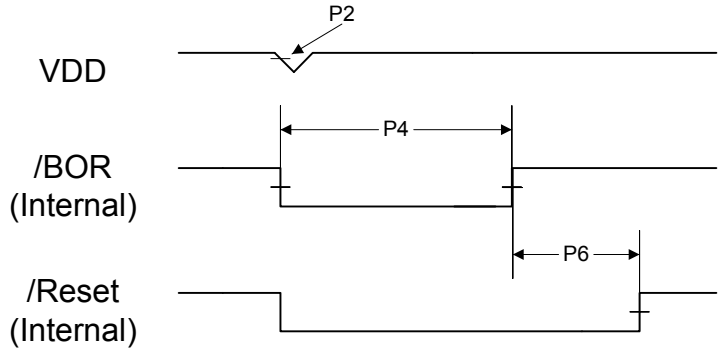
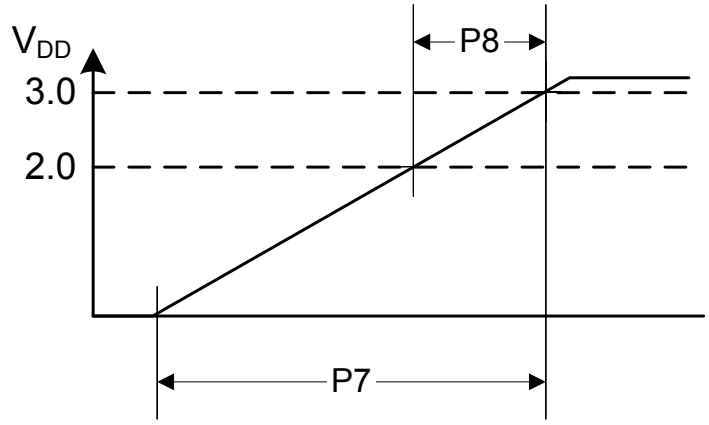


**Figure 19-6. Power-On Reset and Voltage Parameters**




## 19.6 Reset

**Table 19-5. Reset Characteristics**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| R1 | $T_{IRHWR}$ | Internal reset timeout after hardware reset ($\overline{RST}$ pin) | - | - | 2 | ms |
| R2 | $T_{IRSWR}$ | Internal reset timeout after software-initiated system reset | - | - | 2 | ms |
| R3 | $T_{IRWDR}$ | Internal reset timeout after watchdog reset | - | - | 2 | ms |
| R4 | $T_{IRMFR}$ | Internal reset timeout after MOSC failure reset | - | - | 2 | ms |
| R5 | $T_{MIN}$ | Minimum $\overline{RST}$ pulse width[a] | 2 | - | - | µs |

a. This specification must be met in order to guarantee proper reset operation.

**Figure 19-7. External Reset Timing ($\overline{RST}$)**

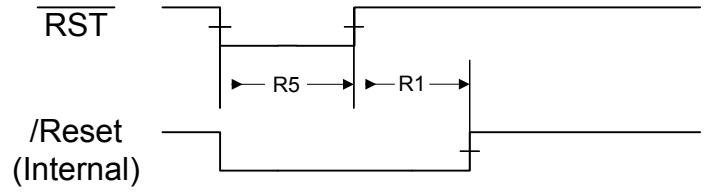



**Figure 19-8. Software Reset Timing**

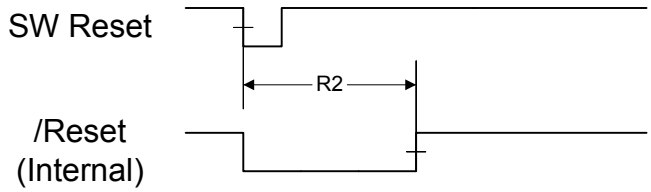


**Figure 19-9. Watchdog Reset Timing**

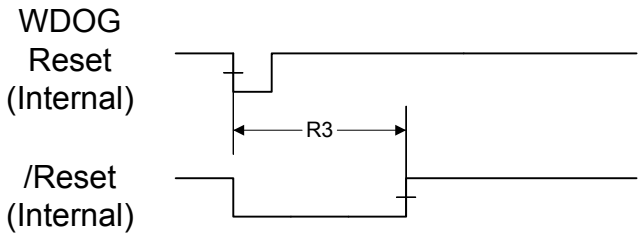
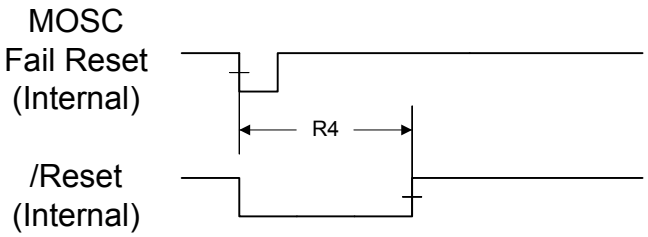



**Figure 19-10. MOSC Failure Reset Timing**




## 19.7 On-Chip Low Drop-Out (LDO) Regulator

**Table 19-6. LDO Regulator Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $C_{LDO}$ | External filter capacitor size for internal power supply | 1.0 | - | 3.0 | µF |
| $V_{LDO}$ | LDO output voltage | 1.235 | 1.3 | 1.365 | V |

## 19.8 Clocks

The following sections provide specifications on the various clock sources and mode.

### 19.8.1 PLL Specifications

The following tables provide specifications for using the PLL.

**Table 19-7. Phase Locked Loop (PLL) Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{REF_XTAL}$ | Crystal reference[a] | 3.579545 | - | 16.384 | MHz |
| $F_{REF_EXT}$ | External clock reference[a] | 3.579545 | - | 16.384 | MHz |
| $F_{PLL}$ | PLL frequency[b] | - | 400 | - | MHz |
| $T_{READY}$ | PLL lock time | 0.562[c] | - | 1.38[d] | ms |

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

c. Using a 16.384-MHz crystal

d. Using 3.5795-MHz crystal

Table 19-8 on page 791 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

**Table 19-8. Actual PLL Frequency**

| XTAL | Crystal Frequency (MHz) | PLL Frequency (MHz) | Error |
|---|---|---|---|
| 0x04 | 3.5795 | 400.904 | 0.0023% |
| 0x05 | 3.6864 | 398.1312 | 0.0047% |
| 0x06 | 4.0 | 400 | - |
| 0x07 | 4.096 | 401.408 | 0.0035% |
| 0x08 | 4.9152 | 398.1312 | 0.0047% |
| 0x09 | 5.0 | 400 | - |
| 0x0A | 5.12 | 399.36 | 0.0016% |
| 0x0B | 6.0 | 400 | - |
| 0x0C | 6.144 | 399.36 | 0.0016% |
| 0x0D | 7.3728 | 398.1312 | 0.0047% |
| 0x0E | 8.0 | 400 | - |
| 0x0F | 8.192 | 398.6773333 | 0.0033% |
| 0x10 | 10.0 | 400 | - |
| 0x11 | 12.0 | 400 | - |
| 0x12 | 12.288 | 401.408 | 0.0035% |
| 0x13 | 13.56 | 397.76 | 0.0056% |
| 0x14 | 14.318 | 400.90904 | 0.0023% |
| 0x15 | 16.0 | 400 | - |
| 0x16 | 16.384 | 404.1386667 | 0.010% |

## 19.8.2 PIOSC Specifications

**Table 19-9. PIOSC Clock Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{PIOSC25}$ | Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C | - | ±0.25% | ±1% | - |
| $F_{PIOSCT}$ | Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C, across specified temperature range | - | - | ±3% | - |
| $F_{PIOSCUCAL}$ | Internal 16-MHz precision oscillator frequency variance, user calibrated at a chosen temperature | - | ±0.25% | ±1% | - |

## 19.8.3 Internal 30-kHz Oscillator Specifications

**Table 19-10. 30-kHz Clock Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{IOSC30KHZ}$ | Internal 30-KHz oscillator frequency | 15 | 30 | 45 | KHz |

## 19.8.4 Hibernation Clock Source Specifications

**Table 19-11. Hibernation Clock Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{HIBOSC}$ | Hibernation module oscillator frequency | - | 4.194304 | - | MHz |
| $F_{HIBOSC_XTAL}$ | Crystal reference for hibernation oscillator | - | 4.194304 | - | MHz |

**Table 19-11. Hibernation Clock Characteristics *(continued)***

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $T_{HIBOSC_START}$ | Hibernation oscillator startup time[a] | - | - | 10 | ms |
| $F_{HIBOSC_EXT}$ | External clock reference for hibernation module | - | 32.768 | - | KHz |
| $DC_{HIBOSC_EXT}$ | External clock reference duty cycle | 45 | - | 55 | % |

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

**Table 19-12. HIB Oscillator Input Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{HIBOSC}$ | Hibernation module oscillator frequency | - | 4.194304 | - | MHz |
| $TOL_{HIBOSC}$ | Hibernation oscillator frequency tolerance | - | Defined by customer application requirements | - | PPM |

## 19.8.5 Main Oscillator Specifications

**Table 19-13. Main Oscillator Clock Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $F_{MOSC}$ | Main oscillator frequency | 1 | - | 16.384 | MHz |
| $T_{MOSC_PER}$ | Main oscillator period | 61 | - | 1000 | ns |
| $T_{MOSC_SETTLE}$ | Main oscillator settling time[a] | 17.5 | - | 20 | ms |
| $T_{REF_XTAL_BYPASS}$ | Crystal reference using the main oscillator (PLL in BYPASS mode)[b] | 1 | - | 16.384 | MHz |
| $F_{REF_EXT_BYPASS}$ | External clock reference (PLL in BYPASS mode)[b] | 0 | - | 50 | MHz |
| $DC_{MOSC_EXT}$ | External clock reference duty cycle | 45 | - | 55 | % |

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

b. If the ADC is used, the crystal reference must be 16 MHz ± .03% when the PLL is bypassed.

**Table 19-14. Supported MOSC Crystal Frequencies**

| Crystal Frequency (MHz) Not Using the PLL | Crystal Frequency (MHz) Using the PLL |
|---|---|
| 1.000 MHz | reserved |
| 1.8432 MHz | reserved |
| 2.000 MHz | reserved |
| 2.4576 MHz | reserved |
| 3.579545 MHz | |
| 3.6864 MHz | |
| 4 MHz | |
| 4.096 MHz | |
| 4.9152 MHz | |
| 5 MHz | |
| 5.12 MHz | |
| 6 MHz (reset value) | |

**Table 19-14. Supported MOSC Crystal Frequencies *(continued)***

| Crystal Frequency (MHz) Not Using the PLL | Crystal Frequency (MHz) Using the PLL |
|---|---|
| 6.144 MHz | |
| 7.3728 MHz | |
| 8 MHz | |
| 8.192 MHz | |
| 10.0 MHz | |
| 12.0 MHz | |
| 12.288 MHz | |
| 13.56 MHz | |
| 14.31818 MHz | |
| 16.0 MHz | |
| 16.384 MHz | |

### 19.8.6 System Clock Specification with ADC Operation

**Table 19-15. 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) | 15.9952 | 16 | 16.0048 | MHz |

## 19.9 Sleep Modes

**Table 19-16. Sleep Modes AC Characteristics**[a]

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| D1 | $T_{WAKE_S}$ | Time to wake from interrupt in sleep or deep-sleep mode, not using the PLL | - | - | 7 | system clocks |
| D2 | $T_{WAKE_PLL_S}$ | Time to wake from interrupt in sleep or deep-sleep mode when using the PLL | - | - | $T_{READY}$ | ms |
| D3 | $T_{ENTER_DS}$ | Time to enter deep-sleep mode from sleep request | - | 0 | 35[b] | ms |

a. Values in this table assume the IOSC is the clock source during sleep or deep-sleep mode.

b. Nominal specification occurs 99.9995% of the time.

## 19.10 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 263.

**Table 19-17. Hibernation Module Battery Characteristics**

| Parameter | Parameter Name | Min | Nominal | Max | Unit |
|---|---|---|---|---|---|
| $V_{BAT}$ | Battery supply voltage | 2.4 | 3.0 | 3.6 | V |
| $V_{LOWBAT}$ | Low battery detect voltage | 1.8 | - | 2.2 | V |

### Table 19-18. Hibernation Module AC Characteristics

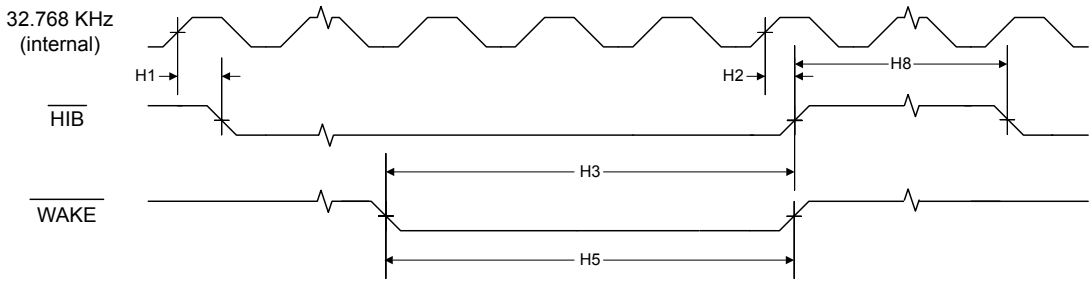
| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| H1 | $T_{HIB_LOW}$ | Internal 32.768 KHz clock reference rising edge to HIB asserted | 20 | - | - | µs |
| H2 | $T_{HIB_HIGH}$ | Internal 32.768 KHz clock reference rising edge to HIB deasserted | - | 30 | - | µs |
| H3 | $T_{WAKE_TO_HIB}$ | WAKE assert to HIB desassert (wake up time), internal Hibernation oscillator running during hibernation[a] | 62 | - | 124 | µs |
| H4 | $T_{WAKE_TO_HIB}$ | WAKE assert to HIB desassert (wake up time), internal Hibernation oscillator stopped during hibernation[a] | - | - | 10 | ms |
| H5 | $T_{WAKE_CLOCK}$ | WAKE assertion time, internal Hibernation oscillator running during hibernation | 62 | - | - | µs |
| H6 | $T_{WAKE_NOCLOCK}$ | WAKE assertion time, internal Hibernation oscillator stopped during hibernation[b] | 10 | - | - | ms |
| H7 | $T_{HIB_REG_ACCESS}$ | Time required for a write to a non-volatile register in the HIB module to complete | 92 | - | - | µs |
| H8 | $T_{HIB_TO_HIB}$ | HIB high time between assertions | 100 | - | - | ms |
| H9 | $T_{ENTER_HIB}$ | Time to enter hibernation mode from hibernation request | - | 0 | 35[c] | ms |

a. Code begins executing after the time period specified by $T_{IRPOR}$ following the deassertion of HIB.

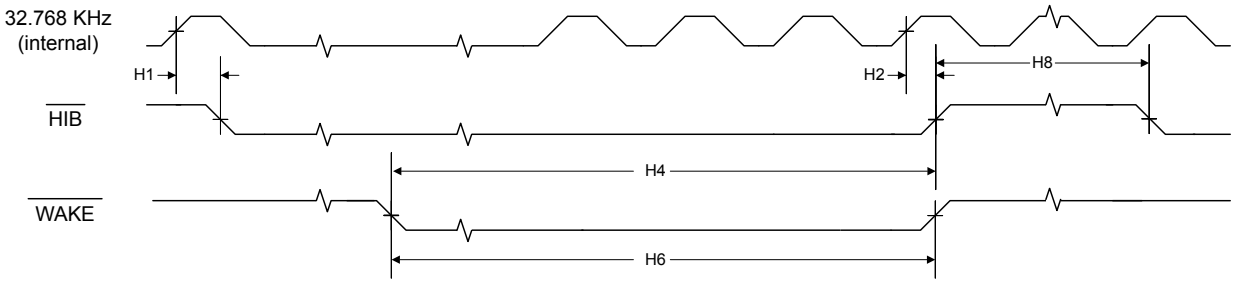
b. This mode is used when the PINWEN bit is set and the RTCEN bit is clear in the **HIBCTL** register.

c. Nominal specification occurs 99.998% of the time.

### Figure 19-11. Hibernation Module Timing with Internal Oscillator Running in Hibernation




### Figure 19-12. Hibernation Module Timing with Internal Oscillator Stopped in Hibernation



## 19.11 Flash Memory

**Table 19-19. Flash Memory Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $PE_{CYC}$ | Number of guaranteed program/erase cycles before failure[a] | 15,000 | - | - | cycles |
| $T_{RET}$ | Data retention, -40˚C to +85˚C | 10 | - | - | years |
| $T_{PROG}$ | Word program time | - | - | 1 | ms |
| $T_{BPROG}$ | Buffer program time | - | - | 1 | ms |
| $T_{ERASE}$ | Page erase time | - | - | 12 | ms |
| $T_{ME}$ | Mass erase time | - | - | 16 | ms |

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

## 19.12 GPIO Module

**Note:** All GPIOs are 5-V tolerant, except PB0 and PB1. See “Signal Description” on page 395 for more information on GPIO configuration.

**Table 19-20. GPIO Module Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $R_{GPIOPU}$ | GPIO internal pull-up resistor | 100 | - | 300 | kΩ |
| $R_{GPIOPD}$ | GPIO internal pull-down resistor | 200 | - | 500 | kΩ |
| $I_{LKG}$ | GPIO input leakage current[a] | - | - | 2 | µA |
| $T_{GPIOR}$ | GPIO Rise Time, 2-mA drive[b] | - | 14 | 20 | ns |
| | GPIO Rise Time, 4-mA drive[b] | | 7 | 10 | ns |
| | GPIO Rise Time, 8-mA drive[b] | | 4 | 5 | ns |
| | GPIO Rise Time, 8-mA drive with slew rate control[b] | | 6 | 8 | ns |
| $T_{GPIOF}$ | GPIO Fall Time, 2-mA drive[c] | - | 14 | 21 | ns |
| | GPIO Fall Time, 4-mA drive[c] | | 7 | 11 | ns |
| | GPIO Fall Time, 8-mA drive[c] | | 4 | 6 | ns |
| | GPIO Fall Time, 8-mA drive with slew rate control[c] | | 6 | 8 | ns |

a. The leakage current is measured with GND or VDD 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 pullup/pulldown resistor is disabled.

b. Time measured from 20% to 80% of $V_{DD}$.

c. Time measured from 80% to 20% of $V_{DD}$.

# 19.13 Analog-to-Digital Converter (ADC)

### Table 19-21. ADC Characteristics[a]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{ADCIN}$ | Maximum single-ended, full-scale analog input voltage, using internal reference | - | - | 3.0 | V |
| | Maximum single-ended, full-scale analog input voltage, using external reference | - | - | $V_{REFA}$ | V |
| | Minimum single-ended, full-scale analog input voltage | 0.0 | - | - | V |
| | Maximum differential, full-scale analog input voltage, using internal reference | - | - | 1.5 | V |
| | Maximum differential, full-scale analog input voltage, using external reference | - | - | $V_{REFA}/2$ | V |
| | Minimum differential, full-scale analog input voltage | 0.0 | - | - | V |
| N | Resolution | | 12 | | bits |
| $F_{ADC}$ | ADC internal clock frequency[b] | 15.9952 | 16 | 16.0048 | MHz |
| $T_{ADCCONV}$ | Conversion time[c] | | 1 | | µs |
| $F_{ADCCONV}$ | Conversion rate[c] | | 1000 | | k samples/s |
| $T_{ADCSAMP}$ | Sample time | 125 | - | - | ns |
| $T_{LT}$ | Latency from trigger to start of conversion | - | 2 | - | system clocks |
| $I_L$ | ADC input leakage | - | - | 2.0 | µA |
| $R_{ADC}$ | ADC equivalent resistance | - | - | 10 | kΩ |
| $C_{ADC}$ | ADC equivalent capacitance | 0.9 | 1.0 | 1.1 | pF |
| $E_L$ | Integral nonlinearity (INL) error, 12-bit mode | - | - | ±8 | LSB |
| | Integral nonlinearity (INL) error, 10-bit mode | - | - | ±2 | LSB |
| $E_D$ | Differential nonlinearity (DNL) error, 12-bit mode | - | - | ±4 | LSB |
| | Differential nonlinearity (DNL) error, 10-bit mode | - | - | ±2 | LSB |
| $E_O$ | Offset error, 12-bit mode | - | - | ±40 | LSB |
| | Offset error, 10-bit mode | - | - | ±10 | LSB |
| $E_G$ | Full-scale gain error, 12-bit mode | - | - | ±100 | LSB |
| | Full-scale gain error, 10-bit mode | - | - | ±25 | LSB |
| $E_{TS}$ | Temperature sensor accuracy[d] | - | - | ±5 | °C |

a. The ADC reference voltage is 3.0 V. This reference voltage is internally generated from the 3.3 VDDA supply by a band gap circuit.

b. The ADC must be clocked from the PLL or directly from an external clock source to operate properly.

c. The conversion time and rate scale from the specified number if the ADC internal clock frequency is any value other than 16 MHz.

d. Note that this parameter does not include ADC error.

**Figure 19-13. ADC Input Equivalency Diagram**

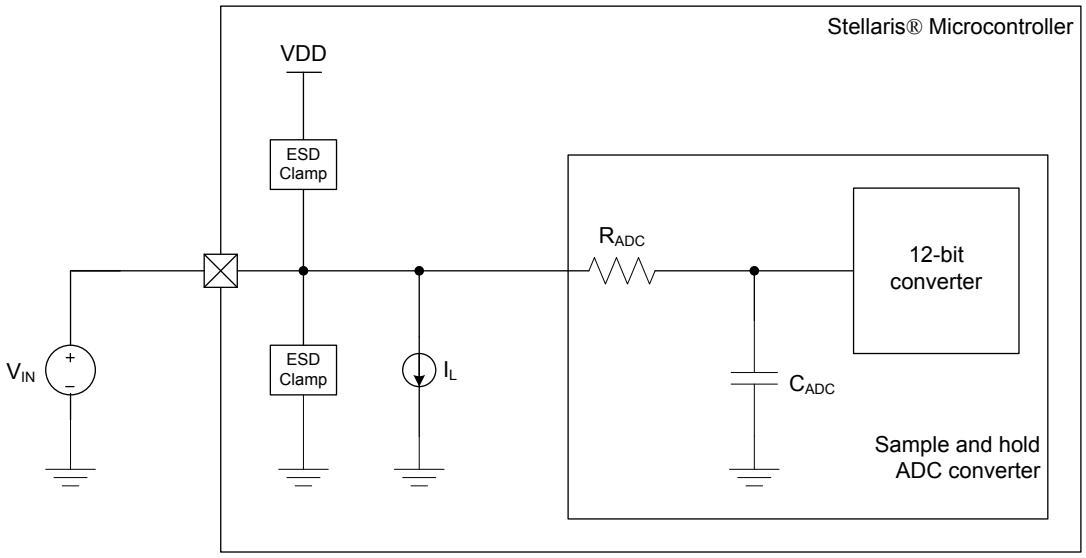



**Table 19-22. ADC Module External Reference Characteristics**[a]

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{REFA}$ | External voltage reference for ADC, when the VREF field in the **ADCCTL** register is 0x1[b] | 2.97 | - | 3.03 | V |
| | External voltage reference for ADC, when the VREF field in the **ADCCTL** register is 0x3[c] | 0.99 | - | 1.01 | V |
| $I_L$ | External voltage reference leakage current | - | - | 2.0 | µA |

a. Care must be taken to supply a reference voltage of acceptable quality.

b. Ground is always used as the reference level for the minimum conversion value.

c. Ground is always used as the reference level for the minimum conversion value.

**Table 19-23. ADC Module Internal Reference Characteristics**

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|
| $V_{REFI}$ | Internal voltage reference for ADC | - | 3.0 | - | V |

# 19.14 Synchronous Serial Interface (SSI)

**Table 19-24. SSI Characteristics**

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| S1 | $T_{CLK_PER}$ | SSIClk cycle time[a] | 100 | - | - | ns |
| S2 | $T_{CLK_HIGH}$ | SSIClk high time | - | 0.5 | - | t clk_per |
| S3 | $T_{CLK_LOW}$ | SSIClk low time | - | 0.5 | - | t clk_per |
| S4 | $T_{CLKRF}$ | SSIClk rise/fall time[b] | - | 4 | 6 | ns |
| S5 | $T_{DMD}$ | Data from master valid delay time | 0 | - | 1 | system clocks |
| S6 | $T_{DMS}$ | Data from master setup time | 1 | - | - | system clocks |
| S7 | $T_{DMH}$ | Data from master hold time | 2 | - | - | system clocks |
| S8 | $T_{DSS}$ | Data from slave setup time | 1 | - | - | system clocks |

**Table 19-24. SSI Characteristics *(continued)***

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| S9 | $T_{DSH}$ | Data from slave hold time | 2 | - | - | system clocks |

a. In master mode, the system clock must be at least twice as fast as the SSIClk; in slave mode, the system clock must be at least 12 times faster than the SSIClk.

b. Note that the delays shown are using 8-mA drive strength.

**Figure 19-14. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement**

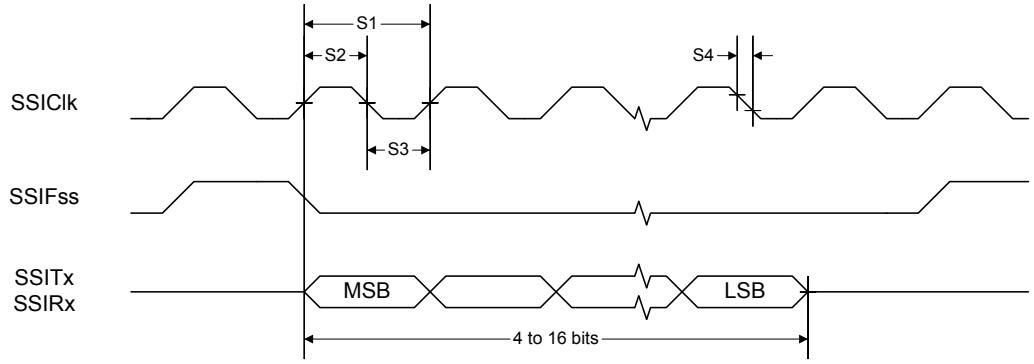



**Figure 19-15. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer**

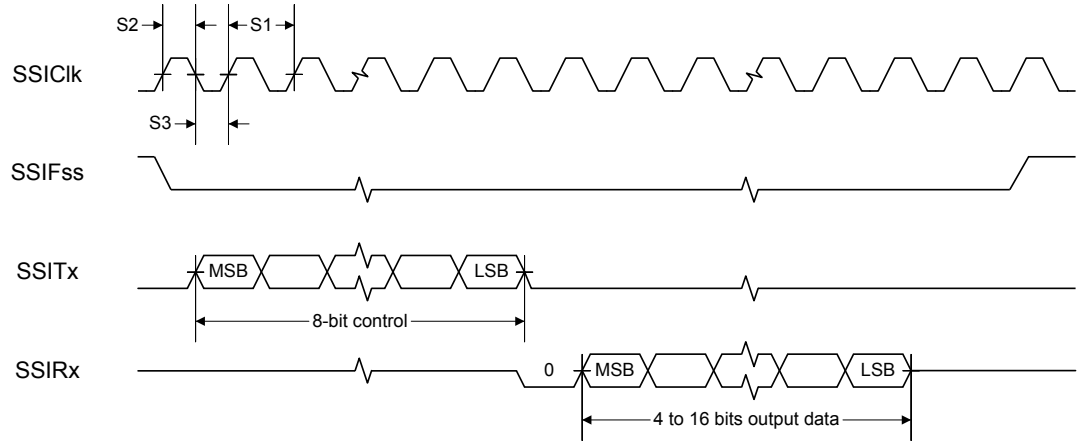
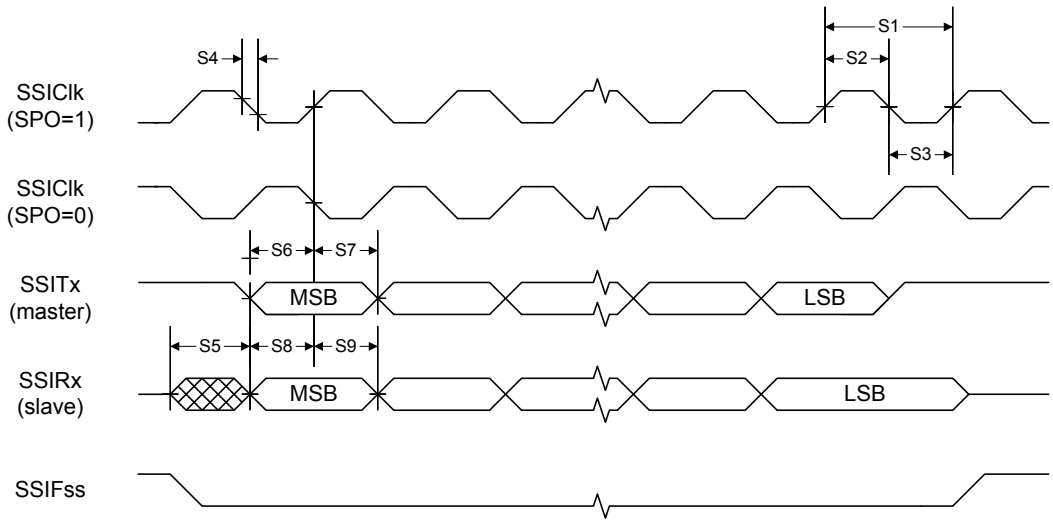


**Figure 19-16. SSI Timing for SPI Frame Format (FRF=00), with SPH=1**




## 19.15 Inter-Integrated Circuit ($I^2C$) Interface

**Table 19-25. $I^2C$ Characteristics**

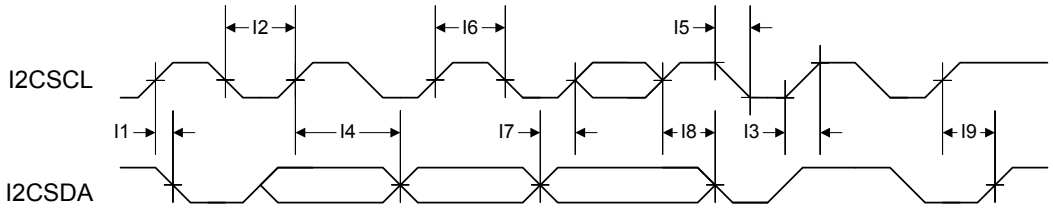
| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|---|---|
| I1[a] | $T_{SCH}$ | Start condition hold time | 36 | - | - | system clocks |
| I2[a] | $T_{LP}$ | Clock Low period | 36 | - | - | system clocks |
| I3[b] | $T_{SRT}$ | I2CSCL/I2CSDA rise time ($V_{IL}$ =0.5 V to $V_{IH}$ =2.4 V) | - | - | (see note b) | ns |
| I4[a] | $T_{DH}$ | Data hold time | 2 | - | - | 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[a] | $T_{DS}$ | 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 |

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 are open-drain-type outputs, 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 19-17. I²C Timing**




## 19.16 Ethernet Controller

**Table 19-26. 100BASE-TX Transmitter Characteristics[a]**

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| Peak output amplitude | 950 | - | 1050 | mVpk |
| Output amplitude symmetry | 98 | - | 102 | % |
| Output overshoot | - | - | 5 | % |
| Rise/Fall time | 3 | - | 5 | ns |
| Rise/Fall time imbalance | - | - | 500 | ps |
| Duty cycle distortion | - | - | ±250 | ps |
| Jitter | - | - | 1.4 | ns |

a. Measured at the line side of the transformer.

**Table 19-27. 100BASE-TX Transmitter Characteristics (informative)[a]**

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| Return loss | 16 | - | - | dB |
| Open-circuit inductance | 350 | - | - | µH |

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

**Table 19-28. 100BASE-TX Receiver Characteristics**

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| Signal detect assertion threshold | 600 | 700 | - | mVppd |
| Signal detect de-assertion threshold | 350 | 425 | - | mVppd |
| Differential input resistance | - | 3.6 | - | kΩ |
| Jitter tolerance (pk-pk) | 4 | - | - | ns |
| Baseline wander tracking | -80 | - | +80 | % |
| Signal detect assertion time | - | - | 1000 | µs |
| Signal detect de-assertion time | - | - | 4 | µs |

**Table 19-29. 10BASE-T Transmitter Characteristics[a]**

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| Peak differential output signal | 2.2 | - | 2.7 | V |
| Harmonic content | 27 | - | - | dB |
| Link pulse width | - | 100 | - | ns |
| Start-of-idle pulse width, Last bit 0 | - | 300 | - | ns |

**Table 19-29. 10BASE-T Transmitter Characteristics *(continued)***

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| Start-of-idle pulse width, Last bit 1 | - | 350 | - | ns |

a. The Manchester-encoded data pulses, the link pulse and the start-of-idle pulse are tested against the templates and using the procedures found in Clause 14 of *IEEE 802.3*.

**Table 19-30. 10BASE-T Transmitter Characteristics (informative)**[a]

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| Output return loss | 15 | - | - | dB |
| Output impedance balance | 29-17log(f/10) | - | - | dB |
| Peak common-mode output voltage | - | - | 50 | mV |
| Common-mode rejection | - | - | 100 | mV |
| Common-mode rejection jitter | - | - | 1 | ns |

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

**Table 19-31. 10BASE-T Receiver Characteristics**

| Parameter Name | Min | Nom | Max | Unit |
|---|---|---|---|---|
| Jitter tolerance (pk-pk) | 30 | 26 | - | ns |
| Input squelched threshold | 340 | 440 | 540 | mVppd |
| Differential input resistance | - | 3.6 | - | kΩ |
| Common-mode rejection | 25 | - | - | V |

**Table 19-32. Isolation Transformers**[a]

| Name | Value | Condition |
|---|---|---|
| Turns ratio | 1 CT : 1 CT | +/- 5% |
| Open-circuit inductance | 350 uH (min) | @ 10 mV, 10 kHz |
| Leakage inductance | 0.40 uH (max) | @ 1 MHz (min) |
| Inter-winding capacitance | 25 pF (max) | |
| DC resistance | 0.9 Ohm (max) | |
| Insertion loss | 0.4 dB (typ) | 0-65 MHz |
| HIPOT | 1500 | Vrms |

a. Two simple 1:1 isolation transformers are required at the line interface. Transformers with integrated common-mode chokes are recommended for exceeding FCC requirements. This table gives the recommended line transformer characteristics.

**Note:** The 100Base-TX amplitude specifications assume a transformer loss of 0.4 dB.

## 19.17 Current Consumption

This section provides information on typical and maximum power consumption under various conditions. Unless otherwise indicated, current consumption numbers include use of the on-chip LDO regulator and therefore include $I_{DDC}$.

### 19.17.1 Nominal Power Consumption

The following table provides nominal figures for current consumption.

**Table 19-33. Nominal Power Consumption**

| Parameter | Parameter Name | Conditions | Nom | Unit |
|---|---|---|---|---|
| $I_{DD_RUN}$ | Run mode 1 (Flash loop) | $V_{DD}$ = 3.3 V<br>Code= while(1){} executed out of Flash<br>Peripherals = All ON<br>System Clock = 80 MHz (with PLL)<br>Temp = 25°C | 90 | mA |
| $I_{DD_DEEPSLEEP}$ | Deep-sleep mode | Peripherals = All OFF<br>System Clock = IOSC30KHZ/64<br>Temp = 25°C | 550 | µA |
| $I_{HIB_NORTC}$ | Hibernate mode (external wake, RTC disabled, I/O not powered[a]) | $V_{BAT}$ = 3.0 V<br>$V_{DD}$ = 0 V<br>$V_{DDA}$ = 0 V<br>Peripherals = All OFF<br>System Clock = OFF<br>Hibernate Module = 0 kHz | 30 | µA |
| $I_{HIB_RTC}$ | Hibernate mode (RTC enabled, I/O not powered[a]) | $V_{BAT}$ = 3.0 V<br>$V_{DD}$ = 0 V<br>$V_{DDA}$ = 0 V<br>Peripherals = All OFF<br>System Clock = OFF<br>Hibernate Module = 32 kHz | 44 | µA |

a. The VDD3ON mode must be disabled for the I/O ring to be unpowered.

## 19.17.2 Maximum Current Consumption

The current measurements specified in the table that follows are maximum values under the following conditions:

- $V_{DD}$ = 3.6 V
- $V_{DDC}$ = 1.3 V
- $V_{BAT}$ = 3.25 V
- $V_{DDA}$ = 3.6 V
- Temperature = 25°C
- Clock source (MOSC) =3.579545-MHz crystal oscillator

**Table 19-34. Detailed Current Specifications**

| Parameter | Parameter Name | Conditions | Max | Unit |
|---|---|---|---|---|
| $I_{DD_RUN}$ | Run mode 1 (Flash loop) | $V_{DD}$ = 3.6 V<br>Code= while(1){} executed out of Flash<br>Peripherals = All ON<br>System Clock = 80 MHz (with PLL)<br>Temperature = 25°C | 135 | mA |

**Table 19-34. Detailed Current Specifications *(continued)***

| Parameter | Parameter Name | Conditions | Max | Unit |
|---|---|---|---|---|
| $I_{DD_SLEEP}$ | Sleep mode | $V_{DD}$ = 3.6 V<br>Peripherals = All Clock Gated<br>System Clock = 80 MHz (with PLL)<br>Temperature = 25°C | 46 | mA |
| $I_{DD_DEEPSLEEP}$ | Deep-Sleep mode | $V_{DD}$ = 3.6 V<br>Peripherals = All Clock Gated<br>System Clock = IOSC30/64<br>Temperature = 25°C | 1.6 | mA |

**Table 19-35. Hibernation Detailed Current Specifications**

| Parameter | Parameter Name | Conditions | Max | Unit |
|---|---|---|---|---|
| $I_{HIB_NORTC}$ | Hibernate mode (external wake, RTC disabled, I/O not powered[a]) | $V_{BAT}$ = 3.25 V<br>$V_{DD}$ = 0 V<br>$V_{DDA}$ = 0 V<br>Peripherals = All OFF<br>System Clock = OFF<br>Hibernate Module = 0 kHz<br>Temperature = 25°C | 118 | µA |
| $I_{HIB_RTC}$ | Hibernate mode (RTC enabled, I/O not powered[a]) | $V_{BAT}$ = 3.25 V<br>$V_{DD}$ = 0 V<br>$V_{DDA}$ = 0 V<br>Peripherals = All OFF<br>System Clock = OFF<br>Hibernate Module = 32.768 kHz<br>Temperature = 25°C | 141 | µA |

a. The VDD3ON mode must be disabled for the I/O ring to be unpowered.

**Table 19-36. External $V_{DDC}$ Source Current Specifications**

| Parameter | Parameter Name | Conditions | Max | Unit |
|---|---|---|---|---|
| $I_{DDC_RUN}$ | Run mode 1 (Flash loop), $V_{DDC}$ current | $V_{DD}$ = 3.6 V<br>$V_{DDC}$ = 1.3 V<br>Code= while(1){} executed out of Flash<br>Peripherals = All ON<br>System Clock = 80 MHz (with PLL)<br>Temperature = 25°C | 98 | mA |

# A Register Quick Reference

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **The Cortex-M3 Processor** | | | | | | | | | | | | | | | |
| **R0, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R1, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R2, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R3, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R4, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R5, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R6, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R7, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R8, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R9, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R10, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R11, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **R12, type R/W, , reset -** (see page 55) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **SP, type R/W, , reset -** (see page 56) | | | | | | | | | | | | | | | |
| SP | | | | | | | | | | | | | | | |
| SP | | | | | | | | | | | | | | | |
| **LR, type R/W, , reset 0xFFFF.FFFF** (see page 57) | | | | | | | | | | | | | | | |
| LINK | | | | | | | | | | | | | | | |
| LINK | | | | | | | | | | | | | | | |
| **PC, type R/W, , reset -** (see page 58) | | | | | | | | | | | | | | | |
| PC | | | | | | | | | | | | | | | |
| PC | | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **PSR, type R/W, , reset 0x0100.0000** (see page 59) | | | | | | | | | | | | | | | |
| N | Z | C | V | Q | ICI / IT | | THUMB | | | | | | | | |
| ICI / IT | | | | | | | | | ISRNUM | | | | | | |
| **PRIMASK, type R/W, , reset 0x0000.0000** (see page 63) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PRIMASK |
| **FAULTMASK, type R/W, , reset 0x0000.0000** (see page 64) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | FAULTMASK |
| **BASEPRI, type R/W, , reset 0x0000.0000** (see page 65) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | BASEPRI | | | | | | | |
| **CONTROL, type R/W, , reset 0x0000.0000** (see page 66) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | ASP | TMPL |
| **Cortex-M3 Peripherals**<br>**System Timer (SysTick) Registers**<br>Base 0xE000.E000 | | | | | | | | | | | | | | | |
| **STCTRL, type R/W, offset 0x010, reset 0x0000.0004** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | COUNT |
| | | | | | | | | | | | | | CLK_SRC | INTEN | ENABLE |
| **STRELOAD, type R/W, offset 0x014, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | RELOAD | | | | | | | |
| RELOAD | | | | | | | | | | | | | | | |
| **STCURRENT, type R/WC, offset 0x018, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | CURRENT | | | | | | | |
| CURRENT | | | | | | | | | | | | | | | |
| **Cortex-M3 Peripherals**<br>**Nested Vectored Interrupt Controller (NVIC) Registers**<br>Base 0xE000.E000 | | | | | | | | | | | | | | | |
| **EN0, type R/W, offset 0x100, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **EN1, type R/W, offset 0x104, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | INT | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **DIS0, type R/W, offset 0x180, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **DIS1, type R/W, offset 0x184, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | INT | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **PEND0, type R/W, offset 0x200, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **PEND1, type R/W, offset 0x204, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | INT | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **UNPEND0, type R/W, offset 0x280, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **UNPEND1, type R/W, offset 0x284, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | INT | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **ACTIVE0, type RO, offset 0x300, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **ACTIVE1, type RO, offset 0x304, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | INT | | | | | | |
| INT | | | | | | | | | | | | | | | |
| **PRI0, type R/W, offset 0x400, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI1, type R/W, offset 0x404, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI2, type R/W, offset 0x408, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI3, type R/W, offset 0x40C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI4, type R/W, offset 0x410, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI5, type R/W, offset 0x414, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI6, type R/W, offset 0x418, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI7, type R/W, offset 0x41C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI8, type R/W, offset 0x420, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI9, type R/W, offset 0x424, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI10, type R/W, offset 0x428, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI11, type R/W, offset 0x42C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI12, type R/W, offset 0x430, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |
| **PRI13, type R/W, offset 0x434, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| INTD | | | | | | | | INTC | | | | | | | |
| INTB | | | | | | | | INTA | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **SWTRIG, type WO, offset 0xF00, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | INTID | | | | | |
| **Cortex-M3 Peripherals**<br>**System Control Block (SCB) Registers**<br>Base 0xE000.E000 | | | | | | | | | | | | | | | |
| **ACTLR, type R/W, offset 0x008, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | DISFOLD | DISWBUF | DISMCYC |
| **CPUID, type RO, offset 0xD00, reset 0x412F.C230** | | | | | | | | | | | | | | | |
| IMP | | | | | | | | VAR | | | | CON | | | |
| PARTNO | | | | | | | | | | | | REV | | | |
| **INTCTRL, type R/W, offset 0xD04, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| NMISET | | | PENDSV | UNPENDSV | PENDSTSET | PENDSTCLR | | ISRPRE | ISRPEND | | | | VECPEND | | |
| VECPEND | | | | RETBASE | | | | | VECACT | | | | | | |
| **VTABLE, type R/W, offset 0xD08, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | BASE | OFFSET | | | | | | | | | | | | |
| OFFSET | | | | | | | | | | | | | | | |
| **APINT, type R/W, offset 0xD0C, reset 0xFA05.0000** | | | | | | | | | | | | | | | |
| VECTKEY | | | | | | | | | | | | | | | |
| ENDIANESS | | | | | PRIGROUP | | | | | | | | SYSRESREQ | VECTCLRACT | VECTRESET |
| **SYSCTRL, type R/W, offset 0xD10, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | SEVONPEND | | SLEEPDEEP | SLEEPEXIT | |
| **CFGCTRL, type R/W, offset 0xD14, reset 0x0000.0200** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | STKALIGN | BFHFNMIGN | | | | DIV0 | UNALIGNED | | MAINPEND | BASETHR |
| **SYSPRI1, type R/W, offset 0xD18, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | USAGE | | | | | | | |
| BUS | | | | | | | | MEM | | | | | | | |
| **SYSPRI2, type R/W, offset 0xD1C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| SVC | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| **SYSPRI3, type R/W, offset 0xD20, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| TICK | | | | | | | | PENDSV | | | | | | | |
| | | | | | | | | DEBUG | | | | | | | |
| **SYSHNDCTRL, type R/W, offset 0xD24, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | USAGE | BUS | MEM |
| SVC | BUSP | MEMP | USAGEP | TICK | PNDSV | | MON | SVCA | | | | USGA | | BUSA | MEMA |
| **FAULTSTAT, type R/W1C, offset 0xD28, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | DIV0 | UNALIGN | | | | | NOCP | INVPC | INVSTAT | UNDEF |
| BFARV | | | BSTKE | BUSTKE | IMPRE | PRECISE | IBUS | MMARV | | | MSTKE | MUSTKE | | DERR | IERR |
| **HFAULTSTAT, type R/W1C, offset 0xD2C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| DBG | FORCED | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | VECT | |
| **MMADDR, type R/W, offset 0xD34, reset -** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| **FAULTADDR, type R/W, offset 0xD38, reset -** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Cortex-M3 Peripherals**<br>**Memory Protection Unit (MPU) Registers**<br>Base 0xE000.E000 | | | | | | | | | | | | | | | |
| **MPUTYPE, type RO, offset 0xD90, reset 0x0000.0800** | | | | | | | | | | | | | | | |
| | | | | | | | | IREGION | | | | | | | |
| DREGION | | | | | | | | | | | | | | | SEPARATE |
| **MPUCTRL, type R/W, offset 0xD94, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | PRIVDEFEN | HFNMIENA | ENABLE |
| **MPUNUMBER, type R/W, offset 0xD98, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | NUMBER | | |
| **MPUBASE, type R/W, offset 0xD9C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | VALID | | REGION | | |
| **MPUBASE1, type R/W, offset 0xDA4, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | VALID | | REGION | | |
| **MPUBASE2, type R/W, offset 0xDAC, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | VALID | | REGION | | |
| **MPUBASE3, type R/W, offset 0xDB4, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | VALID | | REGION | | |
| **MPUATTR, type R/W, offset 0xDA0, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | XN | | AP | | | | | TEX | | | S | C | B |
| SRD | | | | | | | | | | SIZE | | | | | ENABLE |
| **MPUATTR1, type R/W, offset 0xDA8, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | XN | | AP | | | | | TEX | | | S | C | B |
| SRD | | | | | | | | | | SIZE | | | | | ENABLE |
| **MPUATTR2, type R/W, offset 0xDB0, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | XN | | AP | | | | | TEX | | | S | C | B |
| SRD | | | | | | | | | | SIZE | | | | | ENABLE |
| **MPUATTR3, type R/W, offset 0xDB8, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | XN | | AP | | | | | TEX | | | S | C | B |
| SRD | | | | | | | | | | SIZE | | | | | ENABLE |
| **System Control**<br>Base 0x400F.E000 | | | | | | | | | | | | | | | |
| **DID0, type RO, offset 0x000, reset -** (see page 187) | | | | | | | | | | | | | | | |
| | VER | | | | | | | CLASS | | | | | | | |
| MAJOR | | | | | | | | MINOR | | | | | | | |
| **PBORCTL, type R/W, offset 0x030, reset 0x0000.0002** (see page 189) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | BORIOR | |
| **RIS, type RO, offset 0x050, reset 0x0000.0000** (see page 190) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | MOSCPUPRIS | | PLLLRIS | | | | | BORRIS | |
| **IMC, type R/W, offset 0x054, reset 0x0000.0000** (see page 192) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | MOSCPUPIM | | PLLLIM | | | | | BORIM | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **MISC, type R/W1C, offset 0x058, reset 0x0000.0000** (see page 194) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | MOSCPUPMIS | | PLLLMIS | | | | | BORMIS | |
| **RESC, type R/W, offset 0x05C, reset -** (see page 196) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | MOSCFAIL |
| | | | | | | | | | | WDT1 | SW | WDT0 | BOR | POR | EXT |
| **RCC, type R/W, offset 0x060, reset 0x0780.3AD1** (see page 198) | | | | | | | | | | | | | | | |
| | | | | ACG | SYSDIV | | | | USESYSDIV | | | | | | |
| | | PWRDN | | BYPASS | XTAL | | | | | OSCSRC | | | | IOSCDIS | MOSCDIS |
| **PLLCFG, type RO, offset 0x064, reset -** (see page 202) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | F | | | | | | | | | | R | | | |
| **GPIOHBCTL, type R/W, offset 0x06C, reset 0x0000.0000** (see page 203) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PORTH | PORTG | PORTF | PORTE | PORTD | PORTC | PORTB | PORTA |
| **RCC2, type R/W, offset 0x070, reset 0x07C0.6810** (see page 205) | | | | | | | | | | | | | | | |
| USERCC2 | DIV400 | | | SYSDIV2 | | | | | SYSDIV2LSB | | | | | | |
| | | PWRDN2 | | BYPASS2 | | | | | OSCSRC2 | | | | | | |
| **MOSCCTL, type R/W, offset 0x07C, reset 0x0000.0000** (see page 208) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | CVAL |
| **DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000** (see page 209) | | | | | | | | | | | | | | | |
| | | | | DSDIVORIDE | | | | | | | | | | | |
| | | | | | | | | | DSOSCSRC | | | | | | |
| **PIOSCCAL, type R/W, offset 0x150, reset 0x0000.0000** (see page 211) | | | | | | | | | | | | | | | |
| UTEN | | | | | | | | | | | | | | | |
| | | | | | | CAL | UPDATE | | UT | | | | | | |
| **PIOSCSTAT, type RO, offset 0x154, reset 0x0000.0040** (see page 213) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DT | | | |
| | | | | | | RESULT | | | | | | CT | | | |
| **DID1, type RO, offset 0x004, reset -** (see page 214) | | | | | | | | | | | | | | | |
| VER | | | | FAM | | | | PARTNO | | | | | | | |
| PINCOUNT | | | | | | | | TEMP | | | PKG | | ROHS | QUAL | |
| **DC0, type RO, offset 0x008, reset 0x00FF.00BF** (see page 216) | | | | | | | | | | | | | | | |
| SRAMSZ | | | | | | | | | | | | | | | |
| FLASHSZ | | | | | | | | | | | | | | | |
| **DC1, type RO, offset 0x010, reset -** (see page 217) | | | | | | | | | | | | | | | |
| | | | WDT1 | | | | | | | | | | | ADC1 | ADC0 |
| MINSYSDIV | | | | MAXADC1SPD | | MAXADC0SPD | | MPU | HIB | TEMPSNS | PLL | WDT0 | SWO | SWD | JTAG |
| **DC2, type RO, offset 0x014, reset 0x000F.5037** (see page 219) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| | I2C1 | | I2C0 | | | | | | | SSI1 | SSI0 | | UART2 | UART1 | UART0 |
| **DC3, type RO, offset 0x018, reset 0xBFFF.0000** (see page 221) | | | | | | | | | | | | | | | |
| 32KHZ | | CCP5 | CCP4 | CCP3 | CCP2 | CCP1 | CCP0 | ADC0AIN7 | ADC0AIN6 | ADC0AIN5 | ADC0AIN4 | ADC0AIN3 | ADC0AIN2 | ADC0AIN1 | ADC0AIN0 |
| | | | | | | | | | | | | | | | |
| **DC4, type RO, offset 0x01C, reset 0x0004.F0FF** (see page 223) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | PICAL | | |
| CCP7 | CCP6 | UDMA | ROM | | | | | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| **DC5, type RO, offset 0x020, reset 0x0000.0000** (see page 225) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **DC6, type RO, offset 0x024, reset 0x0000.0000** (see page 226) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| **DC7, type RO, offset 0x028, reset 0xFFFF.FFFF** (see page 227) | | | | | | | | | | | | | | | |
| | DMACH30 | DMACH29 | DMACH28 | DMACH27 | DMACH26 | DMACH25 | DMACH24 | DMACH23 | DMACH22 | DMACH21 | DMACH20 | DMACH19 | DMACH18 | DMACH17 | DMACH16 |
| DMACH15 | DMACH14 | DMACH13 | DMACH12 | DMACH11 | DMACH10 | DMACH9 | DMACH8 | DMACH7 | DMACH6 | DMACH5 | DMACH4 | DMACH3 | DMACH2 | DMACH1 | DMACH0 |
| **DC8, type RO, offset 0x02C, reset 0xFFFF.FFFF** (see page 231) | | | | | | | | | | | | | | | |
| ADC1AIN15 | ADC1AIN14 | ADC1AIN13 | ADC1AIN12 | ADC1AIN11 | ADC1AIN10 | ADC1AIN9 | ADC1AIN8 | ADC1AIN7 | ADC1AIN6 | ADC1AIN5 | ADC1AIN4 | ADC1AIN3 | ADC1AIN2 | ADC1AIN1 | ADC1AIN0 |
| ADC0AIN15 | ADC0AIN14 | ADC0AIN13 | ADC0AIN12 | ADC0AIN11 | ADC0AIN10 | ADC0AIN9 | ADC0AIN8 | ADC0AIN7 | ADC0AIN6 | ADC0AIN5 | ADC0AIN4 | ADC0AIN3 | ADC0AIN2 | ADC0AIN1 | ADC0AIN0 |
| **DC9, type RO, offset 0x190, reset 0x00FF.00FF** (see page 234) | | | | | | | | | | | | | | | |
| | | | | | | | | ADC1DC7 | ADC1DC6 | ADC1DC5 | ADC1DC4 | ADC1DC3 | ADC1DC2 | ADC1DC1 | ADC1DC0 |
| | | | | | | | | ADC0DC7 | ADC0DC6 | ADC0DC5 | ADC0DC4 | ADC0DC3 | ADC0DC2 | ADC0DC1 | ADC0DC0 |
| **NVMSTAT, type RO, offset 0x1A0, reset 0x0000.0001** (see page 236) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | FWB |
| **RCGC0, type R/W, offset 0x100, reset 0x00000040** (see page 237) | | | | | | | | | | | | | | | |
| | | | WDT1 | | | | | | | | | | | ADC1 | ADC0 |
| | | | | MAXADC1SPD | | MAXADC0SPD | | | HIB | | | WDT0 | | | |
| **SCGC0, type R/W, offset 0x110, reset 0x00000040** (see page 239) | | | | | | | | | | | | | | | |
| | | | WDT1 | | | | | | | | | | | ADC1 | ADC0 |
| | | | | MAXADC1SPD | | MAXADC0SPD | | | HIB | | | WDT0 | | | |
| **DCGC0, type R/W, offset 0x120, reset 0x00000040** (see page 241) | | | | | | | | | | | | | | | |
| | | | WDT1 | | | | | | | | | | | ADC1 | ADC0 |
| | | | | | | | | | HIB | | | WDT0 | | | |
| **RCGC1, type R/W, offset 0x104, reset 0x00000000** (see page 243) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| | I2C1 | | I2C0 | | | | | | | SSI1 | SSI0 | | UART2 | UART1 | UART0 |
| **SCGC1, type R/W, offset 0x114, reset 0x00000000** (see page 245) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| | I2C1 | | I2C0 | | | | | | | SSI1 | SSI0 | | UART2 | UART1 | UART0 |
| **DCGC1, type R/W, offset 0x124, reset 0x00000000** (see page 248) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| | I2C1 | | I2C0 | | | | | | | SSI1 | SSI0 | | UART2 | UART1 | UART0 |
| **RCGC2, type R/W, offset 0x108, reset 0x00000000** (see page 251) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | UDMA | | | | | | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| **SCGC2, type R/W, offset 0x118, reset 0x00000000** (see page 253) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | UDMA | | | | | | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| **DCGC2, type R/W, offset 0x128, reset 0x00000000** (see page 255) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | UDMA | | | | | | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| **SRCR0, type R/W, offset 0x040, reset 0x00000000** (see page 257) | | | | | | | | | | | | | | | |
| | | | WDT1 | | | | | | | | | | | ADC1 | ADC0 |
| | | | | | | | | | HIB | | | WDT0 | | | |
| **SRCR1, type R/W, offset 0x044, reset 0x00000000** (see page 259) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 |
| | I2C1 | | I2C0 | | | | | | | SSI1 | SSI0 | | UART2 | UART1 | UART0 |
| **SRCR2, type R/W, offset 0x048, reset 0x00000000** (see page 261) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | UDMA | | | | | | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Hibernation Module**<br>Base 0x400F.C000 | | | | | | | | | | | | | | | |
| **HIBRTCC, type RO, offset 0x000, reset 0x0000.0000** (see page 274) | | | | | | | | | | | | | | | |
| RTCC | | | | | | | | | | | | | | | |
| RTCC | | | | | | | | | | | | | | | |
| **HIBRTCM0, type R/W, offset 0x004, reset 0xFFFF.FFFF** (see page 275) | | | | | | | | | | | | | | | |
| RTCM0 | | | | | | | | | | | | | | | |
| RTCM0 | | | | | | | | | | | | | | | |
| **HIBRTCM1, type R/W, offset 0x008, reset 0xFFFF.FFFF** (see page 276) | | | | | | | | | | | | | | | |
| RTCM1 | | | | | | | | | | | | | | | |
| RTCM1 | | | | | | | | | | | | | | | |
| **HIBRTCLD, type R/W, offset 0x00C, reset 0xFFFF.FFFF** (see page 277) | | | | | | | | | | | | | | | |
| RTCLD | | | | | | | | | | | | | | | |
| RTCLD | | | | | | | | | | | | | | | |
| **HIBCTL, type R/W, offset 0x010, reset 0x8000.0000** (see page 278) | | | | | | | | | | | | | | | |
| WRC | | | | | | | | | | | | | | | |
| | | | | | | | VDD3ON | VABORT | CLK32EN | LOWBATEN | PINWEN | RTCWEN | CLKSEL | HIBREQ | RTCEN |
| **HIBIM, type R/W, offset 0x014, reset 0x0000.0000** (see page 281) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCALT1 | RTCALT0 |
| **HIBRIS, type RO, offset 0x018, reset 0x0000.0000** (see page 283) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCALT1 | RTCALT0 |
| **HIBMIS, type RO, offset 0x01C, reset 0x0000.0000** (see page 285) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCALT1 | RTCALT0 |
| **HIBIC, type R/W1C, offset 0x020, reset 0x0000.0000** (see page 287) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCALT1 | RTCALT0 |
| **HIBRTCT, type R/W, offset 0x024, reset 0x0000.7FFF** (see page 288) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| TRIM | | | | | | | | | | | | | | | |
| **HIBDATA, type R/W, offset 0x030-0x12C, reset -** (see page 289) | | | | | | | | | | | | | | | |
| RTD | | | | | | | | | | | | | | | |
| RTD | | | | | | | | | | | | | | | |
| **Internal Memory**<br>**Flash Memory Registers (Flash Control Offset)**<br>Base 0x400F.D000 | | | | | | | | | | | | | | | |
| **FMA, type R/W, offset 0x000, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | OFFSET | | |
| OFFSET | | | | | | | | | | | | | | | |
| **FMD, type R/W, offset 0x004, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **FMC, type R/W, offset 0x008, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| WRKEY | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMT | MERASE | ERASE | WRITE |
| **FCRIS, type RO, offset 0x00C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | PRIS | ARIS |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **FCIM, type R/W, offset 0x010, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | PMASK | AMASK |
| **FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | PMISC | AMISC |
| **FMC2, type R/W, offset 0x020, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| WRKEY | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | WRBUF |
| **FWBVAL, type R/W, offset 0x030, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| FWB[n] | | | | | | | | | | | | | | | |
| FWB[n] | | | | | | | | | | | | | | | |
| **FCTL, type R/W, offset 0x0F8, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | USDACK | USDREQ |
| **FWBn, type R/W, offset 0x100 - 0x17C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **Internal Memory**<br>**Memory Registers (System Control Offset)**<br>Base 0x400F.E000 | | | | | | | | | | | | | | | |
| **RMCTL, type R/W1C, offset 0x0F0, reset -** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | BA |
| **FMPRE0, type R/W, offset 0x130 and 0x200, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE0, type R/W, offset 0x134 and 0x400, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **BOOTCFG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE** | | | | | | | | | | | | | | | |
| NW | | | | | | | | | | | | | | | |
| PORT | | | PIN | | | POL | EN | | | | | | | DBG1 | DBG0 |
| **USER_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| NW | DATA | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| NW | DATA | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| NW | DATA | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| NW | DATA | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **FMPRE1, type R/W, offset 0x204, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| **FMPRE2, type R/W, offset 0x208, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **FMPRE3, type R/W, offset 0x20C, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| **FMPRE4, type R/W, offset 0x210, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| **FMPRE5, type R/W, offset 0x214, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| **FMPRE6, type R/W, offset 0x218, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| **FMPRE7, type R/W, offset 0x21C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| READ_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE1, type R/W, offset 0x404, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE2, type R/W, offset 0x408, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE3, type R/W, offset 0x40C, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE4, type R/W, offset 0x410, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE5, type R/W, offset 0x414, reset 0xFFFF.FFFF** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE6, type R/W, offset 0x418, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **FMPPE7, type R/W, offset 0x41C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| PROG_ENABLE | | | | | | | | | | | | | | | |
| **Micro Direct Memory Access (µDMA)**<br>**µDMA Channel Control Structure (Offset from Channel Control Table Base)**<br>Base n/a | | | | | | | | | | | | | | | |
| **DMASRCENDP, type R/W, offset 0x000, reset -** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| **DMADSTENDP, type R/W, offset 0x004, reset -** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| **DMACHCTL, type R/W, offset 0x008, reset -** | | | | | | | | | | | | | | | |
| DSTINC | | DSTSIZE | | SRCINC | | SRCSIZE | | | | | | | | ARBSIZE | |
| ARBSIZE | | | | XFERSIZE | | | | | | | | NXTUSEBURST | | XFERMODE | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Micro Direct Memory Access (µDMA)**<br>**µDMA Registers (Offset from µDMA Base Address)**<br>Base 0x400F.F000 | | | | | | | | | | | | | | | |
| **DMASTAT, type RO, offset 0x000, reset 0x001F.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | DMACHANS | | | | |
| | | | | | | | | STATE | | | | | | | MASTEN |
| **DMACFG, type WO, offset 0x004, reset -** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | MASTEN |
| **DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| **DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200** | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| **DMAWAITSTAT, type RO, offset 0x010, reset 0xFFFF.FFC0** | | | | | | | | | | | | | | | |
| WAITREQ[n] | | | | | | | | | | | | | | | |
| WAITREQ[n] | | | | | | | | | | | | | | | |
| **DMASWREQ, type WO, offset 0x014, reset -** | | | | | | | | | | | | | | | |
| SWREQ[n] | | | | | | | | | | | | | | | |
| SWREQ[n] | | | | | | | | | | | | | | | |
| **DMAUSEBURSTSET, type R/W, offset 0x018, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| **DMAUSEBURSTCLR, type WO, offset 0x01C, reset -** | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| **DMAREQMASKSET, type R/W, offset 0x020, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| **DMAREQMASKCLR, type WO, offset 0x024, reset -** | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| **DMAENASET, type R/W, offset 0x028, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| **DMAENACLR, type WO, offset 0x02C, reset -** | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| **DMAALTSET, type R/W, offset 0x030, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| **DMAALTCLR, type WO, offset 0x034, reset -** | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| **DMAPRIOSET, type R/W, offset 0x038, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| SET[n] | | | | | | | | | | | | | | | |
| **DMAPRIOCLR, type WO, offset 0x03C, reset -** | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |
| CLR[n] | | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **DMAERRCLR, type R/W, offset 0x04C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | ERRCLR |
| **DMACHASGN, type R/W, offset 0x500, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| CHASGN[n] | | | | | | | | | | | | | | | |
| CHASGN[n] | | | | | | | | | | | | | | | |
| **DMACHIS, type R/W1C, offset 0x504, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| CHIS[n] | | | | | | | | | | | | | | | |
| CHIS[n] | | | | | | | | | | | | | | | |
| **DMAPeriphID0, type RO, offset 0xFE0, reset 0x0000.0030** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| PID0 | | | | | | | | | | | | | | | |
| **DMAPeriphID1, type RO, offset 0xFE4, reset 0x0000.00B2** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| PID1 | | | | | | | | | | | | | | | |
| **DMAPeriphID2, type RO, offset 0xFE8, reset 0x0000.000B** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| PID2 | | | | | | | | | | | | | | | |
| **DMAPeriphID3, type RO, offset 0xFEC, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| PID3 | | | | | | | | | | | | | | | |
| **DMAPeriphID4, type RO, offset 0xFD0, reset 0x0000.0004** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| PID4 | | | | | | | | | | | | | | | |
| **DMAPCellID0, type RO, offset 0xFF0, reset 0x0000.000D** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| CID0 | | | | | | | | | | | | | | | |
| **DMAPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| CID1 | | | | | | | | | | | | | | | |
| **DMAPCellID2, type RO, offset 0xFF8, reset 0x0000.0005** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| CID2 | | | | | | | | | | | | | | | |
| **DMAPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| CID3 | | | | | | | | | | | | | | | |

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

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
GPIO Port G (APB) base: 0x4002.6000
GPIO Port G (AHB) base: 0x4005.E000
GPIO Port H (APB) base: 0x4002.7000
GPIO Port H (AHB) base: 0x4005.F000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **GPIODATA, type R/W, offset 0x000, reset 0x0000.0000** (see page 408) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **GPIODIR, type R/W, offset 0x400, reset 0x0000.0000** (see page 409) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DIR | | | | | | | |
| **GPIOIS, type R/W, offset 0x404, reset 0x0000.0000** (see page 410) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | IS | | | | | | | |
| **GPIOIBE, type R/W, offset 0x408, reset 0x0000.0000** (see page 411) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | IBE | | | | | | | |
| **GPIOIEV, type R/W, offset 0x40C, reset 0x0000.0000** (see page 412) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | IEV | | | | | | | |
| **GPIOIM, type R/W, offset 0x410, reset 0x0000.0000** (see page 413) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | IME | | | | | | | |
| **GPIORIS, type RO, offset 0x414, reset 0x0000.0000** (see page 414) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | RIS | | | | | | | |
| **GPIOMIS, type RO, offset 0x418, reset 0x0000.0000** (see page 415) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | MIS | | | | | | | |
| **GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000** (see page 417) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | IC | | | | | | | |
| **GPIOAFSEL, type R/W, offset 0x420, reset -** (see page 418) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | AFSEL | | | | | | | |
| **GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF** (see page 420) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DRV2 | | | | | | | |
| **GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000** (see page 421) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DRV4 | | | | | | | |
| **GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000** (see page 422) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DRV8 | | | | | | | |
| **GPIOODR, type R/W, offset 0x50C, reset 0x0000.0000** (see page 423) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | ODE | | | | | | | |
| **GPIOPUR, type R/W, offset 0x510, reset -** (see page 424) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PUE | | | | | | | |
| **GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000** (see page 426) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PDE | | | | | | | |
| **GPIOSLR, type R/W, offset 0x518, reset 0x0000.0000** (see page 428) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | SRL | | | | | | | |
| **GPIODEN, type R/W, offset 0x51C, reset -** (see page 429) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DEN | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001** (see page 431) | | | | | | | | | | | | | | | |
| LOCK | | | | | | | | | | | | | | | |
| LOCK | | | | | | | | | | | | | | | |
| **GPIOCR, type -, offset 0x524, reset -** (see page 432) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CR | | | | | | | |
| **GPIOAMSEL, type R/W, offset 0x528, reset 0x0000.0000** (see page 434) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | GPIOAMSEL | | | | | | | |
| **GPIOPCTL, type R/W, offset 0x52C, reset -** (see page 435) | | | | | | | | | | | | | | | |
| PMC7 | | | | PMC6 | | | | PMC5 | | | | PMC4 | | | |
| PMC3 | | | | PMC2 | | | | PMC1 | | | | PMC0 | | | |
| **GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000** (see page 437) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID4 | | | | | | | |
| **GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000** (see page 438) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID5 | | | | | | | |
| **GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000** (see page 439) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID6 | | | | | | | |
| **GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000** (see page 440) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID7 | | | | | | | |
| **GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061** (see page 441) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID0 | | | | | | | |
| **GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000** (see page 442) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID1 | | | | | | | |
| **GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018** (see page 443) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID2 | | | | | | | |
| **GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001** (see page 444) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID3 | | | | | | | |
| **GPIOPCellID0, type RO, offset 0xFF0, reset 0x0000.000D** (see page 445) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID0 | | | | | | | |
| **GPIOPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0** (see page 446) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID1 | | | | | | | |
| **GPIOPCellID2, type RO, offset 0xFF8, reset 0x0000.0005** (see page 447) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID2 | | | | | | | |
| **GPIOPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1** (see page 448) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID3 | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **General-Purpose Timers**<br>Timer 0 base: 0x4003.0000<br>Timer 1 base: 0x4003.1000<br>Timer 2 base: 0x4003.2000<br>Timer 3 base: 0x4003.3000 | | | | | | | | | | | | | | | |
| **GPTMCFG, type R/W, offset 0x000, reset 0x0000.0000** (see page 465) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | GPTMCFG | | |
| **GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000** (see page 466) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | TASNAPS | TAWOT | TAMIE | TACDIR | TAAMS | TACMR | TAMR | |
| **GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000** (see page 468) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | TBSNAPS | TBWOT | TBMIE | TBCDIR | TBAMS | TBCMR | TBMR | |
| **GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000** (see page 470) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | TBPWML | TBOTE | | TBEVENT | | TBSTALL | TBEN | | TAPWML | TAOTE | RTCEN | TAEVENT | | TASTALL | TAEN |
| **GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000** (see page 473) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | TBMIM | CBEIM | CBMIM | TBTOIM | | | | TAMIM | RTCIM | CAEIM | CAMIM | TATOIM |
| **GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000** (see page 475) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | TBMRIS | CBERIS | CBMRIS | TBTORIS | | | | TAMRIS | RTCRIS | CAERIS | CAMRIS | TATORIS |
| **GPTMMIS, type RO, offset 0x020, reset 0x0000.0000** (see page 478) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | TBMMIS | CBEMIS | CBMMIS | TBTOMIS | | | | TAMMIS | RTCMIS | CAEMIS | CAMMIS | TATOMIS |
| **GPTMICR, type W1C, offset 0x024, reset 0x0000.0000** (see page 481) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | TBMCINT | CBECINT | CBMCINT | TBTOCINT | | | | TAMCINT | RTCCINT | CAECINT | CAMCINT | TATOCINT |
| **GPTMTAILR, type R/W, offset 0x028, reset 0xFFFF.FFFF** (see page 483) | | | | | | | | | | | | | | | |
| | | | | | | | | TAILR | | | | | | | |
| | | | | | | | | TAILR | | | | | | | |
| **GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF** (see page 484) | | | | | | | | | | | | | | | |
| | | | | | | | | TBILR | | | | | | | |
| | | | | | | | | TBILR | | | | | | | |
| **GPTMTAMATCHR, type R/W, offset 0x030, reset 0xFFFF.FFFF** (see page 485) | | | | | | | | | | | | | | | |
| | | | | | | | | TAMR | | | | | | | |
| | | | | | | | | TAMR | | | | | | | |
| **GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF** (see page 486) | | | | | | | | | | | | | | | |
| | | | | | | | | TBMR | | | | | | | |
| | | | | | | | | TBMR | | | | | | | |
| **GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000** (see page 487) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | TAPSR | | | | | | | |
| **GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000** (see page 488) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | TBPSR | | | | | | | |
| **GPTMTAPMR, type R/W, offset 0x040, reset 0x0000.0000** (see page 489) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | TAPSMR | | | | | | | |
| **GPTMTBPMR, type R/W, offset 0x044, reset 0x0000.0000** (see page 490) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | TBPSMR | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **GPTMTAR, type RO, offset 0x048, reset 0xFFFF.FFFF** (see page 491) | | | | | | | | | | | | | | | |
| TAR | | | | | | | | | | | | | | | |
| TAR | | | | | | | | | | | | | | | |
| **GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF** (see page 492) | | | | | | | | | | | | | | | |
| TBR | | | | | | | | | | | | | | | |
| TBR | | | | | | | | | | | | | | | |
| **GPTMTAV, type RW, offset 0x050, reset 0xFFFF.FFFF** (see page 493) | | | | | | | | | | | | | | | |
| TAV | | | | | | | | | | | | | | | |
| TAV | | | | | | | | | | | | | | | |
| **GPTMTBV, type RW, offset 0x054, reset 0x0000.FFFF** (see page 494) | | | | | | | | | | | | | | | |
| TBV | | | | | | | | | | | | | | | |
| TBV | | | | | | | | | | | | | | | |
| **Watchdog Timers**<br>WDT0 base: 0x4000.0000<br>WDT1 base: 0x4000.1000 | | | | | | | | | | | | | | | |
| **WDTLOAD, type R/W, offset 0x000, reset 0xFFFF.FFFF** (see page 499) | | | | | | | | | | | | | | | |
| WDTLOAD | | | | | | | | | | | | | | | |
| WDTLOAD | | | | | | | | | | | | | | | |
| **WDTVALUE, type RO, offset 0x004, reset 0xFFFF.FFFF** (see page 500) | | | | | | | | | | | | | | | |
| WDTVALUE | | | | | | | | | | | | | | | |
| WDTVALUE | | | | | | | | | | | | | | | |
| **WDTCTL, type R/W, offset 0x008, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1)** (see page 501) | | | | | | | | | | | | | | | |
| WRC | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | RESEN | INTEN |
| **WDTICR, type WO, offset 0x00C, reset -** (see page 503) | | | | | | | | | | | | | | | |
| WDTINTCLR | | | | | | | | | | | | | | | |
| WDTINTCLR | | | | | | | | | | | | | | | |
| **WDTRIS, type RO, offset 0x010, reset 0x0000.0000** (see page 504) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | WDTRIS |
| **WDTMIS, type RO, offset 0x014, reset 0x0000.0000** (see page 505) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | WDTMIS |
| **WDTTEST, type R/W, offset 0x418, reset 0x0000.0000** (see page 506) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | STALL | | | | | | | | |
| **WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000** (see page 507) | | | | | | | | | | | | | | | |
| WDTLOCK | | | | | | | | | | | | | | | |
| WDTLOCK | | | | | | | | | | | | | | | |
| **WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000** (see page 508) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID4 | | | | | | | |
| **WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000** (see page 509) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID5 | | | | | | | |
| **WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000** (see page 510) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID6 | | | | | | | |
| **WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000** (see page 511) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID7 | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005** (see page 512) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID0 | | | | | | | |
| **WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018** (see page 513) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID1 | | | | | | | |
| **WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018** (see page 514) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID2 | | | | | | | |
| **WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001** (see page 515) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID3 | | | | | | | |
| **WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D** (see page 516) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID0 | | | | | | | |
| **WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0** (see page 517) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID1 | | | | | | | |
| **WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0006** (see page 518) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID2 | | | | | | | |
| **WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1** (see page 519) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID3 | | | | | | | |
| **Analog-to-Digital Converter (ADC)**<br>ADC0 base: 0x4003.8000<br>ADC1 base: 0x4003.9000 | | | | | | | | | | | | | | | |
| **ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000** (see page 542) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | ASEN3 | ASEN2 | ASEN1 | ASEN0 |
| **ADCRIS, type RO, offset 0x004, reset 0x0000.0000** (see page 543) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | INRDC |
| | | | | | | | | | | | | INR3 | INR2 | INR1 | INR0 |
| **ADCIM, type R/W, offset 0x008, reset 0x0000.0000** (see page 545) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DCONSS3 | DCONSS2 | DCONSS1 | DCONSS0 |
| | | | | | | | | | | | | MASK3 | MASK2 | MASK1 | MASK0 |
| **ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000** (see page 547) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DCINSS3 | DCINSS2 | DCINSS1 | DCINSS0 |
| | | | | | | | | | | | | IN3 | IN2 | IN1 | IN0 |
| **ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000** (see page 550) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | OV3 | OV2 | OV1 | OV0 |
| **ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000** (see page 552) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| EM3 | | | | EM2 | | | | EM1 | | | | EM0 | | | |
| **ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000** (see page 556) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | UV3 | UV2 | UV1 | UV0 |
| **ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210** (see page 557) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | SS3 | | | | SS2 | | | | SS1 | | | | SS0 | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **ADCSPC, type R/W, offset 0x024, reset 0x0000.0000** (see page 559) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PHASE | | | |
| **ADCPSSI, type R/W, offset 0x028, reset -** (see page 561) | | | | | | | | | | | | | | | |
| GSYNC | | | | SYNCWAIT | | | | | | | | | | | |
| | | | | | | | | | | | | SS3 | SS2 | SS1 | SS0 |
| **ADCSAC, type R/W, offset 0x030, reset 0x0000.0000** (see page 563) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | AVG | | |
| **ADCDCISC, type R/W1C, offset 0x034, reset 0x0000.0000** (see page 564) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DCINT7 | DCINT6 | DCINT5 | DCINT4 | DCINT3 | DCINT2 | DCINT1 | DCINT0 |
| **ADCCTL, type R/W, offset 0x038, reset 0x0000.0000** (see page 566) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | RES | | | VREF | |
| **ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000** (see page 567) | | | | | | | | | | | | | | | |
| MUX7 | | | | MUX6 | | | | MUX5 | | | | MUX4 | | | |
| MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| **ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000** (see page 569) | | | | | | | | | | | | | | | |
| TS7 | IE7 | END7 | D7 | TS6 | IE6 | END6 | D6 | TS5 | IE5 | END5 | D5 | TS4 | IE4 | END4 | D4 |
| TS3 | IE3 | END3 | D3 | TS2 | IE2 | END2 | D2 | TS1 | IE1 | END1 | D1 | TS0 | IE0 | END0 | D0 |
| **ADCSSFIFO0, type RO, offset 0x048, reset -** (see page 572) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | DATA | | | | | | | | | | | |
| **ADCSSFIFO1, type RO, offset 0x068, reset -** (see page 572) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | DATA | | | | | | | | | | | |
| **ADCSSFIFO2, type RO, offset 0x088, reset -** (see page 572) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | DATA | | | | | | | | | | | |
| **ADCSSFIFO3, type RO, offset 0x0A8, reset -** (see page 572) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | DATA | | | | | | | | | | | |
| **ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100** (see page 573) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | FULL | | | | EMPTY | HPTR | | | | TPTR | | | |
| **ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100** (see page 573) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | FULL | | | | EMPTY | HPTR | | | | TPTR | | | |
| **ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100** (see page 573) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | FULL | | | | EMPTY | HPTR | | | | TPTR | | | |
| **ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100** (see page 573) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | FULL | | | | EMPTY | HPTR | | | | TPTR | | | |
| **ADCSSOP0, type R/W, offset 0x050, reset 0x0000.0000** (see page 575) | | | | | | | | | | | | | | | |
| | | | S7DCOP | | | | S6DCOP | | | | S5DCOP | | | | S4DCOP |
| | | | S3DCOP | | | | S2DCOP | | | | S1DCOP | | | | S0DCOP |
| **ADCSSDC0, type R/W, offset 0x054, reset 0x0000.0000** (see page 577) | | | | | | | | | | | | | | | |
| S7DCSEL | | | | S6DCSEL | | | | S5DCSEL | | | | S4DCSEL | | | |
| S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000** (see page 579) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| **ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000** (see page 579) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| **ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000** (see page 580) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| TS3 | IE3 | END3 | D3 | TS2 | IE2 | END2 | D2 | TS1 | IE1 | END1 | D1 | TS0 | IE0 | END0 | D0 |
| **ADCSSCTL2, type R/W, offset 0x084, reset 0x0000.0000** (see page 580) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| TS3 | IE3 | END3 | D3 | TS2 | IE2 | END2 | D2 | TS1 | IE1 | END1 | D1 | TS0 | IE0 | END0 | D0 |
| **ADCSSOP1, type R/W, offset 0x070, reset 0x0000.0000** (see page 582) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | S3DCOP | | | | S2DCOP | | | | S1DCOP | | | | S0DCOP |
| **ADCSSOP2, type R/W, offset 0x090, reset 0x0000.0000** (see page 582) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | S3DCOP | | | | S2DCOP | | | | S1DCOP | | | | S0DCOP |
| **ADCSSDC1, type R/W, offset 0x074, reset 0x0000.0000** (see page 583) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |
| **ADCSSDC2, type R/W, offset 0x094, reset 0x0000.0000** (see page 583) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |
| **ADCSSMUX3, type R/W, offset 0x0A0, reset 0x0000.0000** (see page 585) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | MUX0 | | | |
| **ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002** (see page 586) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TS0 | IE0 | END0 | D0 |
| **ADCSSOP3, type R/W, offset 0x0B0, reset 0x0000.0000** (see page 587) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | S0DCOP |
| **ADCSSDC3, type R/W, offset 0x0B4, reset 0x0000.0000** (see page 588) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | S0DCSEL | | | |
| **ADCDCRIC, type R/W, offset 0xD00, reset 0x0000.0000** (see page 589) | | | | | | | | | | | | | | | |
| | | | | | | | | DCTRIG7 | DCTRIG6 | DCTRIG5 | DCTRIG4 | DCTRIG3 | DCTRIG2 | DCTRIG1 | DCTRIG0 |
| | | | | | | | | DCINT7 | DCINT6 | DCINT5 | DCINT4 | DCINT3 | DCINT2 | DCINT1 | DCINT0 |
| **ADCDCCTL0, type R/W, offset 0xE00, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |
| **ADCDCCTL1, type R/W, offset 0xE04, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |
| **ADCDCCTL2, type R/W, offset 0xE08, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |
| **ADCDCCTL3, type R/W, offset 0xE0C, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **ADCDCCTL4, type R/W, offset 0xE10, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |
| **ADCDCCTL5, type R/W, offset 0xE14, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |
| **ADCDCCTL6, type R/W, offset 0xE18, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |
| **ADCDCCTL7, type R/W, offset 0xE1C, reset 0x0000.0000** (see page 594) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | CIE | CIC | | CIM | |
| **ADCDCCMP0, type R/W, offset 0xE40, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **ADCDCCMP1, type R/W, offset 0xE44, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **ADCDCCMP2, type R/W, offset 0xE48, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **ADCDCCMP3, type R/W, offset 0xE4C, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **ADCDCCMP4, type R/W, offset 0xE50, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **ADCDCCMP5, type R/W, offset 0xE54, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **ADCDCCMP6, type R/W, offset 0xE58, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **ADCDCCMP7, type R/W, offset 0xE5C, reset 0x0000.0000** (see page 596) | | | | | | | | | | | | | | | |
| | | | | COMP1 | | | | | | | | | | | |
| | | | | COMP0 | | | | | | | | | | | |
| **Universal Asynchronous Receivers/Transmitters (UARTs)**<br>UART0 base: 0x4000.C000<br>UART1 base: 0x4000.D000<br>UART2 base: 0x4000.E000 | | | | | | | | | | | | | | | |
| **UARTDR, type R/W, offset 0x000, reset 0x0000.0000** (see page 612) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | OE | BE | PE | FE | DATA | | | | | | | |
| **UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register)** (see page 614) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | OE | BE | PE | FE |
| **UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Error Clear Register)** (see page 614) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DATA | | | | | | | |
| **UARTFR, type RO, offset 0x018, reset 0x0000.0090** (see page 617) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | RI | TXFE | RXFF | TXFF | RXFE | BUSY | DCD | DSR | CTS |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **UARTILPR, type R/W, offset 0x020, reset 0x0000.0000** (see page 620) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | ILPDVSR | | | | | | | |
| **UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000** (see page 621) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| DIVINT | | | | | | | | | | | | | | | |
| **UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000** (see page 622) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | DIVFRAC | | | | | |
| **UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000** (see page 623) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | SPS | WLEN | | FEN | STP2 | EPS | PEN | BRK |
| **UARTCTL, type R/W, offset 0x030, reset 0x0000.0300** (see page 625) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| CTSEN | RTSEN | | | RTS | DTR | RXE | TXE | LBE | LIN | HSE | EOT | SMART | SIRLP | SIREN | UARTEN |
| **UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012** (see page 629) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | RXIFLSEL | | | TXIFLSEL | | |
| **UARTIM, type R/W, offset 0x038, reset 0x0000.0000** (see page 631) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| LME5IM | LME1IM | LMSBIM | | | OEIM | BEIM | PEIM | FEIM | RTIM | TXIM | RXIM | DSRIM | DCDIM | CTSIM | RIIM |
| **UARTRIS, type RO, offset 0x03C, reset 0x0000.000F** (see page 635) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| LME5RIS | LME1RIS | LMSBRIS | | | OERIS | BERIS | PERIS | FERIS | RTRIS | TXRIS | RXRIS | DSRRIS | DCDRIS | CTSRIS | RIRIS |
| **UARTMIS, type RO, offset 0x040, reset 0x0000.0000** (see page 638) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| LME5MIS | LME1MIS | LMSBMIS | | | OEMIS | BEMIS | PEMIS | FEMIS | RTMIS | TXMIS | RXMIS | DSRMIS | DCDMIS | CTSMIS | RIMIS |
| **UARTICR, type W1C, offset 0x044, reset 0x0000.0000** (see page 641) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| LME5IC | LME1IC | LMSBIC | | | OEIC | BEIC | PEIC | FEIC | RTIC | TXIC | RXIC | DSRMIC | DCDMIC | CTSMIC | RIMIC |
| **UARTDMACTL, type R/W, offset 0x048, reset 0x0000.0000** (see page 643) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | DMAERR | TXDMAE | RXDMAE |
| **UARTLCTL, type R/W, offset 0x090, reset 0x0000.0000** (see page 644) | | | | | | | | | | | | | | | |
| | | | | | | | | | | BLEN | | | | | MASTER |
| **UARTLSS, type RO, offset 0x094, reset 0x0000.0000** (see page 645) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| TSS | | | | | | | | | | | | | | | |
| **UARTLTIM, type RO, offset 0x098, reset 0x0000.0000** (see page 646) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| TIMER | | | | | | | | | | | | | | | |
| **UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000** (see page 647) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID4 | | | | | | | |
| **UARTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000** (see page 648) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID5 | | | | | | | |
| **UARTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000** (see page 649) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID6 | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **UARTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000** (see page 650) | | | | | | | | | | | | | | | |
| | | | | | | | | PID7 | | | | | | | |
| **UARTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0060** (see page 651) | | | | | | | | | | | | | | | |
| | | | | | | | | PID0 | | | | | | | |
| **UARTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000** (see page 652) | | | | | | | | | | | | | | | |
| | | | | | | | | PID1 | | | | | | | |
| **UARTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018** (see page 653) | | | | | | | | | | | | | | | |
| | | | | | | | | PID2 | | | | | | | |
| **UARTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001** (see page 654) | | | | | | | | | | | | | | | |
| | | | | | | | | PID3 | | | | | | | |
| **UARTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D** (see page 655) | | | | | | | | | | | | | | | |
| | | | | | | | | CID0 | | | | | | | |
| **UARTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0** (see page 656) | | | | | | | | | | | | | | | |
| | | | | | | | | CID1 | | | | | | | |
| **UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005** (see page 657) | | | | | | | | | | | | | | | |
| | | | | | | | | CID2 | | | | | | | |
| **UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1** (see page 658) | | | | | | | | | | | | | | | |
| | | | | | | | | CID3 | | | | | | | |
| **Synchronous Serial Interface (SSI)**<br>SSI0 base: 0x4000.8000<br>SSI1 base: 0x4000.9000 | | | | | | | | | | | | | | | |
| **SSICR0, type R/W, offset 0x000, reset 0x0000.0000** (see page 674) | | | | | | | | | | | | | | | |
| SCR | | | | | | | | SPH | SPO | FRF | | DSS | | | |
| **SSICR1, type R/W, offset 0x004, reset 0x0000.0000** (see page 676) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | EOT | SOD | MS | SSE | LBM |
| **SSIDR, type R/W, offset 0x008, reset 0x0000.0000** (see page 678) | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| **SSISR, type RO, offset 0x00C, reset 0x0000.0003** (see page 679) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | BSY | RFF | RNE | TNF | TFE |
| **SSICPSR, type R/W, offset 0x010, reset 0x0000.0000** (see page 681) | | | | | | | | | | | | | | | |
| | | | | | | | | CPSDVSR | | | | | | | |
| **SSIIM, type R/W, offset 0x014, reset 0x0000.0000** (see page 682) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXIM | RXIM | RTIM | RORIM |
| **SSIRIS, type RO, offset 0x018, reset 0x0000.0008** (see page 683) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXRIS | RXRIS | RTRIS | RORRIS |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **SSIMIS, type RO, offset 0x01C, reset 0x0000.0000** (see page 685) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXMIS | RXMIS | RTMIS | RORMIS |
| **SSIICR, type W1C, offset 0x020, reset 0x0000.0000** (see page 687) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | RTIC | RORIC |
| **SSIDMACTL, type R/W, offset 0x024, reset 0x0000.0000** (see page 688) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | TXDMAE | RXDMAE |
| **SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000** (see page 689) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID4 | | | | | | | |
| **SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000** (see page 690) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID5 | | | | | | | |
| **SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000** (see page 691) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID6 | | | | | | | |
| **SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000** (see page 692) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID7 | | | | | | | |
| **SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022** (see page 693) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID0 | | | | | | | |
| **SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000** (see page 694) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID1 | | | | | | | |
| **SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018** (see page 695) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID2 | | | | | | | |
| **SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001** (see page 696) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | PID3 | | | | | | | |
| **SSIPCellID0, type RO, offset 0xFF0, reset 0x0000.000D** (see page 697) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID0 | | | | | | | |
| **SSIPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0** (see page 698) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID1 | | | | | | | |
| **SSIPCellID2, type RO, offset 0xFF8, reset 0x0000.0005** (see page 699) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID2 | | | | | | | |
| **SSIPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1** (see page 700) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | CID3 | | | | | | | |
| **Inter-Integrated Circuit ($I^2C$) Interface**<br>**$I^2C$ Master**<br>I2C 0 base: 0x4002.0000<br>I2C 1 base: 0x4002.1000 | | | | | | | | | | | | | | | |
| **I2CMSA, type R/W, offset 0x000, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | SA | | | | | | | R/S |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **I2CMCS, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register)** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | BUSBSY | IDLE | ARBLST | DATACK | ADRACK | ERROR | BUSY |
| **I2CMCS, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Control Register)** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | ACK | STOP | START | RUN |
| **I2CMDR, type R/W, offset 0x008, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DATA | | | | | | | |
| **I2CMTPR, type R/W, offset 0x00C, reset 0x0000.0001** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | TPR | | | | | | |
| **I2CMIMR, type R/W, offset 0x010, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | IM |
| **I2CMRIS, type RO, offset 0x014, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | RIS |
| **I2CMMIS, type RO, offset 0x018, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | MIS |
| **I2CMICR, type WO, offset 0x01C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | IC |
| **I2CMCR, type R/W, offset 0x020, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | SFE | MFE | | | | LPBK |
| **Inter-Integrated Circuit ($I^2C$) Interface**<br>**$I^2C$ Slave**<br>I2C 0 base: 0x4002.0000<br>I2C 1 base: 0x4002.1000 | | | | | | | | | | | | | | | |
| **I2CSOAR, type R/W, offset 0x800, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | OAR | | | | | | |
| **I2CSCSR, type RO, offset 0x804, reset 0x0000.0000 (Read-Only Status Register)** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | FBR | TREQ | RREQ |
| **I2CSCSR, type WO, offset 0x804, reset 0x0000.0000 (Write-Only Control Register)** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | DA |
| **I2CSDR, type R/W, offset 0x808, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | DATA | | | | | | | |
| **I2CSIMR, type R/W, offset 0x80C, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | STOPIM | STARTIM | DATAIM |
| **I2CSRIS, type RO, offset 0x810, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | STOPRIS | STARTRIS | DATARIS |
| **I2CSMIS, type RO, offset 0x814, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | STOPMIS | STARTMIS | DATAMIS |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **I2CSICR, type WO, offset 0x818, reset 0x0000.0000** | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | STOPIC | STARTIC | DATAIC |

# B Ordering and Contact Information

## B.1 Ordering Information

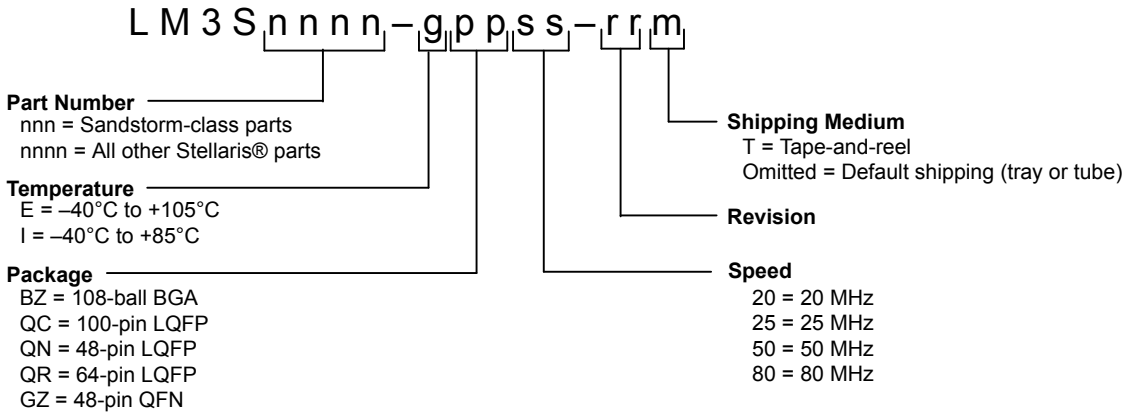


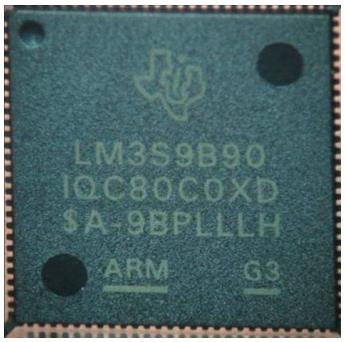

**Table B-1. Part Ordering Information**

| Orderable Part Number | Description |
|---|---|
| LM3S1G58-IQC80-A2 | Stellaris® LM3S1G58 Microcontroller Industrial Temperature 100-pin LQFP |
| LM3S1G58-IBZ80-A2 | Stellaris LM3S1G58 Microcontroller Industrial Temperature 108-ball BGA |
| LM3S1G58-IQC80-A2T | Stellaris LM3S1G58 Microcontroller Industrial Temperature 100-pin LQFP Tape-and-reel |
| LM3S1G58-IBZ80-A2T | Stellaris LM3S1G58 Microcontroller Industrial Temperature 108-ball BGA Tape-and-reel |

## B.2 Part Markings

The Stellaris microcontrollers are marked with an identifying number. This code contains the following information:

- The first line indicates the part number. In the example figure below, this is the LM3S9B90.
- In the second line, the first seven characters indicate the temperature, package, speed, and revision. In the example below, this is an Industrial temperature (I), 100-pin LQFP package (QC), 80-MHz (80), revision C0 (C0) device.
- The remaining characters contain internal tracking numbers.




## B.3 Kits

The Stellaris Family 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 Stellaris 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 website at www.ti.com/stellaris for the latest tools available, or ask your distributor.

## B.4 Support Information

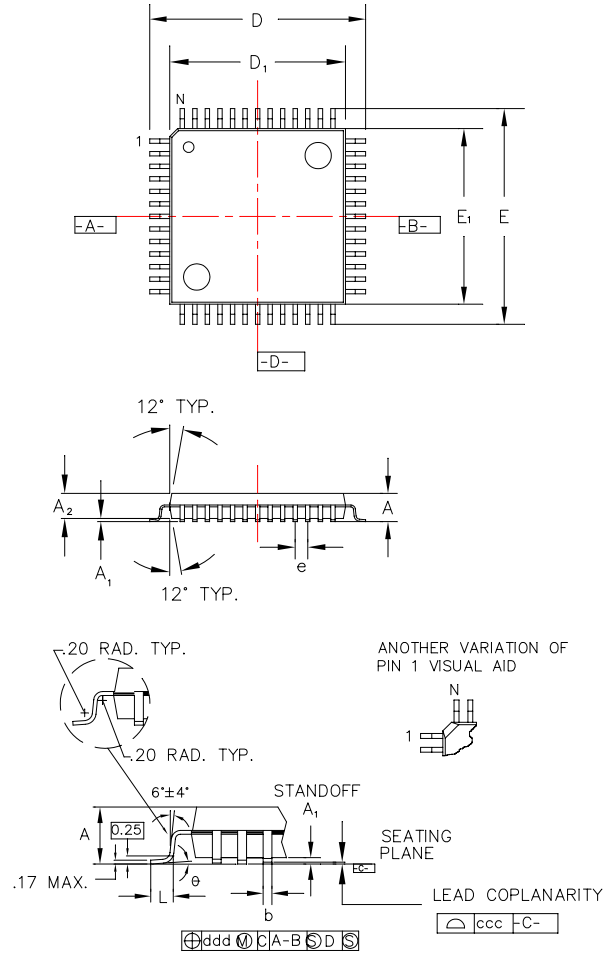
For support on Stellaris products, contact the TI Worldwide Product Information Center nearest you: http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm.

# C Package Information

## C.1 100-Pin LQFP Package

### C.1.1 Package Dimensions

**Figure C-1. 100-Pin LQFP Package Dimensions**



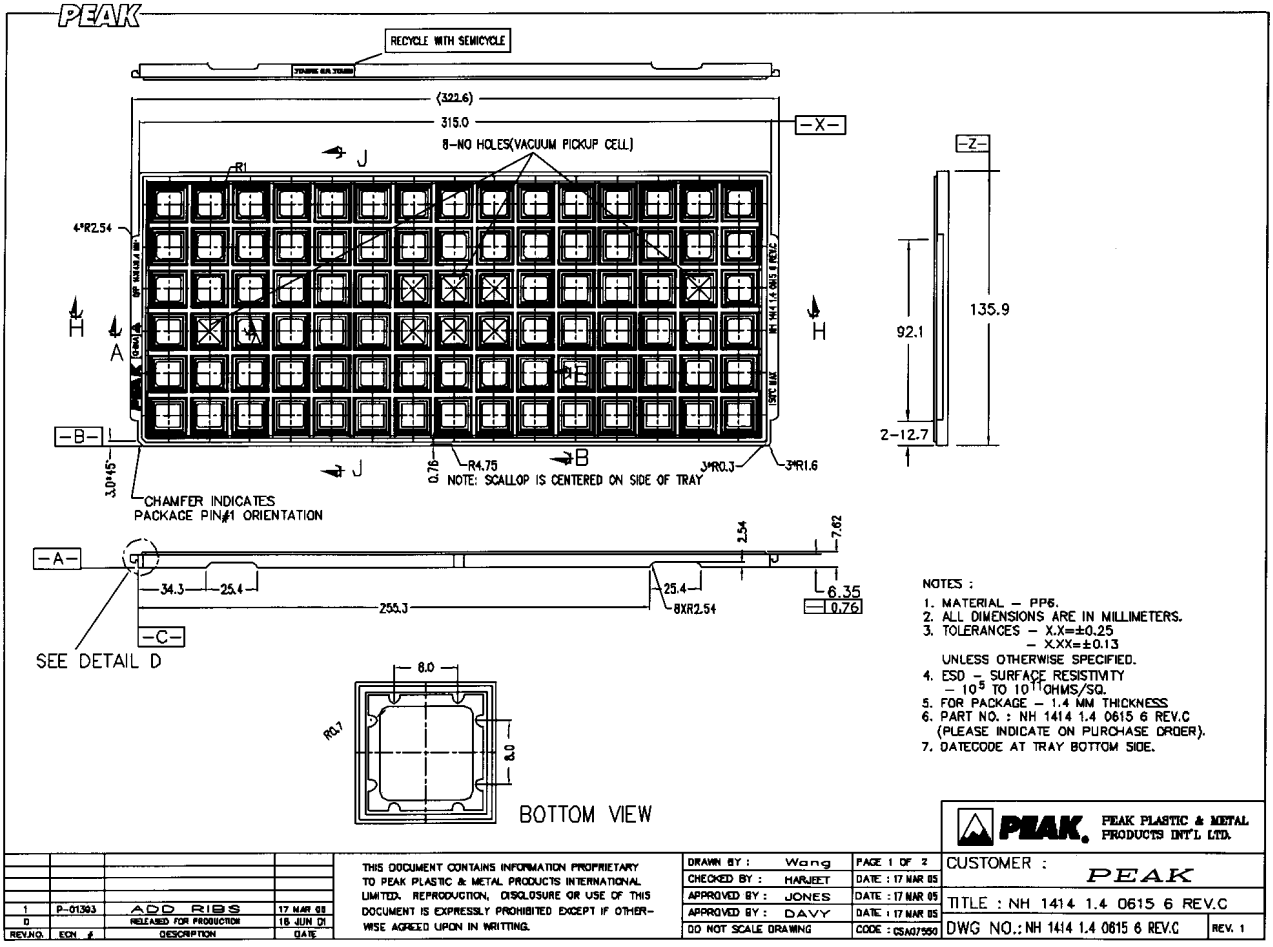

**Note:** The following notes apply to the package drawing.

1. All dimensions shown in mm.
2. Dimensions shown are nominal with tolerances indicated.
3. Foot length 'L' is measured at gage plane 0.25 mm above seating plane.

| Body +2.00 mm Footprint, 1.4 mm package thickness | | |
|---|---|---|
| Symbols | Leads | 100L |
| A | Max. | 1.60 |
| $A_1$ | - | 0.05 Min./0.15 Max. |
| $A_2$ | ±0.05 | 1.40 |
| D | ±0.20 | 16.00 |
| $D_1$ | ±0.05 | 14.00 |
| E | ±0.20 | 16.00 |
| $E_1$ | ±0.05 | 14.00 |
| L | +0.15/-0.10 | 0.60 |
| e | Basic | 0.50 |
| b | +0.05 | 0.22 |
| θ | - | 0˚-7˚ |
| ddd | Max. | 0.08 |
| ccc | Max. | 0.08 |
| JEDEC Reference Drawing | | MS-026 |
| Variation Designator | | BED |

## C.1.2 Tray Dimensions

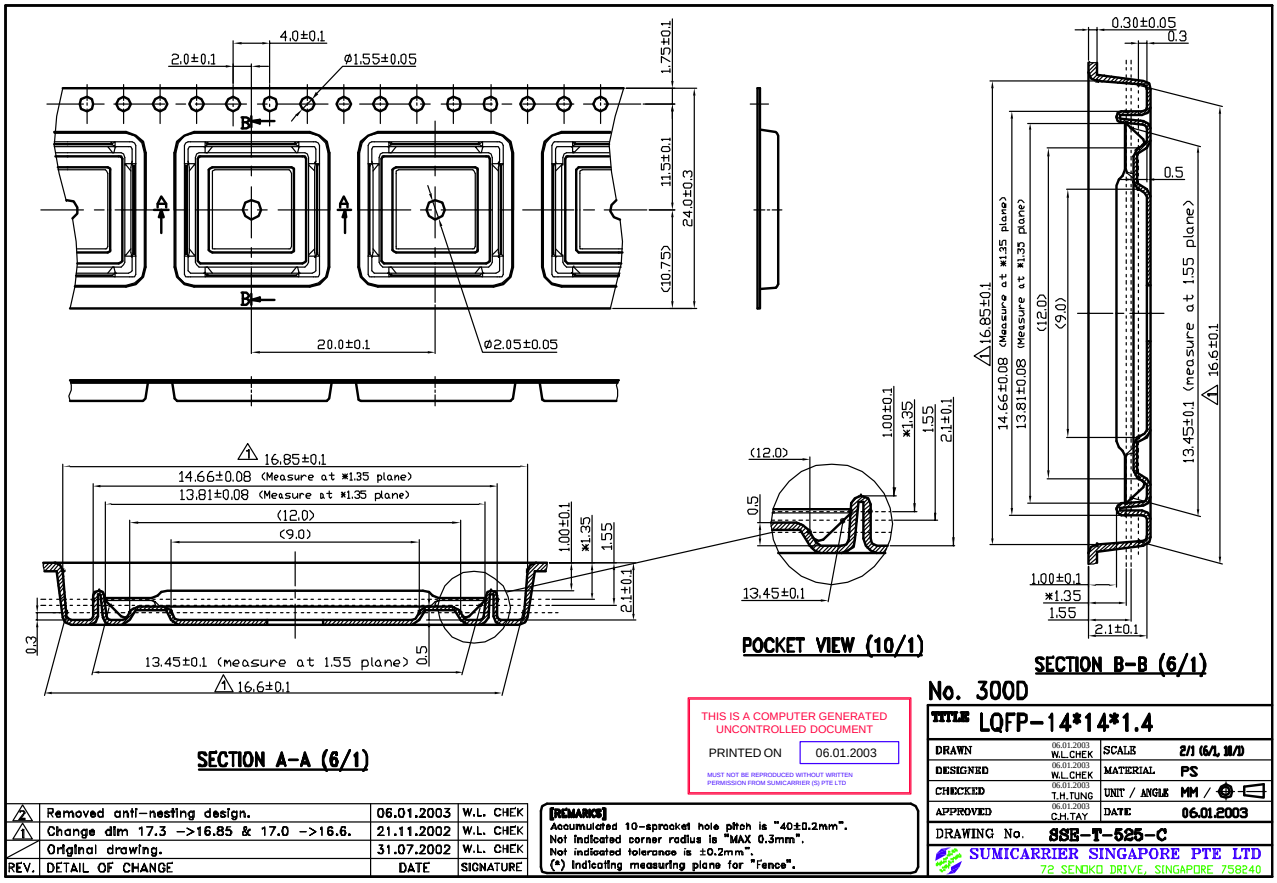
**Figure C-2. 100-Pin LQFP Tray Dimensions**




## C.1.3 Tape and Reel Dimensions

**Note:** In the figure that follows, pin 1 is located in the top right corner of the device.

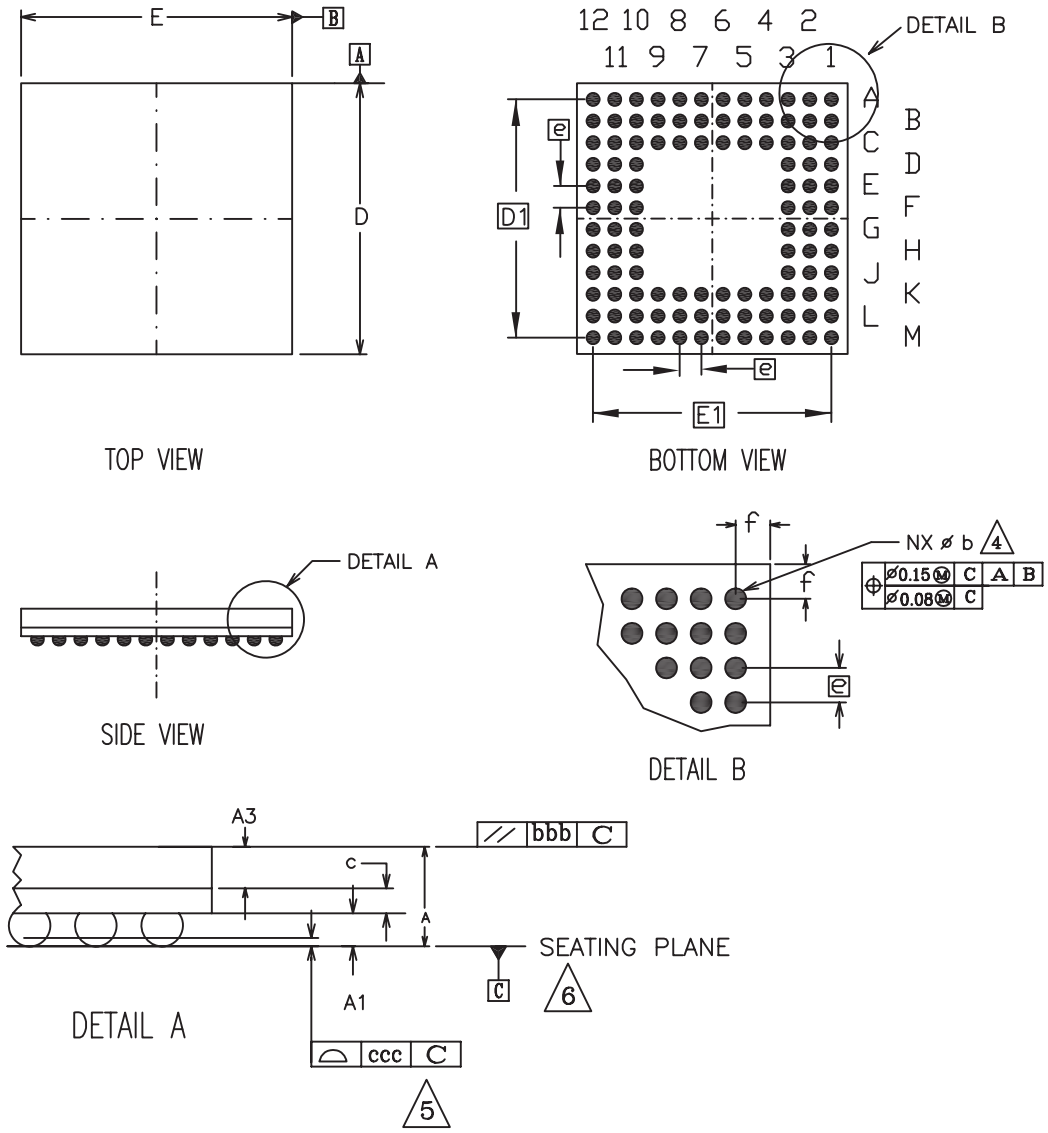
## Figure C-3. 100-Pin LQFP Tape and Reel Dimensions



## C.2 108-Ball BGA Package

### C.2.1 Package Dimensions

**Figure C-4. 108-Ball BGA Package Dimensions**



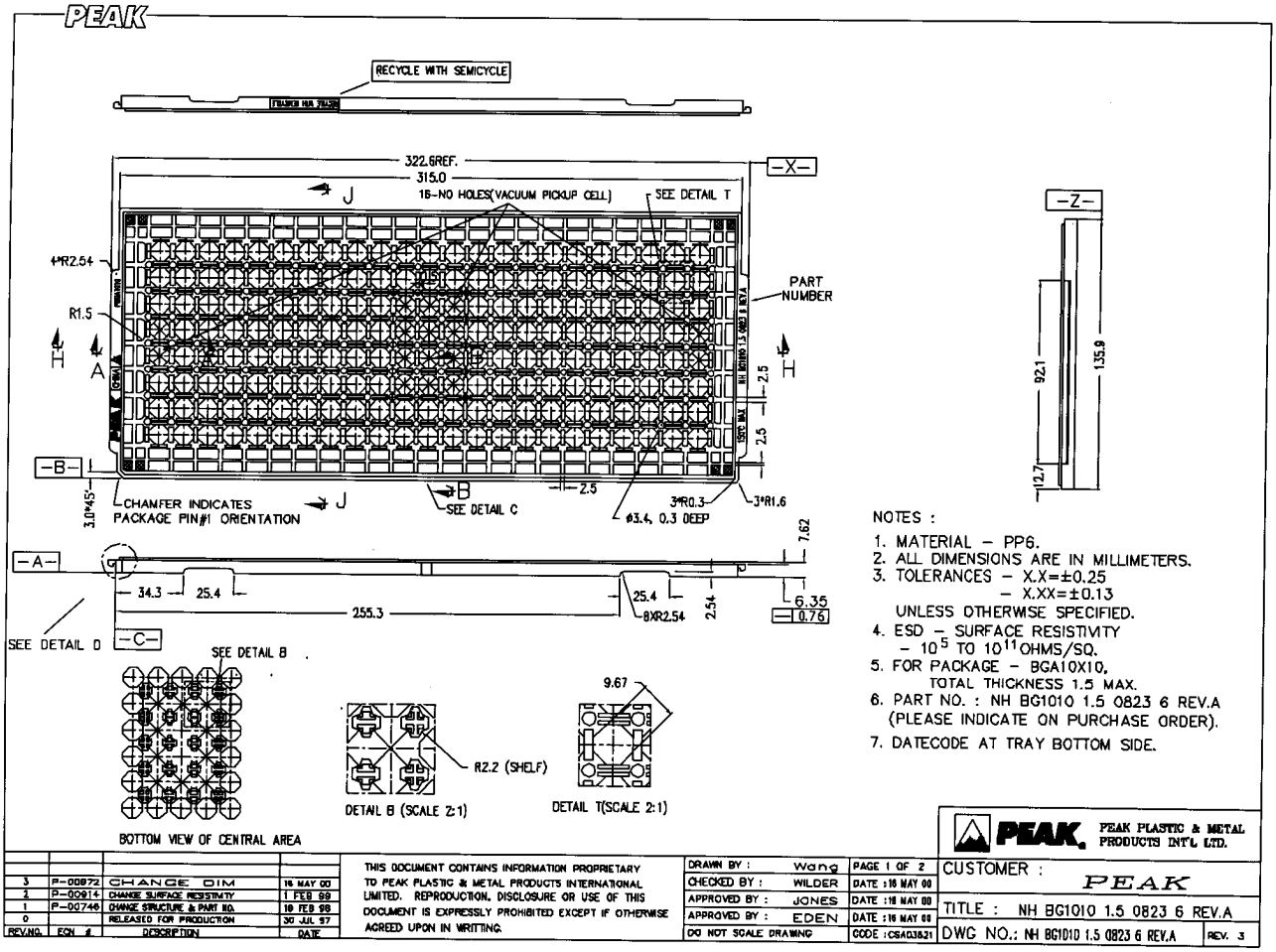
**Note:** The following notes apply to the package drawing.

1. ALL DIMENSIONS ARE IN MILLIMETERS.
2. 'e' REPRESENTS THE BASIC SOLDER BALL GRID PITCH.
3. 'M' REPRESENTS THE BASIC SOLDER BALL MATRIX SIZE.
   AND SYMBOL 'N' IS THE NUMBER OF BALLS AFTER DEPOPULATING.
4. 'b' IS MEASURABLE AT THE MAXIMUM SOLDER BALL DIAMETER AFTER REFLOW
   PARALLEL TO PRIMARY DAIUM [C] .
5. DIMENSION 'ccc' IS MEASURED PARALLEL TO PRIMARY DATUM [C].
6. PRIMARY DATUM [C] AND SEATING PLANE ARE DEFINED BY THE SPHERICAL
   CROWNS OF THE SOLDER BALLS.
7. PACKAGE SURFACE SHALL BE MATTE FINISH CHARMILLES 24 TO 27.
8. SUBSTRATE MATERIAL BASE IS BT RESIN.
9. THE OVERALL PACKAGE THICKNESS "A" ALREADY CONSIDERS COLLAPSE BALLS
10. DIMENSIONING AND TOLERANCING PER ASME Y14.5M 1994.
11. EXCEPT DIMENSION b.

| Symbols | MIN | NOM | MAX |
|---|---|---|---|
| A | 1.22 | 1.36 | 1.50 |
| A1 | 0.29 | 0.34 | 0.39 |
| A3 | 0.65 | 0.70 | 0.75 |
| c | 0.28 | 0.32 | 0.36 |
| D | 9.85 | 10.00 | 10.15 |
| D1 | | 8.80 BSC | |
| E | 9.85 | 10.00 | 10.15 |
| E1 | | 8.80 BSC | |
| b | 0.43 | 0.48 | 0.53 |
| bbb | | .20 | |
| ddd | | .12 | |
| e | | 0.80 BSC | |
| f | - | 0.60 | - |
| M | | 12 | |
| n | | 108 | |
| REF: JEDEC MO-219F | | | |

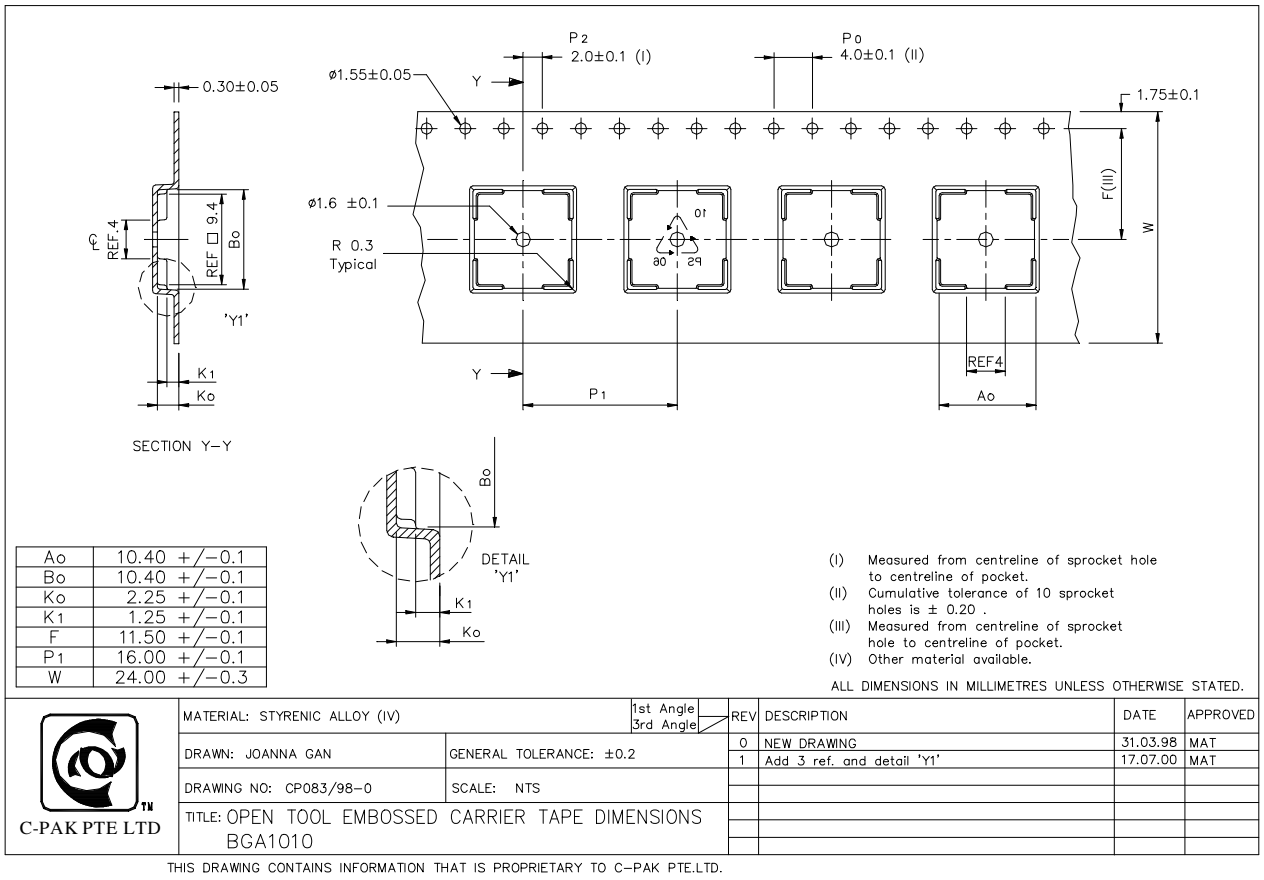
## C.2.2 Tray Dimensions

**Figure C-5. 108-Ball BGA Tray Dimensions**



## C.2.3 Tape and Reel Dimensions

**Figure C-6. 108-Ball BGA Tape and Reel Dimensions**




| | |
|---|---|
| Ao | 10.40 +/–0.1 |
| Bo | 10.40 +/–0.1 |
| Ko | 2.25 +/–0.1 |
| K1 | 1.25 +/–0.1 |
| F | 11.50 +/–0.1 |
| P1 | 16.00 +/–0.1 |
| W | 24.00 +/–0.3 |

## PACKAGING INFORMATION

| Orderable Device | Status (1) | Package Type | Package Drawing | Pins | Package Qty | Eco Plan (2) | Lead/ Ball Finish | MSL Peak Temp (3) | Samples (Requires Login) |
|---|---|---|---|---|---|---|---|---|---|
| LM3S1G58-IBZ80-A1 | ACTIVE | NFBGA | ZCR | 108 | 184 | TBD | Call TI | Call TI | |
| LM3S1G58-IBZ80-A1T | ACTIVE | NFBGA | ZCR | 108 | 1500 | TBD | Call TI | Call TI | |
| LM3S1G58-IQC80-A1 | ACTIVE | LQFP | PZ | 100 | 90 | TBD | Call TI | Call TI | |
| LM3S1G58-IQC80-A1T | ACTIVE | LQFP | PZ | 100 | 1000 | TBD | Call TI | Call TI | |

(1) The marketing status values are defined as follows:
**ACTIVE:** Product device recommended for new designs.
**LIFEBUY:** TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
**NRND:** Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
**PREVIEW:** Device has been announced but is not in production. Samples may or may not be available.
**OBSOLETE:** TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
**TBD:** The Pb-Free/Green conversion plan has not been defined.
**Pb-Free (RoHS):** TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
**Pb-Free (RoHS Exempt):** This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
**Green (RoHS & no Sb/Br):** TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

**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.

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Audio | www.ti.com/audio | Communications and Telecom | www.ti.com/communications |
| Amplifiers | amplifier.ti.com | Computers and Peripherals | www.ti.com/computers |
| Data Converters | dataconverter.ti.com | Consumer Electronics | www.ti.com/consumer-apps |
| DLP® Products | www.dlp.com | Energy and Lighting | www.ti.com/energy |
| DSP | dsp.ti.com | Industrial | www.ti.com/industrial |
| Clocks and Timers | www.ti.com/clocks | Medical | www.ti.com/medical |
| Interface | interface.ti.com | Security | www.ti.com/security |
| Logic | logic.ti.com | Space, Avionics and Defense | www.ti.com/space-avionics-defense |
| Power Mgmt | power.ti.com | Transportation and Automotive | www.ti.com/automotive |
| Microcontrollers | microcontroller.ti.com | Video and Imaging | www.ti.com/video |
| RFID | www.ti-rfid.com | | |
| OMAP Mobile Processors | www.ti.com/omap | | |
| Wireless Connctivity | www.ti.com/wirelessconnectivity | | |
| | **TI E2E Community Home Page** | | e2e.ti.com |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2011, Texas Instruments Incorporated

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View LM3S1G58-IBZ80-A1 on WIN SOURCE
- Texas Instruments Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management