Stellaris® LM3S8938 Microcontroller

DATA SHEET

DS-LM3S8938-15852.2743
SPMS174I

Copyright © 2007-2014
Texas Instruments Incorporated

Copyright

Copyright © 2007-2014 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 **25**
About This Document **32**
Audience 32
About This Manual 32
Related Documents 32
Documentation Conventions 33
1 Architectural Overview **35**
1.1 Product Features 35
1.2 Target Applications 43
1.3 High-Level Block Diagram 43
1.4 Functional Overview 45
1.4.1 ARM Cortex™-M3 45
1.4.2 Motor Control Peripherals 46
1.4.3 Analog Peripherals 46
1.4.4 Serial Communications Peripherals 47
1.4.5 System Peripherals 49
1.4.6 Memory Peripherals 49
1.4.7 Additional Features 50
1.4.8 Hardware Details 50
2 The Cortex-M3 Processor **52**
2.1 Block Diagram 53
2.2 Overview 54
2.2.1 System-Level Interface 54
2.2.2 Integrated Configurable Debug 54
2.2.3 Trace Port Interface Unit (TPIU) 55
2.2.4 Cortex-M3 System Component Details 55
2.3 Programming Model 56
2.3.1 Processor Mode and Privilege Levels for Software Execution 56
2.3.2 Stacks 56
2.3.3 Register Map 57
2.3.4 Register Descriptions 58
2.3.5 Exceptions and Interrupts 71
2.3.6 Data Types 71
2.4 Memory Model 71
2.4.1 Memory Regions, Types and Attributes 73
2.4.2 Memory System Ordering of Memory Accesses 73
2.4.3 Behavior of Memory Accesses 73
2.4.4 Software Ordering of Memory Accesses 74
2.4.5 Bit-Banding 75
2.4.6 Data Storage 77
2.4.7 Synchronization Primitives 78
2.5 Exception Model 79
2.5.1 Exception States 80
2.5.2 Exception Types 80
2.5.3 Exception Handlers 83

2.5.4 Vector Table 83
2.5.5 Exception Priorities 84
2.5.6 Interrupt Priority Grouping 85
2.5.7 Exception Entry and Return 85
2.6 Fault Handling 87
2.6.1 Fault Types 88
2.6.2 Fault Escalation and Hard Faults 88
2.6.3 Fault Status Registers and Fault Address Registers 89
2.6.4 Lockup 89
2.7 Power Management 89
2.7.1 Entering Sleep Modes 90
2.7.2 Wake Up from Sleep Mode 90
2.8 Instruction Set Summary 91
3 Cortex-M3 Peripherals 94
3.1 Functional Description 94
3.1.1 System Timer (SysTick) 94
3.1.2 Nested Vectored Interrupt Controller (NVIC) 95
3.1.3 System Control Block (SCB) 97
3.1.4 Memory Protection Unit (MPU) 97
3.2 Register Map 102
3.3 System Timer (SysTick) Register Descriptions 104
3.4 NVIC Register Descriptions 108
3.5 System Control Block (SCB) Register Descriptions 121
3.6 Memory Protection Unit (MPU) Register Descriptions 148
4 JTAG Interface 158
4.1 Block Diagram 159
4.2 Signal Description 159
4.3 Functional Description 160
4.3.1 JTAG Interface Pins 160
4.3.2 JTAG TAP Controller 162
4.3.3 Shift Registers 163
4.3.4 Operational Considerations 163
4.4 Initialization and Configuration 166
4.5 Register Descriptions 166
4.5.1 Instruction Register (IR) 166
4.5.2 Data Registers 169
5 System Control 171
5.1 Signal Description 171
5.2 Functional Description 171
5.2.1 Device Identification 172
5.2.2 Reset Control 172
5.2.3 Power Control 176
5.2.4 Clock Control 177
5.2.5 System Control 182
5.3 Initialization and Configuration 183
5.4 Register Map 184
5.5 Register Descriptions 185

6 Hibernation Module 238
6.1 Block Diagram 239
6.2 Signal Description 239
6.3 Functional Description 240
6.3.1 Register Access Timing 240
6.3.2 Clock Source 241
6.3.3 Battery Management 242
6.3.4 Real-Time Clock 242
6.3.5 Battery-Backed Memory 243
6.3.6 Power Control 243
6.3.7 Initiating Hibernate 243
6.3.8 Interrupts and Status 244
6.4 Initialization and Configuration 244
6.4.1 Initialization 244
6.4.2 RTC Match Functionality (No Hibernation) 244
6.4.3 RTC Match/Wake-Up from Hibernation 245
6.4.4 External Wake-Up from Hibernation 245
6.4.5 RTC/External Wake-Up from Hibernation 245
6.5 Register Map 245
6.6 Register Descriptions 246
7 Internal Memory 259
7.1 Block Diagram 259
7.2 Functional Description 259
7.2.1 SRAM Memory 259
7.2.2 Flash Memory 260
7.3 Flash Memory Initialization and Configuration 262
7.3.1 Flash Programming 262
7.3.2 Nonvolatile Register Programming 263
7.4 Register Map 264
7.5 Flash Register Descriptions (Flash Control Offset) 265
7.6 Flash Register Descriptions (System Control Offset) 273
8 General-Purpose Input/Outputs (GPIOs) 286
8.1 Signal Description 286
8.2 Functional Description 291
8.2.1 Data Control 292
8.2.2 Interrupt Control 293
8.2.3 Mode Control 294
8.2.4 Commit Control 294
8.2.5 Pad Control 294
8.2.6 Identification 295
8.3 Initialization and Configuration 295
8.4 Register Map 296
8.5 Register Descriptions 298
9 General-Purpose Timers 333
9.1 Block Diagram 334
9.2 Signal Description 335
9.3 Functional Description 335
9.3.1 GPTM Reset Conditions 335

9.3.2 32-Bit Timer Operating Modes 336
9.3.3 16-Bit Timer Operating Modes 337
9.4 Initialization and Configuration 341
9.4.1 32-Bit One-Shot/Periodic Timer Mode 341
9.4.2 32-Bit Real-Time Clock (RTC) Mode 342
9.4.3 16-Bit One-Shot/Periodic Timer Mode 342
9.4.4 16-Bit Input Edge Count Mode 343
9.4.5 16-Bit Input Edge Timing Mode 343
9.4.6 16-Bit PWM Mode 344
9.5 Register Map 344
9.6 Register Descriptions 345
10 Watchdog Timer 370
10.1 Block Diagram 371
10.2 Functional Description 371
10.3 Initialization and Configuration 372
10.4 Register Map 372
10.5 Register Descriptions 373
11 Analog-to-Digital Converter (ADC) 394
11.1 Block Diagram 394
11.2 Signal Description 395
11.3 Functional Description 396
11.3.1 Sample Sequencers 396
11.3.2 Module Control 397
11.3.3 Hardware Sample Averaging Circuit 398
11.3.4 Analog-to-Digital Converter 398
11.3.5 Differential Sampling 398
11.3.6 Test Modes 400
11.3.7 Internal Temperature Sensor 400
11.4 Initialization and Configuration 401
11.4.1 Module Initialization 401
11.4.2 Sample Sequencer Configuration 401
11.5 Register Map 402
11.6 Register Descriptions 403
12 Universal Asynchronous Receivers/Transmitters (UARTs) 430
12.1 Block Diagram 431
12.2 Signal Description 431
12.3 Functional Description 432
12.3.1 Transmit/Receive Logic 432
12.3.2 Baud-Rate Generation 433
12.3.3 Data Transmission 433
12.3.4 Serial IR (SIR) 434
12.3.5 FIFO Operation 435
12.3.6 Interrupts 435
12.3.7 Loopback Operation 436
12.3.8 IrDA SIR block 437
12.4 Initialization and Configuration 437
12.5 Register Map 438
12.6 Register Descriptions 439

13 Synchronous Serial Interface (SSI) 473
13.1 Block Diagram 473
13.2 Signal Description 473
13.3 Functional Description 474
13.3.1 Bit Rate Generation 474
13.3.2 FIFO Operation 475
13.3.3 Interrupts 475
13.3.4 Frame Formats 475
13.4 Initialization and Configuration 483
13.5 Register Map 484
13.6 Register Descriptions 485
14 Inter-Integrated Circuit (I^2C) Interface 511
14.1 Block Diagram 512
14.2 Signal Description 512
14.3 Functional Description 513
14.3.1 I^2C Bus Functional Overview 513
14.3.2 Available Speed Modes 515
14.3.3 Interrupts 516
14.3.4 Loopback Operation 517
14.3.5 Command Sequence Flow Charts 517
14.4 Initialization and Configuration 524
14.5 Register Map 525
14.6 Register Descriptions (I^2C Master) 526
14.7 Register Descriptions (I^2C Slave) 539
15 Controller Area Network (CAN) Module 548
15.1 Block Diagram 549
15.2 Signal Description 549
15.3 Functional Description 550
15.3.1 Initialization 551
15.3.2 Operation 551
15.3.3 Transmitting Message Objects 552
15.3.4 Configuring a Transmit Message Object 552
15.3.5 Updating a Transmit Message Object 554
15.3.6 Accepting Received Message Objects 554
15.3.7 Receiving a Data Frame 554
15.3.8 Receiving a Remote Frame 555
15.3.9 Receive/Transmit Priority 555
15.3.10 Configuring a Receive Message Object 556
15.3.11 Handling of Received Message Objects 557
15.3.12 Handling of Interrupts 560
15.3.13 Test Mode 560
15.3.14 Bit Timing Configuration Error Considerations 562
15.3.15 Bit Time and Bit Rate 562
15.3.16 Calculating the Bit Timing Parameters 564
15.4 Register Map 567
15.5 CAN Register Descriptions 569

16 Ethernet Controller 595
16.1 Block Diagram 595
16.2 Signal Description 596
16.3 Functional Description 598
16.3.1 MAC Operation 598
16.3.2 Internal MII Operation 601
16.3.3 PHY Operation 601
16.3.4 Interrupts 603
16.4 Initialization and Configuration 603
16.4.1 Hardware Configuration 603
16.4.2 Software Configuration 604
16.5 Ethernet Register Map 605
16.6 Ethernet MAC Register Descriptions 606
16.7 MII Management Register Descriptions 625
17 Analog Comparators 644
17.1 Block Diagram 645
17.2 Signal Description 645
17.3 Functional Description 646
17.3.1 Internal Reference Programming 647
17.4 Initialization and Configuration 648
17.5 Register Map 648
17.6 Register Descriptions 649
18 Pin Diagram 657
19 Signal Tables 659
19.1 100-Pin LQFP Package Pin Tables 659
19.1.1 Signals by Pin Number 659
19.1.2 Signals by Signal Name 663
19.1.3 Signals by Function, Except for GPIO 667
19.1.4 GPIO Pins and Alternate Functions 671
19.2 108-Ball BGA Package Pin Tables 672
19.2.1 Signals by Pin Number 672
19.2.2 Signals by Signal Name 677
19.2.3 Signals by Function, Except for GPIO 681
19.2.4 GPIO Pins and Alternate Functions 684
19.3 Connections for Unused Signals 685
20 Operating Characteristics 688
21 Electrical Characteristics 689
21.1 DC Characteristics 689
21.1.1 Maximum Ratings 689
21.1.2 Recommended DC Operating Conditions 689
21.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics 690
21.1.4 GPIO Module Characteristics 690
21.1.5 Power Specifications 690
21.1.6 Flash Memory Characteristics 692
21.1.7 Hibernation 692
21.1.8 Ethernet Controller 692
21.2 AC Characteristics 692

21.2.1 Load Conditions 692
21.2.2 Clocks 693
21.2.3 JTAG and Boundary Scan 694
21.2.4 Reset 696
21.2.5 Sleep Modes 698
21.2.6 Hibernation Module 698
21.2.7 General-Purpose I/O (GPIO) 698
21.2.8 Analog-to-Digital Converter 699
21.2.9 Synchronous Serial Interface (SSI) 700
21.2.10 Inter-Integrated Circuit (I^2C) Interface 702
21.2.11 Ethernet Controller 703
21.2.12 Analog Comparator 705

A Serial Flash Loader 707
A.1 Serial Flash Loader 707
A.2 Interfaces 707
A.2.1 UART 707
A.2.2 SSI 707
A.3 Packet Handling 708
A.3.1 Packet Format 708
A.3.2 Sending Packets 708
A.3.3 Receiving Packets 708
A.4 Commands 709
A.4.1 COMMAND_PING (0X20) 709
A.4.2 COMMAND_GET_STATUS (0x23) 709
A.4.3 COMMAND_DOWNLOAD (0x21) 709
A.4.4 COMMAND_SEND_DATA (0x24) 710
A.4.5 COMMAND_RUN (0x22) 710
A.4.6 COMMAND_RESET (0x25) 710

B Register Quick Reference 712

C Ordering and Contact Information 735
C.1 Ordering Information 735
C.2 Part Markings 735
C.3 Kits 735
C.4 Support Information 736

D Package Information 737
D.1 100-Pin LQFP Package 737
D.1.1 Package Dimensions 737
D.1.2 Tray Dimensions 739
D.1.3 Tape and Reel Dimensions 739
D.2 108-Ball BGA Package 741
D.2.1 Package Dimensions 741
D.2.2 Tray Dimensions 743
D.2.3 Tape and Reel Dimensions 744

List of Figures

Figure 1-1. Stellaris LM3S8938 Microcontroller High-Level Block Diagram 44
Figure 2-1. CPU Block Diagram 54
Figure 2-2. TPIU Block Diagram 55
Figure 2-3. Cortex-M3 Register Set 57
Figure 2-4. Bit-Band Mapping 77
Figure 2-5. Data Storage 78
Figure 2-6. Vector Table 84
Figure 2-7. Exception Stack Frame 86
Figure 3-1. SRD Use Example 100
Figure 4-1. JTAG Module Block Diagram 159
Figure 4-2. Test Access Port State Machine 163
Figure 4-3. IDCODE Register Format 169
Figure 4-4. BYPASS Register Format 169
Figure 4-5. Boundary Scan Register Format 170
Figure 5-1. Basic $\overline{RST}$ Configuration 173
Figure 5-2. External Circuitry to Extend Power-On Reset 174
Figure 5-3. Reset Circuit Controlled by Switch 174
Figure 5-4. Power Architecture 177
Figure 5-5. Main Clock Tree 179
Figure 6-1. Hibernation Module Block Diagram 239
Figure 6-2. Clock Source Using Crystal 241
Figure 6-3. Clock Source Using Dedicated Oscillator 242
Figure 7-1. Flash Block Diagram 259
Figure 8-1. GPIO Port Block Diagram 292
Figure 8-2. GPIODATA Write Example 293
Figure 8-3. GPIODATA Read Example 293
Figure 9-1. GPTM Module Block Diagram 334
Figure 9-2. 16-Bit Input Edge Count Mode Example 339
Figure 9-3. 16-Bit Input Edge Time Mode Example 340
Figure 9-4. 16-Bit PWM Mode Example 341
Figure 10-1. WDT Module Block Diagram 371
Figure 11-1. ADC Module Block Diagram 395
Figure 11-2. Differential Sampling Range, V_{IN_ODD} = 1.5 V 399
Figure 11-3. Differential Sampling Range, V_{IN_ODD} = 0.75 V 399
Figure 11-4. Differential Sampling Range, V_{IN_ODD} = 2.25 V 400
Figure 11-5. Internal Temperature Sensor Characteristic 401
Figure 12-1. UART Module Block Diagram 431
Figure 12-2. UART Character Frame 433
Figure 12-3. IrDA Data Modulation 435
Figure 13-1. SSI Module Block Diagram 473
Figure 13-2. TI Synchronous Serial Frame Format (Single Transfer) 476
Figure 13-3. TI Synchronous Serial Frame Format (Continuous Transfer) 477
Figure 13-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 477
Figure 13-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 478
Figure 13-6. Freescale SPI Frame Format with SPO=0 and SPH=1 479
Figure 13-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 479

Figure 13-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 480
Figure 13-9. Freescale SPI Frame Format with SPO=1 and SPH=1 481
Figure 13-10. MICROWIRE Frame Format (Single Frame) 481
Figure 13-11. MICROWIRE Frame Format (Continuous Transfer) 482
Figure 13-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements 483
Figure 14-1. I2C Block Diagram 512
Figure 14-2. I2C Bus Configuration 513
Figure 14-3. START and STOP Conditions 513
Figure 14-4. Complete Data Transfer with a 7-Bit Address 514
Figure 14-5. R/S Bit in First Byte 514
Figure 14-6. Data Validity During Bit Transfer on the I2C Bus 514
Figure 14-7. Master Single SEND 518
Figure 14-8. Master Single RECEIVE 519
Figure 14-9. Master Burst SEND 520
Figure 14-10. Master Burst RECEIVE 521
Figure 14-11. Master Burst RECEIVE after Burst SEND 522
Figure 14-12. Master Burst SEND after Burst RECEIVE 523
Figure 14-13. Slave Command Sequence 524
Figure 15-1. CAN Controller Block Diagram 549
Figure 15-2. CAN Data/Remote Frame 550
Figure 15-3. Message Objects in a FIFO Buffer 559
Figure 15-4. CAN Bit Time 563
Figure 16-1. Ethernet Controller 596
Figure 16-2. Ethernet Controller Block Diagram 596
Figure 16-3. Ethernet Frame 598
Figure 16-4. Interface to an Ethernet Jack 604
Figure 17-1. Analog Comparator Module Block Diagram 645
Figure 17-2. Structure of Comparator Unit 647
Figure 17-3. Comparator Internal Reference Structure 647
Figure 18-1. 100-Pin LQFP Package Pin Diagram 657
Figure 18-2. 108-Ball BGA Package Pin Diagram (Top View) 658
Figure 21-1. Load Conditions 693
Figure 21-2. JTAG Test Clock Input Timing 695
Figure 21-3. JTAG Test Access Port (TAP) Timing 695
Figure 21-4. JTAG TRST Timing 696
Figure 21-5. External Reset Timing ($\overline{RST}$) 696
Figure 21-6. Power-On Reset Timing 697
Figure 21-7. Brown-Out Reset Timing 697
Figure 21-8. Software Reset Timing 697
Figure 21-9. Watchdog Reset Timing 697
Figure 21-10. Hibernation Module Timing 698
Figure 21-11. ADC Input Equivalency Diagram 700
Figure 21-12. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement 701
Figure 21-13. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer 701
Figure 21-14. SSI Timing for SPI Frame Format (FRF=00), with SPH=1 702
Figure 21-15. I2C Timing 703
Figure 21-16. External XTLP Oscillator Characteristics 705

Figure D-1. Stellaris LM3S8938 100-Pin LQFP Package Dimensions 737
Figure D-2. 100-Pin LQFP Tray Dimensions 739
Figure D-3. 100-Pin LQFP Tape and Reel Dimensions 740
Figure D-4. Stellaris LM3S8938 108-Ball BGA Package Dimensions 741
Figure D-5. 108-Ball BGA Tray Dimensions 743
Figure D-6. 108-Ball BGA Tape and Reel Dimensions 744

List of Tables

Table 1. Revision History 25
Table 2. Documentation Conventions 33
Table 2-1. Summary of Processor Mode, Privilege Level, and Stack Use 57
Table 2-2. Processor Register Map 58
Table 2-3. PSR Register Combinations 63
Table 2-4. Memory Map 71
Table 2-5. Memory Access Behavior 73
Table 2-6. SRAM Memory Bit-Banding Regions 75
Table 2-7. Peripheral Memory Bit-Banding Regions 76
Table 2-8. Exception Types 81
Table 2-9. Interrupts 82
Table 2-10. Exception Return Behavior 87
Table 2-11. Faults 88
Table 2-12. Fault Status and Fault Address Registers 89
Table 2-13. Cortex-M3 Instruction Summary 91
Table 3-1. Core Peripheral Register Regions 94
Table 3-2. Memory Attributes Summary 97
Table 3-3. TEX, S, C, and B Bit Field Encoding 100
Table 3-4. Cache Policy for Memory Attribute Encoding 101
Table 3-5. AP Bit Field Encoding 101
Table 3-6. Memory Region Attributes for Stellaris Microcontrollers 101
Table 3-7. Peripherals Register Map 102
Table 3-8. Interrupt Priority Levels 127
Table 3-9. Example SIZE Field Values 155
Table 4-1. JTAG_SWD_SWO Signals (100LQFP) 159
Table 4-2. JTAG_SWD_SWO Signals (108BGA) 160
Table 4-3. JTAG Port Pins Reset State 160
Table 4-4. JTAG Instruction Register Commands 167
Table 5-1. System Control & Clocks Signals (100LQFP) 171
Table 5-2. System Control & Clocks Signals (108BGA) 171
Table 5-3. Reset Sources 172
Table 5-4. Clock Source Options 178
Table 5-5. Possible System Clock Frequencies Using the SYSDIV Field 180
Table 5-6. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field 180
Table 5-7. System Control Register Map 184
Table 5-8. RCC2 Fields that Override RCC fields 199
Table 6-1. Hibernate Signals (100LQFP) 239
Table 6-2. Hibernate Signals (108BGA) 240
Table 6-3. Hibernation Module Register Map 246
Table 7-1. Flash Protection Policy Combinations 260
Table 7-2. User-Programmable Flash Memory Resident Registers 264
Table 7-3. Flash Register Map 264
Table 8-1. GPIO Pins With Non-Zero Reset Values 287
Table 8-2. GPIO Pins and Alternate Functions (100LQFP) 287
Table 8-3. GPIO Pins and Alternate Functions (108BGA) 288
Table 8-4. GPIO Signals (100LQFP) 289

Table 8-5. GPIO Signals (108BGA) 290
Table 8-6. GPIO Pad Configuration Examples 295
Table 8-7. GPIO Interrupt Configuration Example 295
Table 8-8. GPIO Register Map 297
Table 9-1. Available CCP Pins 334
Table 9-2. General-Purpose Timers Signals (100LQFP) 335
Table 9-3. General-Purpose Timers Signals (108BGA) 335
Table 9-4. 16-Bit Timer With Prescaler Configurations 338
Table 9-5. Timers Register Map 344
Table 10-1. Watchdog Timer Register Map 372
Table 11-1. ADC Signals (100LQFP) 395
Table 11-2. ADC Signals (108BGA) 395
Table 11-3. Samples and FIFO Depth of Sequencers 396
Table 11-4. Differential Sampling Pairs 398
Table 11-5. ADC Register Map 402
Table 12-1. UART Signals (100LQFP) 431
Table 12-2. UART Signals (108BGA) 432
Table 12-3. UART Register Map 438
Table 13-1. SSI Signals (100LQFP) 474
Table 13-2. SSI Signals (108BGA) 474
Table 13-3. SSI Register Map 484
Table 14-1. I2C Signals (100LQFP) 512
Table 14-2. I2C Signals (108BGA) 512
Table 14-3. Examples of I^2C Master Timer Period versus Speed Mode 515
Table 14-4. Inter-Integrated Circuit (I^2C) Interface Register Map 525
Table 14-5. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3) 530
Table 15-1. Controller Area Network Signals (100LQFP) 550
Table 15-2. Controller Area Network Signals (108BGA) 550
Table 15-3. CAN Protocol Ranges 563
Table 15-4. CANBIT Register Values 563
Table 15-5. CAN Register Map 567
Table 16-1. Ethernet Signals (100LQFP) 597
Table 16-2. Ethernet Signals (108BGA) 597
Table 16-3. TX & RX FIFO Organization 600
Table 16-4. Ethernet Register Map 605
Table 17-1. Analog Comparators Signals (100LQFP) 645
Table 17-2. Analog Comparators Signals (108BGA) 646
Table 17-3. Internal Reference Voltage and ACREFCTL Field Values 647
Table 17-4. Analog Comparators Register Map 649
Table 19-1. Signals by Pin Number 659
Table 19-2. Signals by Signal Name 663
Table 19-3. Signals by Function, Except for GPIO 667
Table 19-4. GPIO Pins and Alternate Functions 671
Table 19-5. Signals by Pin Number 672
Table 19-6. Signals by Signal Name 677
Table 19-7. Signals by Function, Except for GPIO 681
Table 19-8. GPIO Pins and Alternate Functions 684
Table 19-9. Connections for Unused Signals (100-pin LQFP) 685

Table 19-10. Connections for Unused Signals, 108-pin BGA 687
Table 20-1. Temperature Characteristics 688
Table 20-2. Thermal Characteristics 688
Table 20-3. ESD Absolute Maximum Ratings 688
Table 21-1. Maximum Ratings 689
Table 21-2. Recommended DC Operating Conditions 689
Table 21-3. LDO Regulator Characteristics 690
Table 21-4. GPIO Module DC Characteristics 690
Table 21-5. Detailed Power Specifications 691
Table 21-6. Flash Memory Characteristics 692
Table 21-7. Hibernation Module DC Characteristics 692
Table 21-8. Ethernet Controller DC Characteristics 692
Table 21-9. Phase Locked Loop (PLL) Characteristics 693
Table 21-10. Actual PLL Frequency 693
Table 21-11. Clock Characteristics 693
Table 21-12. Crystal Characteristics 694
Table 21-13. System Clock Characteristics with ADC Operation 694
Table 21-14. JTAG Characteristics 694
Table 21-15. Reset Characteristics 696
Table 21-16. Sleep Modes AC Characteristics 698
Table 21-17. Hibernation Module AC Characteristics 698
Table 21-18. GPIO Characteristics 699
Table 21-19. ADC Characteristics 699
Table 21-20. ADC Module Internal Reference Characteristics 700
Table 21-21. SSI Characteristics 700
Table 21-22. I^2C Characteristics 702
Table 21-23. 100BASE-TX Transmitter Characteristics 703
Table 21-24. 100BASE-TX Transmitter Characteristics (informative) 703
Table 21-25. 100BASE-TX Receiver Characteristics 703
Table 21-26. 10BASE-T Transmitter Characteristics 703
Table 21-27. 10BASE-T Transmitter Characteristics (informative) 704
Table 21-28. 10BASE-T Receiver Characteristics 704
Table 21-29. Isolation Transformers 704
Table 21-30. Ethernet Reference Crystal 704
Table 21-31. External XTLP Oscillator Characteristics 705
Table 21-32. Analog Comparator Characteristics 705
Table 21-33. Analog Comparator Voltage Reference Characteristics 706

List of Registers

The Cortex-M3 Processor 52
Register 1: Cortex General-Purpose Register 0 (R0) 59
Register 2: Cortex General-Purpose Register 1 (R1) 59
Register 3: Cortex General-Purpose Register 2 (R2) 59
Register 4: Cortex General-Purpose Register 3 (R3) 59
Register 5: Cortex General-Purpose Register 4 (R4) 59
Register 6: Cortex General-Purpose Register 5 (R5) 59
Register 7: Cortex General-Purpose Register 6 (R6) 59
Register 8: Cortex General-Purpose Register 7 (R7) 59
Register 9: Cortex General-Purpose Register 8 (R8) 59
Register 10: Cortex General-Purpose Register 9 (R9) 59
Register 11: Cortex General-Purpose Register 10 (R10) 59
Register 12: Cortex General-Purpose Register 11 (R11) 59
Register 13: Cortex General-Purpose Register 12 (R12) 59
Register 14: Stack Pointer (SP) 60
Register 15: Link Register (LR) 61
Register 16: Program Counter (PC) 62
Register 17: Program Status Register (PSR) 63
Register 18: Priority Mask Register (PRIMASK) 67
Register 19: Fault Mask Register (FAULTMASK) 68
Register 20: Base Priority Mask Register (BASEPRI) 69
Register 21: Control Register (CONTROL) 70
Cortex-M3 Peripherals 94
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010 105
Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014 107
Register 3: SysTick Current Value Register (STCURRENT), offset 0x018 108
Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100 109
Register 5: Interrupt 32-43 Set Enable (EN1), offset 0x104 110
Register 6: Interrupt 0-31 Clear Enable (DIS0), offset 0x180 111
Register 7: Interrupt 32-43 Clear Enable (DIS1), offset 0x184 112
Register 8: Interrupt 0-31 Set Pending (PEND0), offset 0x200 113
Register 9: Interrupt 32-43 Set Pending (PEND1), offset 0x204 114
Register 10: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280 115
Register 11: Interrupt 32-43 Clear Pending (UNPEND1), offset 0x284 116
Register 12: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300 117
Register 13: Interrupt 32-43 Active Bit (ACTIVE1), offset 0x304 118
Register 14: Interrupt 0-3 Priority (PRI0), offset 0x400 119
Register 15: Interrupt 4-7 Priority (PRI1), offset 0x404 119
Register 16: Interrupt 8-11 Priority (PRI2), offset 0x408 119
Register 17: Interrupt 12-15 Priority (PRI3), offset 0x40C 119
Register 18: Interrupt 16-19 Priority (PRI4), offset 0x410 119
Register 19: Interrupt 20-23 Priority (PRI5), offset 0x414 119
Register 20: Interrupt 24-27 Priority (PRI6), offset 0x418 119
Register 21: Interrupt 28-31 Priority (PRI7), offset 0x41C 119
Register 22: Interrupt 32-35 Priority (PRI8), offset 0x420 119

Register 23: Interrupt 36-39 Priority (PRI9), offset 0x424 119
Register 24: Interrupt 40-43 Priority (PRI10), offset 0x428 119
Register 25: Software Trigger Interrupt (SWTRIG), offset 0xF00 121
Register 26: CPU ID Base (CPUID), offset 0xD00 122
Register 27: Interrupt Control and State (INTCTRL), offset 0xD04 123
Register 28: Vector Table Offset (VTABLE), offset 0xD08 126
Register 29: Application Interrupt and Reset Control (APINT), offset 0xD0C 127
Register 30: System Control (SYSCTRL), offset 0xD10 129
Register 31: Configuration and Control (CFGCTRL), offset 0xD14 131
Register 32: System Handler Priority 1 (SYSPRI1), offset 0xD18 133
Register 33: System Handler Priority 2 (SYSPRI2), offset 0xD1C 134
Register 34: System Handler Priority 3 (SYSPRI3), offset 0xD20 135
Register 35: System Handler Control and State (SYSHNDCTRL), offset 0xD24 136
Register 36: Configurable Fault Status (FAULTSTAT), offset 0xD28 140
Register 37: Hard Fault Status (HFAULTSTAT), offset 0xD2C 146
Register 38: Memory Management Fault Address (MMADDR), offset 0xD34 147
Register 39: Bus Fault Address (FAULTADDR), offset 0xD38 148
Register 40: MPU Type (MPUTYPE), offset 0xD90 149
Register 41: MPU Control (MPUCTRL), offset 0xD94 150
Register 42: MPU Region Number (MPUNUMBER), offset 0xD98 152
Register 43: MPU Region Base Address (MPUBASE), offset 0xD9C 153
Register 44: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4 153
Register 45: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC 153
Register 46: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4 153
Register 47: MPU Region Attribute and Size (MPUATTR), offset 0xDA0 155
Register 48: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8 155
Register 49: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0 155
Register 50: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8 155

System Control 171

Register 1: Device Identification 0 (DID0), offset 0x000 186
Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030 188
Register 3: LDO Power Control (LDOPCTL), offset 0x034 189
Register 4: Raw Interrupt Status (RIS), offset 0x050 190
Register 5: Interrupt Mask Control (IMC), offset 0x054 191
Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058 192
Register 7: Reset Cause (RESC), offset 0x05C 193
Register 8: Run-Mode Clock Configuration (RCC), offset 0x060 194
Register 9: XTAL to PLL Translation (PLLCFG), offset 0x064 198
Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070 199
Register 11: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144 201
Register 12: Device Identification 1 (DID1), offset 0x004 202
Register 13: Device Capabilities 0 (DC0), offset 0x008 204
Register 14: Device Capabilities 1 (DC1), offset 0x010 205
Register 15: Device Capabilities 2 (DC2), offset 0x014 207
Register 16: Device Capabilities 3 (DC3), offset 0x018 209
Register 17: Device Capabilities 4 (DC4), offset 0x01C 211
Register 18: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 213
Register 19: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 215

Register 20: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 217
Register 21: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 219
Register 22: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 222
Register 23: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 225
Register 24: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 228
Register 25: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 230
Register 26: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 232
Register 27: Software Reset Control 0 (SRCR0), offset 0x040 234
Register 28: Software Reset Control 1 (SRCR1), offset 0x044 235
Register 29: Software Reset Control 2 (SRCR2), offset 0x048 237

Hibernation Module 238
Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000 247
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004 248
Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008 249
Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C 250
Register 5: Hibernation Control (HIBCTL), offset 0x010 251
Register 6: Hibernation Interrupt Mask (HIBIM), offset 0x014 253
Register 7: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018 254
Register 8: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C 255
Register 9: Hibernation Interrupt Clear (HIBIC), offset 0x020 256
Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024 257
Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C 258

Internal Memory 259
Register 1: Flash Memory Address (FMA), offset 0x000 266
Register 2: Flash Memory Data (FMD), offset 0x004 267
Register 3: Flash Memory Control (FMC), offset 0x008 268
Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C 270
Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010 271
Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 272
Register 7: USec Reload (USECRL), offset 0x140 274
Register 8: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200 275
Register 9: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400 276
Register 10: User Debug (USER_DBG), offset 0x1D0 277
Register 11: User Register 0 (USER_REG0), offset 0x1E0 278
Register 12: User Register 1 (USER_REG1), offset 0x1E4 279
Register 13: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204 280
Register 14: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208 281
Register 15: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C 282
Register 16: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404 283
Register 17: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408 284
Register 18: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C 285

General-Purpose Input/Outputs (GPIOs) 286
Register 1: GPIO Data (GPIODATA), offset 0x000 299
Register 2: GPIO Direction (GPIODIR), offset 0x400 300
Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404 301
Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 302
Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C 303
Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410 304

Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414 305
Register 8: GPIO Masked Interrupt Status (GPIOMIS), offset 0x418 306
Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C 307
Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 308
Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 310
Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 311
Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 312
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C 313
Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510 314
Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514 315
Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 316
Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C 317
Register 19: GPIO Lock (GPIOLOCK), offset 0x520 318
Register 20: GPIO Commit (GPIOCR), offset 0x524 319
Register 21: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 321
Register 22: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 322
Register 23: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 323
Register 24: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC 324
Register 25: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 325
Register 26: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 326
Register 27: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 327
Register 28: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC 328
Register 29: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 329
Register 30: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 330
Register 31: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 331
Register 32: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC 332

General-Purpose Timers 333

Register 1: GPTM Configuration (GPTMCFG), offset 0x000 346
Register 2: GPTM TimerA Mode (GPTMTAMR), offset 0x004 347
Register 3: GPTM TimerB Mode (GPTMTBMR), offset 0x008 349
Register 4: GPTM Control (GPTMCTL), offset 0x00C 351
Register 5: GPTM Interrupt Mask (GPTMIMR), offset 0x018 354
Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C 356
Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 357
Register 8: GPTM Interrupt Clear (GPTMICR), offset 0x024 358
Register 9: GPTM TimerA Interval Load (GPTMTAILR), offset 0x028 360
Register 10: GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C 361
Register 11: GPTM TimerA Match (GPTMTAMATCHR), offset 0x030 362
Register 12: GPTM TimerB Match (GPTMTBMATCHR), offset 0x034 363
Register 13: GPTM TimerA Prescale (GPTMTAPR), offset 0x038 364
Register 14: GPTM TimerB Prescale (GPTMTBPR), offset 0x03C 365
Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040 366
Register 16: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044 367
Register 17: GPTM TimerA (GPTMTAR), offset 0x048 368
Register 18: GPTM TimerB (GPTMTBR), offset 0x04C 369

Watchdog Timer 370

Register 1: Watchdog Load (WDTLOAD), offset 0x000 374
Register 2: Watchdog Value (WDTVALUE), offset 0x004 375

Register 3: Watchdog Control (WDTCTL), offset 0x008 376
Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C 377
Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 378
Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 379
Register 7: Watchdog Test (WDTTEST), offset 0x418 380
Register 8: Watchdog Lock (WDTLOCK), offset 0xC00 381
Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 382
Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 383
Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 384
Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC 385
Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 386
Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 387
Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 388
Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC 389
Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 390
Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 391
Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 392
Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC 393

Analog-to-Digital Converter (ADC) 394

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000 404
Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004 405
Register 3: ADC Interrupt Mask (ADCIM), offset 0x008 406
Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C 407
Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010 408
Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014 409
Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018 412
Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 413
Register 9: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 415
Register 10: ADC Sample Averaging Control (ADCSAC), offset 0x030 416
Register 11: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 417
Register 12: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 419
Register 13: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 422
Register 14: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 422
Register 15: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 422
Register 16: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 422
Register 17: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C 423
Register 18: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C 423
Register 19: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C 423
Register 20: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC 423
Register 21: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 424
Register 22: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 424
Register 23: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 425
Register 24: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 425
Register 25: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 427
Register 26: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 428
Register 27: ADC Test Mode Loopback (ADCTMLB), offset 0x100 429

Universal Asynchronous Receivers/Transmitters (UARTs) 430

Register 1: UART Data (UARTDR), offset 0x000 440

Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 442
Register 3: UART Flag (UARTFR), offset 0x018 444
Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020 446
Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 447
Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 448
Register 7: UART Line Control (UARTLCRH), offset 0x02C 449
Register 8: UART Control (UARTCTL), offset 0x030 451
Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 453
Register 10: UART Interrupt Mask (UARTIM), offset 0x038 455
Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C 457
Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040 458
Register 13: UART Interrupt Clear (UARTICR), offset 0x044 459
Register 14: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 461
Register 15: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 462
Register 16: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 463
Register 17: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC 464
Register 18: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 465
Register 19: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 466
Register 20: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 467
Register 21: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC 468
Register 22: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 469
Register 23: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 470
Register 24: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 471
Register 25: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC 472

Synchronous Serial Interface (SSI) 473

Register 1: SSI Control 0 (SSICR0), offset 0x000 486
Register 2: SSI Control 1 (SSICR1), offset 0x004 488
Register 3: SSI Data (SSIDR), offset 0x008 490
Register 4: SSI Status (SSISR), offset 0x00C 491
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010 493
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014 494
Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018 496
Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C 497
Register 9: SSI Interrupt Clear (SSIICR), offset 0x020 498
Register 10: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 499
Register 11: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 500
Register 12: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 501
Register 13: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC 502
Register 14: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 503
Register 15: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 504
Register 16: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 505
Register 17: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC 506
Register 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 507
Register 19: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 508
Register 20: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 509
Register 21: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC 510

Inter-Integrated Circuit (I^2C) Interface 511

Register 1: I^2C Master Slave Address (I2CMSA), offset 0x000 527

Register 2: I²C Master Control/Status (I2CMCS), offset 0x004 528
Register 3: I²C Master Data (I2CMDR), offset 0x008 532
Register 4: I²C Master Timer Period (I2CMTPR), offset 0x00C 533
Register 5: I²C Master Interrupt Mask (I2CMIMR), offset 0x010 534
Register 6: I²C Master Raw Interrupt Status (I2CMRIS), offset 0x014 535
Register 7: I²C Master Masked Interrupt Status (I2CMMIS), offset 0x018 536
Register 8: I²C Master Interrupt Clear (I2CMICR), offset 0x01C 537
Register 9: I²C Master Configuration (I2CMCR), offset 0x020 538
Register 10: I²C Slave Own Address (I2CSOAR), offset 0x800 540
Register 11: I²C Slave Control/Status (I2CSCSR), offset 0x804 541
Register 12: I²C Slave Data (I2CSDR), offset 0x808 543
Register 13: I²C Slave Interrupt Mask (I2CSIMR), offset 0x80C 544
Register 14: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x810 545
Register 15: I²C Slave Masked Interrupt Status (I2CSMIS), offset 0x814 546
Register 16: I²C Slave Interrupt Clear (I2CSICR), offset 0x818 547

Controller Area Network (CAN) Module 548

Register 1: CAN Control (CANCTL), offset 0x000 570
Register 2: CAN Status (CANSTS), offset 0x004 572
Register 3: CAN Error Counter (CANERR), offset 0x008 574
Register 4: CAN Bit Timing (CANBIT), offset 0x00C 575
Register 5: CAN Interrupt (CANINT), offset 0x010 576
Register 6: CAN Test (CANTST), offset 0x014 577
Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018 579
Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020 580
Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080 580
Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024 581
Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084 581
Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028 583
Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088 583
Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C 584
Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C 584
Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030 585
Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090 585
Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034 586
Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094 586
Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038 588
Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098 588
Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C 590
Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040 590
Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044 590
Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048 590
Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C 590
Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0 590
Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4 590
Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8 590
Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100 591
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104 591

Register 32: CAN New Data 1 (CANNWDA1), offset 0x120 592
Register 33: CAN New Data 2 (CANNWDA2), offset 0x124 592
Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140 593
Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144 593
Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160 594
Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164 594

Ethernet Controller 595

Register 1: Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK), offset 0x000 607
Register 2: Ethernet MAC Interrupt Mask (MACIM), offset 0x004 610
Register 3: Ethernet MAC Receive Control (MACRCTL), offset 0x008 611
Register 4: Ethernet MAC Transmit Control (MACTCTL), offset 0x00C 612
Register 5: Ethernet MAC Data (MACDATA), offset 0x010 613
Register 6: Ethernet MAC Individual Address 0 (MACIA0), offset 0x014 615
Register 7: Ethernet MAC Individual Address 1 (MACIA1), offset 0x018 616
Register 8: Ethernet MAC Threshold (MACTHR), offset 0x01C 617
Register 9: Ethernet MAC Management Control (MACMCTL), offset 0x020 619
Register 10: Ethernet MAC Management Divider (MACMDV), offset 0x024 620
Register 11: Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C 621
Register 12: Ethernet MAC Management Receive Data (MACMRXD), offset 0x030 622
Register 13: Ethernet MAC Number of Packets (MACNP), offset 0x034 623
Register 14: Ethernet MAC Transmission Request (MACTR), offset 0x038 624
Register 15: Ethernet MAC Timer Support (MACTS), offset 0x03C 625
Register 16: Ethernet PHY Management Register 0 – Control (MR0), address 0x00 626
Register 17: Ethernet PHY Management Register 1 – Status (MR1), address 0x01 628
Register 18: Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02 630
Register 19: Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03 631
Register 20: Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04 632
Register 21: Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05 634
Register 22: Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06 635
Register 23: Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10 636
Register 24: Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17), address 0x11 638
Register 25: Ethernet PHY Management Register 18 – Diagnostic (MR18), address 0x12 640
Register 26: Ethernet PHY Management Register 19 – Transceiver Control (MR19), address 0x13 641
Register 27: Ethernet PHY Management Register 23 – LED Configuration (MR23), address 0x17 642
Register 28: Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24), address 0x18 643

Analog Comparators 644

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000 650
Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004 651
Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008 652
Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010 653
Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020 654
Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040 654
Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x060 654
Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x024 655

Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044 655
Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064 655

Revision History

The revision history table notes changes made between the indicated revisions of the LM3S8938 data sheet.

Table 1. Revision History

Date	Revision	Description
July 2014	.	■ In JTAG chapter, clarified JTAG-to-SWD Switching and SWD-to-JTAG Switching. ■ In System Control chapter, clarified behavior of **Reset Cause (RESC)** register external reset bit. ■ In Internal Memory chapter: – Added sections on Execute-Only Protection, Read-Only Protection, and Permanently Disabling Debug. – Noted that the **Boot Configuration (BOOTCFG)** register requires a POR before committed changes to the Flash-resident registers take effect. ■ In UART chapter: – Clarified that the transmit interrupt is based on a transition through level. – Corrected reset for **UART Raw Interrupt Status (UARTRIS)** register. ■ In Electrical Characteristics chapter, updated Crystal Characteristics and Ethernet Reference Crystal tables. ■ In Ordering and Contact Information appendix, moved orderable part numbers table to addendum. ■ Additional minor data sheet clarifications and corrections.
June 2012	12746.2515	■ Minor data sheet clarifications and corrections.
November 2011	11108	■ Added module-specific pin tables to each chapter in the new Signal Description sections. ■ In Hibernation chapter: – Changed terminology from non-volatile memory to battery-backed memory. – Clarified Hibernation module register reset conditions. ■ In Timer chapter, clarified that in 16-Bit Input Edge Time Mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. ■ In UART chapter, clarified interrupt behavior. ■ In SSI chapter, corrected SSIClk in the figure "Synchronous Serial Frame Format (Single Transfer)". ■ In Signal Tables chapter: – Corrected pin numbers in table "Connections for Unused Signals" (other pin tables were correct). ■ In Electrical Characteristics chapter: – Added parameter "Input voltage for a GPIO configured as an analog input" to the "Maximum Ratings" table. – Corrected Nom values for parameters "TCK clock Low time" and "TCK clock High time" in "JTAG Characteristics" table. – Corrected missing values for "Conversion time" and "Conversion rate" parameters in "ADC Characteristics" table. ■ Additional minor data sheet clarifications and corrections.

Table 1. Revision History *(continued)*

Date	Revision	Description
January 2011	9102	■ In **Application Interrupt and Reset Control (APINT)** register, changed bit name from SYSRESETREQ to SYSRESREQ. ■ Added DEBUG (Debug Priority) bit field to **System Handler Priority 3 (SYSPRI3)** register. ■ Added "Reset Sources" table to System Control chapter. ■ Removed mention of false-start bit detection in the UART chapter. This feature is not supported. ■ Added note that specific module clocks must be enabled before that module's registers can be programmed. There must be a delay of 3 system clocks after the module clock is enabled before any of that module's registers are accessed. ■ Changed I^2C slave register base addresses and offsets to be relative to the I^2C module base address of 0x4002.0000 and 0x4002.1000, so register bases and offsets were changed for all I^2C slave registers. Note that the hw_i2c.h file in the StellarisWare® Driver Library uses a base address of 0x4002.0800 and 0x4002.1800 for the I^2C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that StellarisWare uses the old slave base address for these offsets. ■ Added GNDPHY and VCCPHY to Connections for Unused Signals tables. ■ Corrected nonlinearity and offset error parameters (E_L, E_D and E_O) in ADC Characteristics table. ■ Added specification for maximum input voltage on a non-power pin when the microcontroller is unpowered (V_{NON} parameter in Maximum Ratings table). ■ Additional minor data sheet clarifications and corrections.
September 2010	7787	■ Reorganized ARM Cortex-M3 Processor Core, Memory Map and Interrupts chapters, creating two new chapters, The Cortex-M3 Processor and Cortex-M3 Peripherals. Much additional content was added, including all the Cortex-M3 registers. ■ Changed register names to be consistent with StellarisWare names: the Cortex-M3 **Interrupt Control and Status (ICSR)** register to the **Interrupt Control and State (INTCTRL)** register, and the Cortex-M3 **Interrupt Set Enable (SETNA)** register to the **Interrupt 0-31 Set Enable (EN0)** register. ■ Added clarification of instruction execution during Flash operations. ■ Modified Figure 8-1 on page 292 to clarify operation of the GPIO inputs when used as an alternate function. ■ Added caution not to apply a Low value to PB7 when debugging; a Low value on the pin causes the JTAG controller to be reset, resulting in a loss of JTAG communication. ■ In General-Purpose Timers chapter, clarified operation of the 32-bit RTC mode. ■ In Electrical Characteristics chapter: – Added I_{LKG} parameter (GPIO input leakage current) to Table 21-4 on page 690. – Corrected values for t_{CLKRF} parameter (SSIClk rise/fall time) in Table 21-21 on page 700. – Added "Ethernet Controller DC Characteristics" table (see Table 21-8 on page 692). ■ Added dimensions for Tray and Tape and Reel shipping mediums.

Table 1. Revision History *(continued)*

Date	Revision	Description
June 2010	7393	■ Corrected base address for SRAM in architectural overview chapter. ■ Clarified system clock operation, adding content to “Clock Control” on page 177. ■ Clarified CAN bit timing examples. ■ In Signal Tables chapter, added table "Connections for Unused Signals." ■ In "Thermal Characteristics" table, corrected thermal resistance value from 34 to 32. ■ In "Reset Characteristics" table, corrected value for supply voltage (VDD) rise time. ■ Additional minor data sheet clarifications and corrections.
April 2010	7007	■ Added caution note to the **I²C Master Timer Period (I2CMTPR)** register description and changed field width to 7 bits. ■ Removed erroneous text about restoring the Flash Protection registers. ■ Added note about $\overline{\text{RST}}$ signal routing. ■ Clarified the function of the TnSTALL bit in the **GPTMCTL** register. ■ Corrected XTALNPHY pin description. ■ Additional minor data sheet clarifications and corrections.
January 2010	6712	■ In "System Control" section, clarified Debug Access Port operation after Sleep modes. ■ Clarified wording on Flash memory access errors. ■ Added section on Flash interrupts. ■ Changed the reset value of the **ADC Sample Sequence Result FIFO n (ADCSSFIFOn)** registers to be indeterminate. ■ Clarified operation of SSI transmit FIFO. ■ Made these changes to the Operating Characteristics chapter: – Added storage temperature ratings to "Temperature Characteristics" table – Added "ESD Absolute Maximum Ratings" table ■ Made these changes to the Electrical Characteristics chapter: – In "Flash Memory Characteristics" table, corrected Mass erase time – Added sleep and deep-sleep wake-up times ("Sleep Modes AC Characteristics" table) – In "Reset Characteristics" table, corrected units for supply voltage (VDD) rise time

Table 1. Revision History *(continued)*

Date	Revision	Description
October 2009	6462	■ Deleted MAXADCSPD bit field from **DCGC0** register as it is not applicable in Deep-Sleep mode. ■ Removed erroneous reference to the WRC bit in the Hibernation chapter. ■ Deleted reset value for 16-bit mode from **GPTMTAILR**, **GPTMTAMATCHR**, and **GPTMTAR** registers because the module resets in 32-bit mode. ■ Clarified PWM source for ADC triggering. ■ Clarified CAN bit timing and corrected examples. ■ Made these changes to the Electrical Characteristics chapter: – Removed V_{SIH} and V_{SIL} parameters from Operating Conditions table. – Added table showing actual PLL frequency depending on input crystal. – Changed the name of the $t_{HIB_REG_WRITE}$ parameter to $t_{HIB_REG_ACCESS}$. – Revised ADC electrical specifications to clarify, including reorganizing and adding new data. – Changed SSI set up and hold times to be expressed in system clocks, not ns.
July 2009	5920	Corrected ordering numbers.
July 2009	5902	■ Clarified Power-on reset and $\overline{\text{RST}}$ pin operation; added new diagrams. ■ Corrected the reset value of the **Hibernation Data (HIBDATA)** and **Hibernation Control (HIBCTL)** registers. ■ Clarified explanation of nonvolatile register programming in Internal Memory chapter. ■ Added explanation of reset value to **FMPRE0/1/2/3**, **FMPPE0/1/2/3**, **USER_DBG**, and **USER_REG0/1** registers. ■ Added description for Ethernet PHY power-saving modes. ■ Corrected the reset values for bits 6 and 7 in the Ethernet **MR24** register. ■ Changed buffer type for $\overline{\text{WAKE}}$ pin to TTL and $\overline{\text{HIB}}$ pin to OD. ■ In ADC characteristics table, changed Max value for GAIN parameter from ±1 to ±3 and added E_{IR} (Internal voltage reference error) parameter. ■ Additional minor data sheet clarifications and corrections.
April 2009	5367	■ Added JTAG/SWD clarification (see “Communication with JTAG/SWD” on page 165). ■ Added clarification that the PLL operates at 400 MHz, but is divided by two prior to the application of the output divisor. ■ Added "GPIO Module DC Characteristics" table (see Table 21-4 on page 690). ■ Additional minor data sheet clarifications and corrections.

Table 1. Revision History *(continued)*

Date	Revision	Description
January 2009	4660	■ Corrected bit type for RELOAD bit field in SysTick Reload Value register; changed to R/W. ■ Clarification added as to what happens when the SSI in slave mode is required to transmit but there is no data in the TX FIFO. ■ Corrected bit timing examples in CAN chapter. ■ Added "Hardware Configuration" section to Ethernet Controller chapter. ■ Additional minor data sheet clarifications and corrections.
November 2008	4283	■ Revised High-Level Block Diagram. ■ Additional minor data sheet clarifications and corrections were made.
October 2008	4149	■ Corrected values for DSOSCSRC bit field in **Deep Sleep Clock Configuration (DSLPCLKCFG)** register. ■ The FMA value for the **FMPRE3** register was incorrect in the Flash Resident Registers table in the Internal Memory chapter. The correct value is 0x0000.0006. ■ In the CAN chapter, major improvements were made including a rewrite of the conceptual information and the addition of new figures to clarify how to use the Controller Area Network (CAN) module. ■ In the Ethernet chapter, major improvements were made including a rewrite of the conceptual information and the addition of new figures to clarify how to use the Ethernet Controller interface. ■ Incorrect Comparator Operating Modes tables were removed from the Analog Comparators chapter.
August 2008	3447	■ Added note on clearing interrupts to Interrupts chapter. ■ Added Power Architecture diagram to System Control chapter. ■ Additional minor data sheet clarifications and corrections.
July 2008	3108	■ Corrected resistor value in ERBIAS signal description. ■ Additional minor data sheet clarifications and corrections.

Table 1. Revision History *(continued)*

Date	Revision	Description
May 2008	2972	■ The 108-Ball BGA pin diagram and pin tables had an error. The following signals were erroneously indicated as available and have now been changed to a No Connect (NC): – Ball C1: Changed PE7 to NC – Ball C2: Changed PE6 to NC – Ball D2: Changed PE5 to NC – Ball D1: Changed PE4 to NC – Ball F1: Changed PD7 to NC – Ball F2: Changed PD6 to NC – Ball E2: Changed PD5 to NC – Ball E1: Changed PD4 to NC ■ As noted in the PCN, three of the nine Ethernet LED configuration options are no longer supported: TX Activity (0x2), RX Activity (0x3), and Collision (0x4). These values for the LED0 and LED1 bit fields in the **MR23** register are now marked as reserved. ■ As noted in the PCN, the option to provide VDD25 power from external sources was removed. Use the LDO output as the source of VDD25 input. ■ As noted in the PCN, pin 41 (ball K3 on the BGA package) was renamed from GNDPHY to ERBIAS. A 12.4-kΩ resistor should be connected between ERBIAS and ground to accommodate future device revisions (see “Functional Description” on page 598). ■ Additional minor data sheet clarifications and corrections.
April 2008	2881	■ The Θ_{JA} value was changed from 55.3 to 34 in the "Thermal Characteristics" table in the Operating Characteristics chapter. ■ Bit 31 of the **DC3** register was incorrectly described in prior versions of the data sheet. A reset of 1 indicates that an even CCP pin is present and can be used as a 32-KHz input clock. ■ Values for $I_{DD_HIBERNATE}$ were added to the "Detailed Power Specifications" table in the "Electrical Characteristics" chapter. ■ The "Hibernation Module DC Electricals" table was added to the "Electrical Characteristics" chapter. ■ The $T_{VDDRISE}$ parameter in the "Reset Characteristics" table in the "Electrical Characteristics" chapter was changed from a max of 100 to 250. ■ The maximum value on Core supply voltage (V_{DD25}) in the "Maximum Ratings" table in the "Electrical Characteristics" chapter was changed from 4 to 3. ■ The operational frequency of the internal 30-kHz oscillator clock source is 30 kHz ± 50% (prior data sheets incorrectly noted it as 30 kHz ± 30%). ■ A value of 0x3 in bits 5:4 of the **MISC** register (OSCSRC) indicates the 30-KHz internal oscillator is the input source for the oscillator. Prior data sheets incorrectly noted 0x3 as a reserved value. ■ The reset for bits 6:4 of the **RCC2** register (OSCSRC2) is 0x1 (IOSC). Prior data sheets incorrectly noted the reset was 0x0 (MOSC). ■ Two figures on clock source were added to the "Hibernation Module": – Clock Source Using Crystal – Clock Source Using Dedicated Oscillator

Table 1. Revision History *(continued)*

Date	Revision	Description
		■ The following notes on battery management were added to the "Hibernation Module" chapter: – Battery voltage is not measured while in Hibernate mode. – 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. ■ A note on high-current applications was added to the GPIO chapter: 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 VOL 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. ■ A note on Schmitt inputs was added to the GPIO chapter: Pins configured as digital inputs are Schmitt-triggered. ■ The Buffer type on the WAKE pin changed from OD to - in the Signal Tables. ■ The "Differential Sampling Range" figures in the ADC chapter were clarified. ■ The last revision of the data sheet (revision 2550) introduced two errors that have now been corrected: – The LQFP pin diagrams and pin tables were missing the comparator positive and negative input pins. – The base address was listed incorrectly in the **FMPRE0** and **FMPPE0** register bit diagrams. ■ Additional minor data sheet clarifications and corrections.
March 2008	2550	Started tracking revision history.

About This Document

This data sheet provides reference information for the LM3S8938 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/M4 Instruction Set Technical User's Manual*
- *Stellaris® Graphics Library User's Guide*
- *Stellaris® Peripheral Driver Library User's Guide*

The following related documents are also referenced:

- *ARM® Debug Interface V5 Architecture Specification*
- *ARM® Embedded Trace Macrocell 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 33.

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 71.
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. 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. 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. 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

The 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 Stellaris family offers efficient performance and extensive integration, favorably positioning the device into cost-conscious applications requiring significant control-processing and connectivity capabilities. The Stellaris LM3S8000 series combines Bosch Controller Area Network technology with both a 10/100 Ethernet Media Access Control (MAC) and Physical (PHY) layer.

The LM3S8938 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, motion control, medical instrumentation, and fire and security.

For applications requiring extreme conservation of power, the LM3S8938 microcontroller features a battery-backed Hibernation module to efficiently power down the LM3S8938 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 LM3S8938 microcontroller perfectly for battery applications.

In addition, the LM3S8938 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 LM3S8938 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 735 for ordering information for Stellaris family devices.

1.1 Product Features

The LM3S8938 microcontroller includes the following product features:

- 32-Bit RISC Performance
  - 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
  - System timer (SysTick), providing a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism
  - Thumb®-compatible Thumb-2-only instruction set processor core for high code density
  - 50-MHz operation
  - Hardware-division and single-cycle-multiplication

- Integrated Nested Vectored Interrupt Controller (NVIC) providing deterministic interrupt handling
- 34 interrupts with eight priority levels
- Memory protection unit (MPU), providing a privileged mode for protected operating system functionality
- Unaligned data access, enabling data to be efficiently packed into memory
- Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control

■ ARM® Cortex™-M3 Processor Core

- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Rapid application execution through Harvard architecture characterized by separate buses for instruction and data.
- Exceptional interrupt handling, by implementing the register manipulations required for handling an interrupt in hardware.
- Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining
- Memory protection unit (MPU) to provide a privileged mode of operation for complex applications.
- Migration from the ARM7™ processor family for better performance and power efficiency.
- Full-featured debug solution
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trigger (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
- Optimized for single-cycle flash usage
- Three sleep modes with clock gating for low power
- Single-cycle multiply instruction and hardware divide
- Atomic operations
- ARM Thumb2 mixed 16-/32-bit instruction set

  - 1.25 DMIPS/MHz
- JTAG
  - 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)
- Hibernation
  - System power control using discrete external regulator
  - Dedicated pin for waking from an external signal
  - Low-battery detection, signaling, and interrupt generation
  - 32-bit real-time clock (RTC)
  - Two 32-bit RTC match registers for timed wake-up and interrupt generation
  - Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal
  - RTC predivider trim for making fine adjustments to the clock rate
  - 64 32-bit words of non-volatile memory
  - Programmable interrupts for RTC match, external wake, and low battery events
- Internal Memory
  - 256 KB single-cycle flash
    - User-managed flash block protection on a 2-KB block basis
    - User-managed flash data programming
    - User-defined and managed flash-protection block
  - 64 KB single-cycle SRAM
- GPIOs
  - 3-38 GPIOs, depending on configuration
  - 5-V-tolerant in input configuration
  - Fast toggle capable of a change every two clock cycles
  - 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 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 be configured with an 18-mA pad drive for high-current applications
        - Slew rate control for the 8-mA drive
        - Open drain enables
        - Digital input enables
- General-Purpose Timers
    - Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers/counters. Each GPTM can be configured to operate independently:
        - As a single 32-bit timer
        - As one 32-bit Real-Time Clock (RTC) to event capture
        - For Pulse Width Modulation (PWM)
        - To trigger analog-to-digital conversions
    - 32-bit Timer modes
        - Programmable one-shot timer
        - Programmable periodic timer
        - Real-Time Clock when using an external 32.768-KHz clock as the input
        - User-enabled stalling when the controller asserts CPU Halt flag during debug
        - ADC event trigger
    - 16-bit Timer modes
        - General-purpose timer function with an 8-bit prescaler (for one-shot and periodic modes only)
        - Programmable one-shot timer
        - Programmable periodic timer

        - User-enabled stalling when the controller asserts CPU Halt flag during debug
        - ADC event trigger
    - 16-bit Input Capture modes
        - Input edge count capture
        - Input edge time capture
    - 16-bit PWM mode
        - Simple PWM mode with software-programmable output inversion of the PWM signal
- ARM FiRM-compliant Watchdog Timer
    - 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 controller asserts the CPU Halt flag during debug
- ADC
    - Eight analog input channels
    - Single-ended and differential-input configurations
    - On-chip internal temperature sensor
    - Sample rate of one million samples/second
    - Flexible, configurable analog-to-digital conversion
    - Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
    - Flexible trigger control
        - Controller (software)
        - Timers
        - Analog Comparators
        - GPIO
    - Hardware averaging of up to 64 samples for improved accuracy
    - Converter uses an internal 3-V reference

  - Power and ground for the analog circuitry is separate from the digital power and ground

- UART
  - Three fully programmable 16C550-type UARTs with IrDA support
  - Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
  - Programmable baud-rate generator allowing speeds up to 3.125 Mbps
  - 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
- Synchronous Serial Interface (SSI)
  - Master or slave operation
  - Programmable clock bit rate and prescale
  - Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
  - Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
  - Programmable data frame size from 4 to 16 bits
  - Internal loopback test mode for diagnostic/debug testing
- I^2C
  - Two I^2C modules, each with the following features:

  - Devices on the I²C bus can be designated as either a master or a slave
    - Supports both sending and receiving data as either a master or a slave
    - Supports simultaneous master and slave operation
  - Four I²C modes
    - Master transmit
    - Master receive
    - Slave transmit
    - Slave receive
  - Two transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
  - Master and slave interrupt generation
    - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
    - Slave generates interrupts when data has been sent or requested by a master
  - Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

- Controller Area Network (CAN)
  - CAN protocol version 2.0 part A/B
  - Bit rates up to 1 Mbps
  - 32 message objects with individual identifier masks
  - Maskable interrupt
  - Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
  - Programmable Loopback mode for self-test operation
  - Programmable FIFO mode enables storage of multiple message objects
  - Gluelessly attaches to an external CAN interface through the CANnTX and CANnRX signals
- 10/100 Ethernet Controller
  - Conforms to the *IEEE 802.3-2002 specification*
    - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line
    - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
    - Full-featured auto-negotiation

  - Multiple operational modes
    - Full- and half-duplex 100 Mbps
    - Full- and half-duplex 10 Mbps
    - Power-saving and power-down modes
  - Highly configurable
    - Programmable MAC address
    - LED activity selection
    - Promiscuous mode support
    - CRC error-rejection control
    - User-configurable interrupts
  - Physical media manipulation
    - Automatic MDI/MDI-X cross-over correction
    - Register-programmable transmit amplitude
    - Automatic polarity correction and 10BASE-T signal reception
  - IEEE 1588 Precision Time Protocol - provides highly accurate time stamps for individual packets

- Analog Comparators
  - Three independent integrated analog comparators
  - Configurable for output to drive an output pin, generate an interrupt, or initiate an ADC sample sequence
  - Compare external pin input to external pin input or to internal programmable voltage reference
  - Compare a test voltage against any one of these voltages
    - An individual external reference voltage
    - A shared single external reference voltage
    - A shared internal reference voltage

- Power
  - On-chip Low Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V
  - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
  - Low-power options on controller: Sleep and Deep-sleep modes

  - Low-power options for peripherals: software controls shutdown of individual peripherals
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Flexible Reset Sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Internal low drop-out (LDO) regulator output goes unregulated
- Industrial and extended temperature 100-pin RoHS-compliant LQFP package
- Industrial-range 108-ball RoHS-compliant BGA package

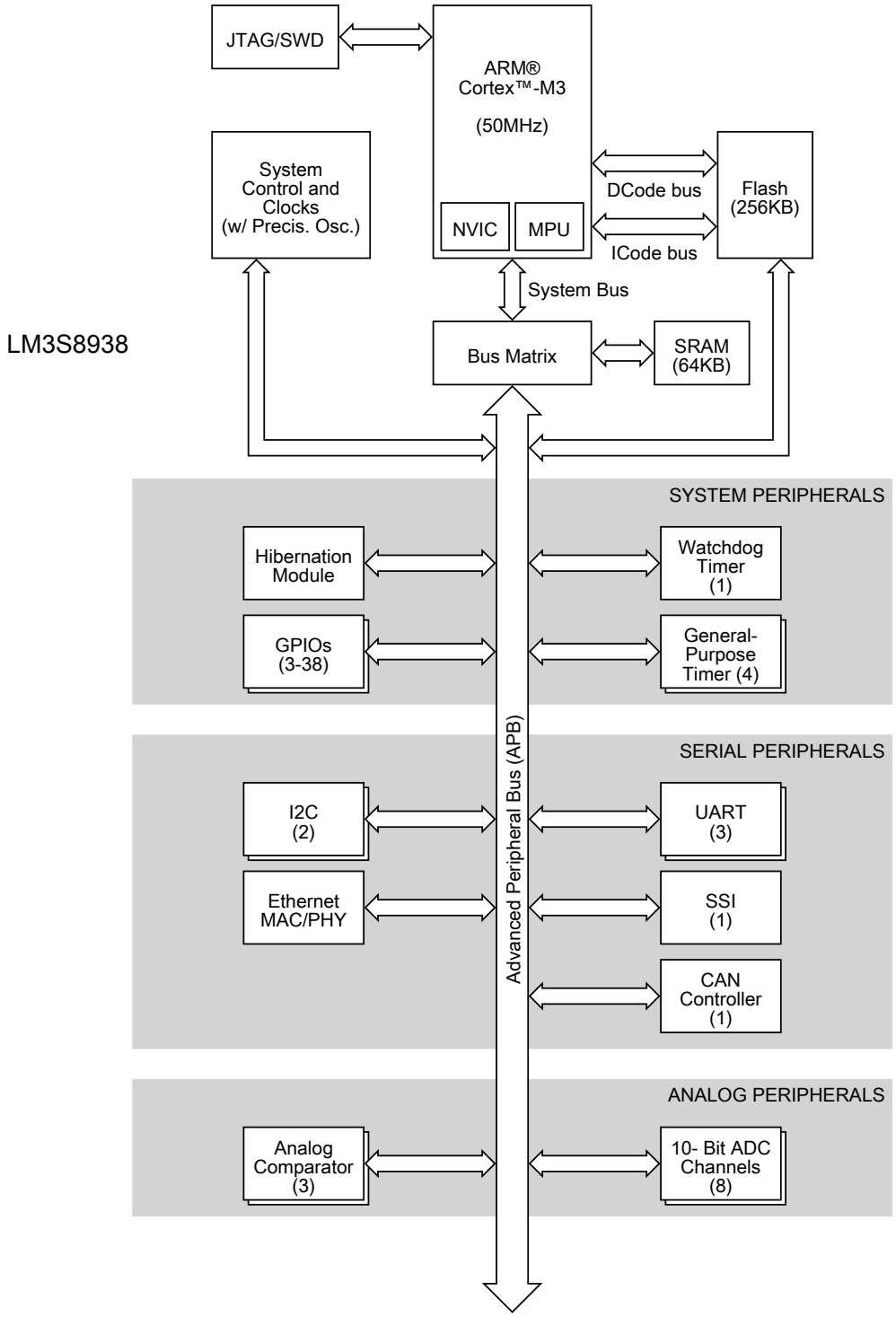
1.2 Target Applications

- Remote monitoring
- Electronic point-of-sale (POS) machines
- Test and measurement equipment
- Network appliances and switches
- 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 44 depicts the features on the Stellaris LM3S8938 microcontroller.

Figure 1-1. Stellaris LM3S8938 Microcontroller High-Level Block Diagram



1.4 Functional Overview

The following sections provide an overview of the features of the LM3S8938 microcontroller. The page number in parenthesis indicates where that feature is discussed in detail. Ordering and support information can be found in “Ordering and Contact Information” on page 735.

1.4.1 ARM Cortex™-M3

1.4.1.1 Processor Core (see page 52)

All members of the Stellaris product family, including the LM3S8938 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.

1.4.1.2 Memory Map (see page 71)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S8938 controller can be found in Table 2-4 on page 71. Register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map.

1.4.1.3 System Timer (SysTick) (see page 94)

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 which 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. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

1.4.1.4 Nested Vectored Interrupt Controller (NVIC) (see page 95)

The LM3S8938 controller includes the ARM Nested Vectored Interrupt Controller (NVIC) on the ARM® Cortex™-M3 core. The NVIC and Cortex-M3 prioritize and handle all exceptions. All exceptions are handled 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, which enables 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. Software can set eight priority levels on 7 exceptions (system handlers) and 34 interrupts.

1.4.1.5 System Control Block (SCB) (see page 97)

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

1.4.1.6 Memory Protection Unit (MPU) (see page 97)

The MPU 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.

1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S8938 controller features Pulse Width Modulation (PWM) outputs.

1.4.2.1 PWM

Pulse width modulation (PWM) is a powerful technique for digitally encoding analog signal levels. High-resolution counters are used to generate a square wave, and the duty cycle of the square wave is modulated to encode an analog signal. Typical applications include switching power supplies and motor control.

On the LM3S8938, PWM motion control functionality can be achieved through:

- The motion control features of the general-purpose timers using the CCP pins

CCP Pins (see page 340)

The General-Purpose Timer Module's CCP (Capture Compare PWM) pins are software programmable to support a simple PWM mode with a software-programmable output inversion of the PWM signal.

1.4.3 Analog Peripherals

To handle analog signals, the LM3S8938 microcontroller offers an Analog-to-Digital Converter (ADC).

For support of analog signals, the LM3S8938 microcontroller offers three analog comparators.

1.4.3.1 ADC (see page 394)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number.

The LM3S8938 ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. Four buffered sample sequences allow rapid sampling of up to eight analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

1.4.3.2 Analog Comparators (see page 644)

An analog comparator is a peripheral that compares two analog voltages, and provides a logical output that signals the comparison result.

The LM3S8938 microcontroller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

A comparator can compare a test voltage against any one of these voltages:

- An individual external reference voltage

- A shared single external reference voltage
- A shared internal reference voltage

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

1.4.4 Serial Communications Peripherals

The LM3S8938 controller supports both asynchronous and synchronous serial communications with:

- Three fully programmable 16C550-type UARTs
- One SSI module
- Two I^2C modules
- One CAN unit
- Ethernet controller

1.4.4.1 UART (see page 430)

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 LM3S8938 controller includes three fully programmable 16C550-type UARTs that support data transfer speeds up to 3.125 Mbps. (Although similar in functionality to a 16C550 UART, it is not register-compatible.) In addition, each UART is capable of supporting IrDA.

Separate 16x8 transmit (TX) and receive (RX) FIFOs reduce CPU interrupt service loading. The UART can generate individually masked interrupts from the RX, TX, modem status, and error conditions. The module provides a single combined interrupt when any of the interrupts are asserted and are unmasked.

1.4.4.2 SSI (see page 473)

Synchronous Serial Interface (SSI) is a four-wire bi-directional full and low-speed communications interface.

The LM3S8938 controller includes one SSI module that provides the functionality for synchronous serial communications with peripheral devices, and can be configured to use the Freescale SPI, MICROWIRE, or TI synchronous serial interface frame formats. The size of the data frame is also configurable, and can be set between 4 and 16 bits, inclusive.

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 TX and RX paths are buffered with internal FIFOs, allowing up to eight 16-bit values to be stored independently.

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 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.

1.4.4.3 I²C (see page 511)

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.

The LM3S8938 controller includes two I^2C modules that provide the ability to communicate to other IC devices over an I^2C bus. The I^2C bus supports devices that can both transmit and receive (write and read) data.

Devices 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 also supports the simultaneous operation as both a master and a slave. The four I^2C modes are: Master Transmit, Master Receive, Slave Transmit, and Slave Receive.

A Stellaris I^2C module can operate at two speeds: Standard (100 Kbps) and Fast (400 Kbps).

Both the I^2C master and slave can generate interrupts. The I^2C master generates interrupts when a transmit or receive operation completes (or aborts due to an error). The I^2C slave generates interrupts when data has been sent or requested by a master.

1.4.4.4 Controller Area Network (see page 548)

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

A transmitter sends a message to all CAN nodes (broadcasting). Each node decides on the basis of the identifier received whether it should process the message. The identifier also determines the priority that the message enjoys in competition for bus access. Each CAN message can transmit from 0 to 8 bytes of user information. The LM3S8938 includes one CAN unit.

1.4.4.5 Ethernet Controller (see page 595)

Ethernet is a frame-based computer networking technology for local area networks (LANs). Ethernet has been standardized as IEEE 802.3. It defines a number of wiring and signaling standards for the physical layer, two means of network access at the Media Access Control (MAC)/Data Link Layer, and a common addressing format.

The Stellaris® Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface device. The Ethernet Controller conforms to IEEE 802.3 specifications and fully supports 10BASE-T and 100BASE-TX standards. In addition, the Ethernet Controller supports automatic MDI/MDI-X cross-over correction.

1.4.5 System Peripherals

1.4.5.1 Programmable GPIOs (see page 286)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections.

The Stellaris GPIO module is comprised of seven 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 3-38 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see "Signal Tables" on page 659 for the signals available to each GPIO pin).

The GPIO module features programmable interrupt generation as either edge-triggered or level-sensitive on all pins, programmable control for GPIO pad configuration, and bit masking in both read and write operations through address lines. Pins configured as digital inputs are Schmitt-triggered.

1.4.5.2 Four Programmable Timers (see page 333)

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 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.

When configured in 32-bit mode, a timer can run as a Real-Time Clock (RTC), one-shot timer or periodic timer. When in 16-bit mode, a timer can run as a one-shot timer or periodic timer, and can extend its precision by using an 8-bit prescaler. A 16-bit timer can also be configured for event capture or Pulse Width Modulation (PWM) generation.

1.4.5.3 Watchdog Timer (see page 370)

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 to the failure of an external device to respond in the expected way.

The Stellaris Watchdog Timer module consists of a 32-bit down counter, a programmable load register, interrupt generation logic, and a locking register.

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.

1.4.6 Memory Peripherals

The LM3S8938 controller offers both single-cycle SRAM and single-cycle Flash memory.

1.4.6.1 SRAM (see page 259)

The LM3S8938 static random access memory (SRAM) controller supports 64 KB SRAM. The internal SRAM of the Stellaris devices starts at base address 0x2000.0000 of the device memory map. To reduce the number of time-consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the new 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.

1.4.6.2 Flash (see page 260)

The LM3S8938 Flash controller supports 256 KB of flash memory. The flash 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.

1.4.7 Additional Features

1.4.7.1 JTAG TAP Controller (see page 158)

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 composed of the standard five pins: TRST, 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. This is implemented 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 outputs. The multiplexer is controlled by the Stellaris JTAG controller, which has comprehensive programming for the ARM, Stellaris, and unimplemented JTAG instructions.

1.4.7.2 System Control and Clocks (see page 171)

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

1.4.7.3 Hibernation Module (see page 238)

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, a real-time clock with a pair of match registers, low-battery detection circuitry, and interrupt signalling to the processor. It also includes 64 32-bit words of non-volatile memory that can be used for saving state during hibernation.

1.4.8 Hardware Details

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

- “Pin Diagram” on page 657
- “Signal Tables” on page 659

- “Operating Characteristics” on page 688
- “Electrical Characteristics” on page 689
- “Package Information” on page 737

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:

- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Rapid application execution through Harvard architecture characterized by separate buses for instruction and data.
- Exceptional interrupt handling, by implementing the register manipulations required for handling an interrupt in hardware.
- Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining
- Memory protection unit (MPU) to provide a privileged mode of operation for complex applications.
- Migration from the ARM7™ processor family for better performance and power efficiency.
- Full-featured debug solution
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trigger (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
- Optimized for single-cycle flash usage
- Three sleep modes with clock gating for low power
- Single-cycle multiply instruction and hardware divide
- Atomic operations
- ARM Thumb2 mixed 16-/32-bit instruction set
- 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/M4 Instruction Set Technical User's Manual*.

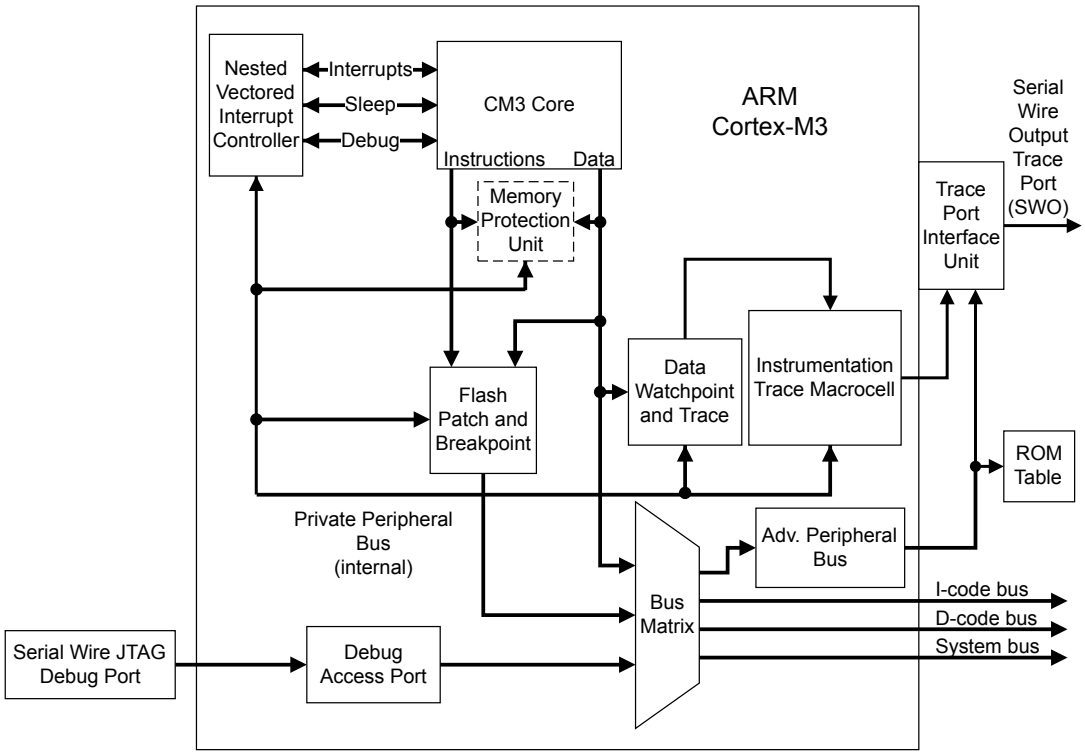
2.1 Block Diagram

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 a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-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 based on Thumb-2 technology, 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.

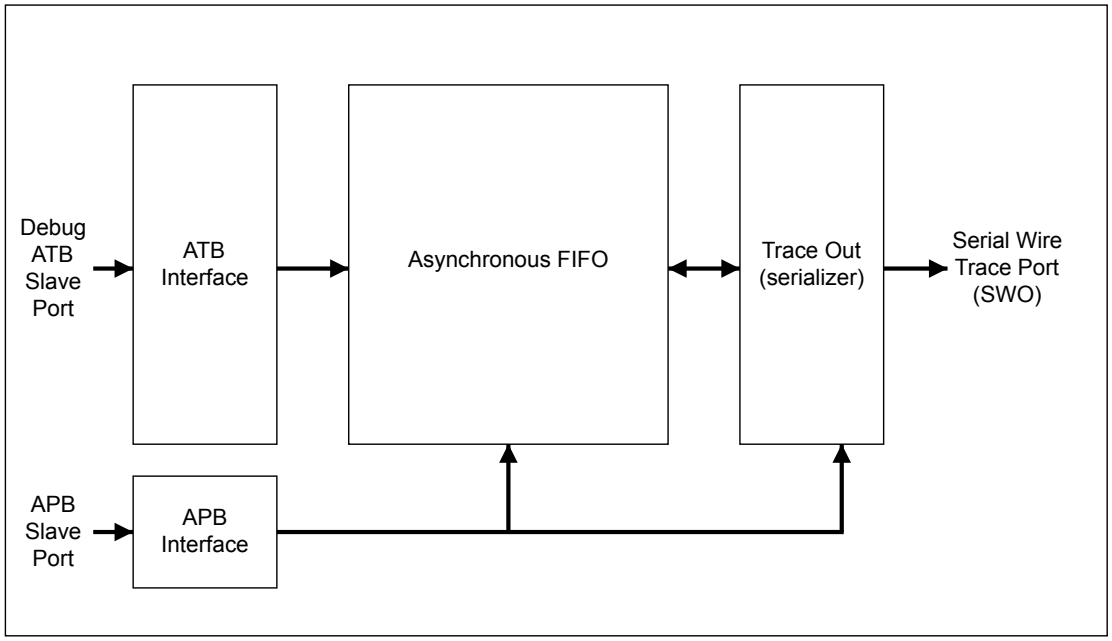
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words in the program code in the CODE memory region. This enables applications stored in a read-only area of Flash 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 55.

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 94).

- Nested Vectored Interrupt Controller (NVIC)

  An embedded interrupt controller that supports low latency interrupt processing (see “Nested Vectored Interrupt Controller (NVIC)” on page 95).

- 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 97).

- 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 97).

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 70) controls whether software execution is privileged or unprivileged. In Handler mode, software execution is always privileged.

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

2.3.2 Stacks

The processor uses a full descending stack, meaning that the stack pointer indicates the last stacked item on the memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks:

the main stack and the process stack, with a pointer for each held in independent registers (see the **SP** register on page 60).

In Thread mode, the **CONTROL** register (see page 70) 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 57.

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 70).

2.3.3 Register Map

Figure 2-3 on page 57 shows the Cortex-M3 register set. Table 2-2 on page 58 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

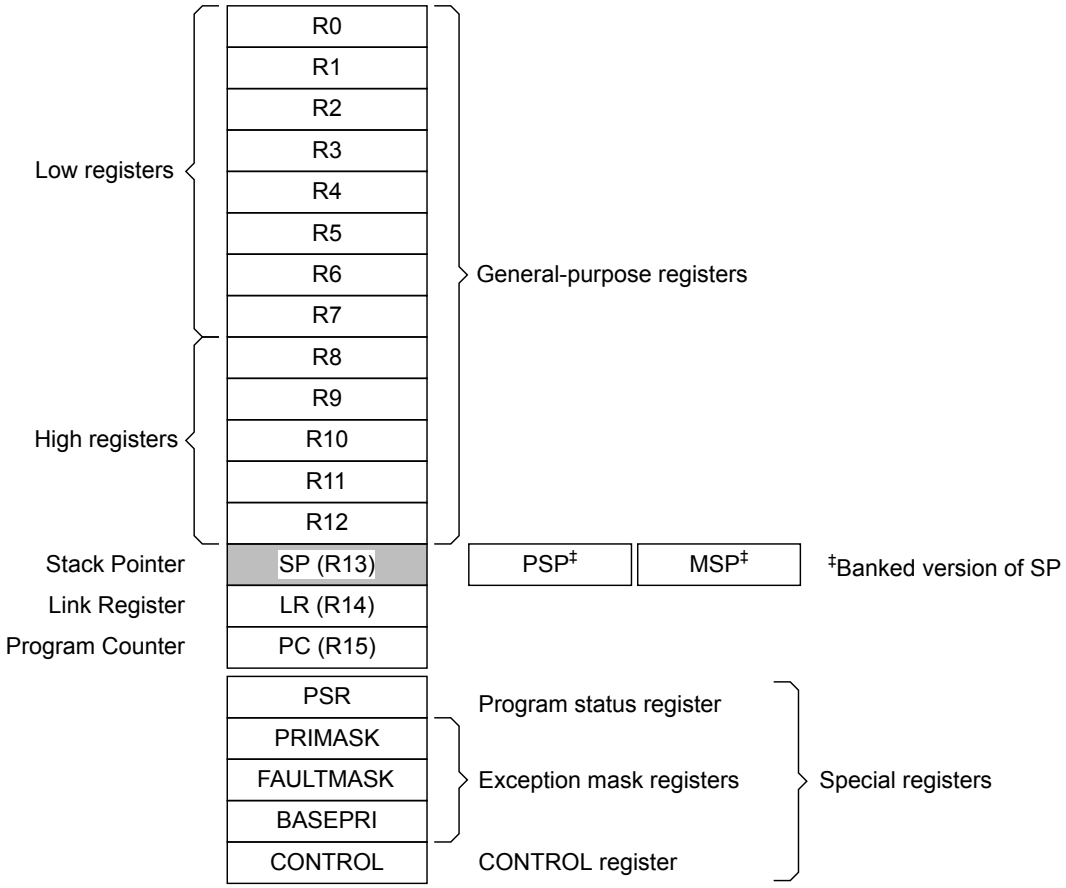


Table 2-2. Processor Register Map

Offset	Name	Type	Reset	Description	See page
-	R0	R/W	-	Cortex General-Purpose Register 0	59
-	R1	R/W	-	Cortex General-Purpose Register 1	59
-	R2	R/W	-	Cortex General-Purpose Register 2	59
-	R3	R/W	-	Cortex General-Purpose Register 3	59
-	R4	R/W	-	Cortex General-Purpose Register 4	59
-	R5	R/W	-	Cortex General-Purpose Register 5	59
-	R6	R/W	-	Cortex General-Purpose Register 6	59
-	R7	R/W	-	Cortex General-Purpose Register 7	59
-	R8	R/W	-	Cortex General-Purpose Register 8	59
-	R9	R/W	-	Cortex General-Purpose Register 9	59
-	R10	R/W	-	Cortex General-Purpose Register 10	59
-	R11	R/W	-	Cortex General-Purpose Register 11	59
-	R12	R/W	-	Cortex General-Purpose Register 12	59
-	SP	R/W	-	Stack Pointer	60
-	LR	R/W	0xFFFF.FFFF	Link Register	61
-	PC	R/W	-	Program Counter	62
-	PSR	R/W	0x0100.0000	Program Status Register	63
-	PRIMASK	R/W	0x0000.0000	Priority Mask Register	67
-	FAULTMASK	R/W	0x0000.0000	Fault Mask Register	68
-	BASEPRI	R/W	0x0000.0000	Base Priority Mask Register	69
-	CONTROL	R/W	0x0000.0000	Control Register	70

2.3.4 Register Descriptions

This section lists and describes the Cortex-M3 registers, in the order shown in Figure 2-3 on page 57. 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)

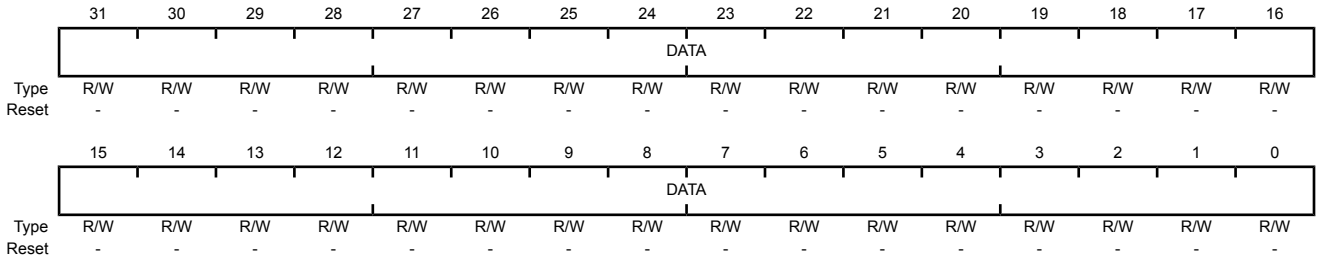
Register 12: Cortex General-Purpose Register 11 (R11)

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 -



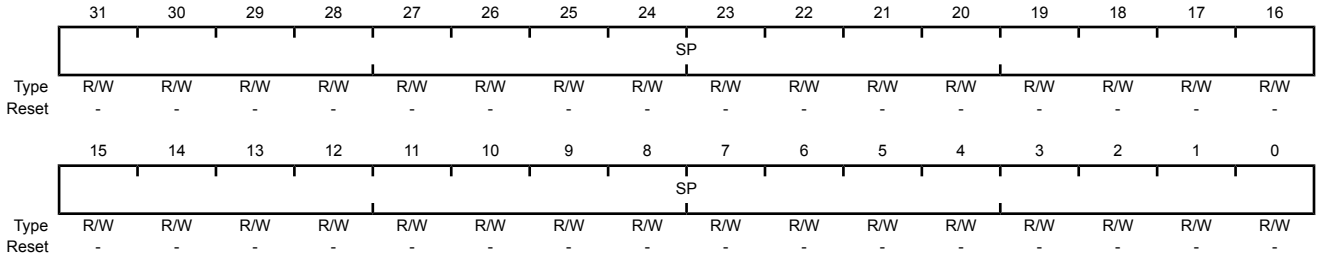
Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	-	Register data.

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.

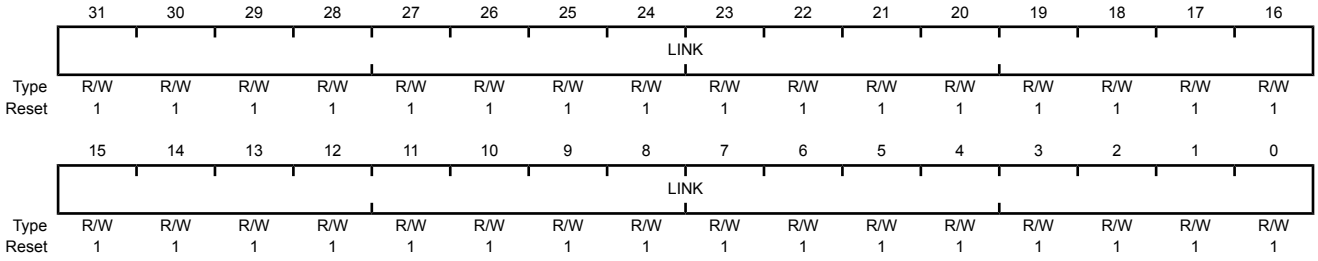
Register 15: Link Register (LR)

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 87 for the values and description.

Link Register (LR)

Type R/W, reset 0xFFFF.FFFF



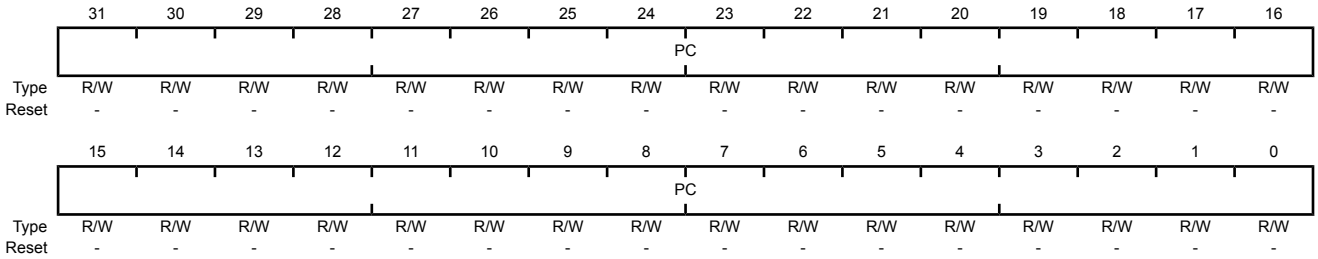
Bit/Field	Name	Type	Reset	Description
31:0	LINK	R/W	0xFFFF.FFFF	This field is the return address.

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 5:0

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 85).

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 63 shows the possible register combinations for the **PSR**. See the `MRS` and `MSR` instruction descriptions in the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information about how to access the program status registers.

Table 2-3. PSR Register Combinations

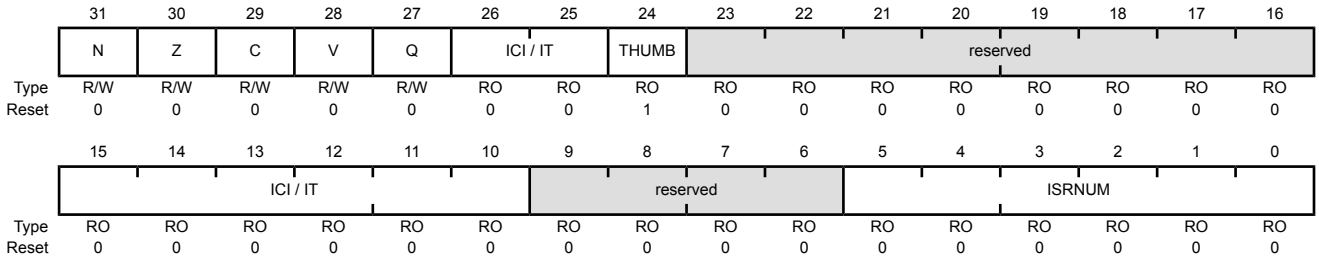
Register	Type	Combination
PSR	R/W[a, b]	**APSR**, **EPSR**, and **IPSR**
IEPSR	RO	**EPSR** and **IPSR**
IAPSR	R/W[a]	**APSR** and **IPSR**
EAPSR	R/W[b]	**APSR** and **EPSR**

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 Value Description 1 The previous operation result was negative or less than. 0 The previous operation result was positive, zero, greater than, or equal. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
30	Z	R/W	0	**APSR** Zero Flag Value Description 1 The previous operation result was zero. 0 The previous operation result was non-zero. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
29	C	R/W	0	**APSR** Carry or Borrow Flag Value Description 1 The previous add operation resulted in a carry bit or the previous subtract operation did not result in a borrow bit. 0 The previous add operation did not result in a carry bit or the previous subtract operation resulted in a borrow bit. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
28	V	R/W	0	**APSR** Overflow Flag Value Description 1 The previous operation resulted in an overflow. 0 The previous operation did not result in an overflow. The value of this bit is only meaningful when accessing **PSR** or **APSR**.
27	Q	R/W	0	**APSR** DSP Overflow and Saturation Flag Value Description 1 DSP Overflow or saturation has occurred. 0 DSP overflow or saturation has not occurred since reset or since the bit was last cleared. The value of this bit is only meaningful when accessing **PSR** or **APSR**. 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 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. When **EPSR** holds the ICI execution state, bits 26:25 are zero. The If-Then block contains up to four instructions following an IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information. The value of this field is only meaningful when accessing **PSR** or **EPSR**.
24	THUMB	RO	1	**EPSR** Thumb State This bit indicates the Thumb state and should always be set. The following can clear the THUMB bit: ■ The BLX, BX and POP{PC} instructions ■ Restoration from the stacked **xPSR** value on an exception return ■ Bit 0 of the vector value on an exception entry or reset Attempting to execute instructions when this bit is clear results in a fault or lockup. See "Lockup" on page 89 for more information. 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 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. 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. 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/M4 Instruction Set Technical User's Manual* for more information. The value of this field is only meaningful when accessing **PSR** or **EPSR**.
9:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

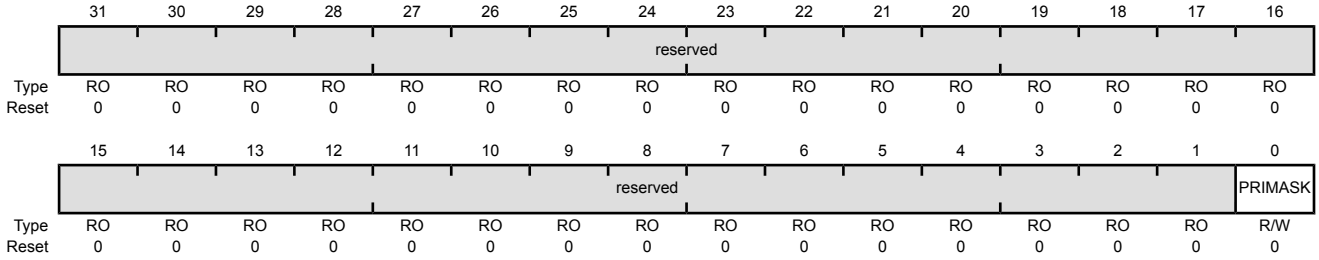
Bit/Field	Name	Type	Reset	Description
5:0	ISRNUM	RO	0x00	**IPSR** ISR Number This field contains the exception type number of the current Interrupt Service Routine (ISR). Value — Description 0x00 — Thread mode 0x01 — Reserved 0x02 — NMI 0x03 — Hard fault 0x04 — Memory management fault 0x05 — Bus fault 0x06 — Usage fault 0x07-0x0A — Reserved 0x0B — SVCall 0x0C — Reserved for Debug 0x0D — Reserved 0x0E — PendSV 0x0F — SysTick 0x10 — Interrupt Vector 0 0x11 — Interrupt Vector 1 ... — ... 0x3B — Interrupt Vector 43 0x3C-0x3F — Reserved See “Exception Types” on page 80 for more information. 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/M4 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 80.

Priority Mask Register (PRIMASK)

Type R/W, reset 0x0000.0000



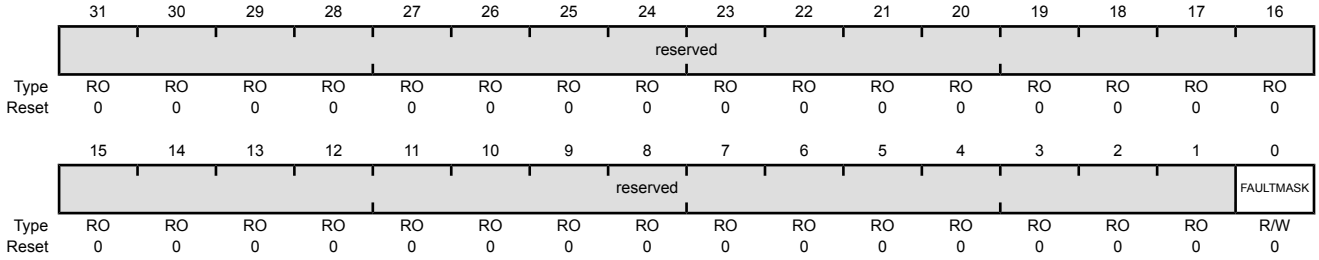
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PRIMASK	R/W	0	Priority Mask Value Description 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/M4 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 80.

Fault Mask Register (FAULTMASK)

Type R/W, reset 0x0000.0000



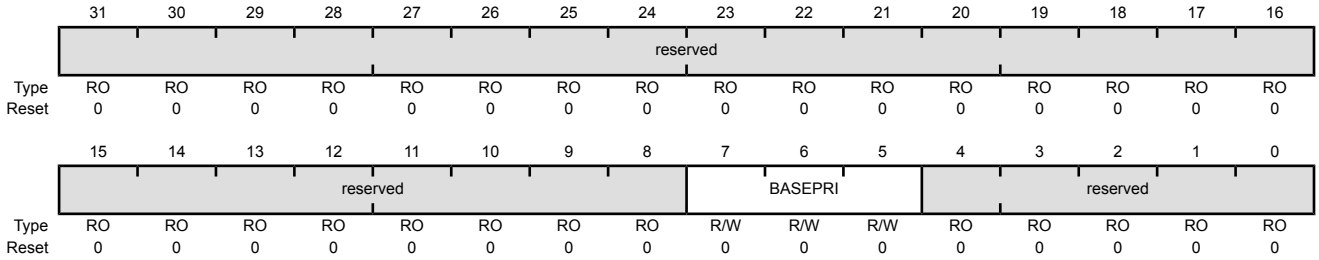
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FAULTMASK	R/W	0	Fault Mask Value Description 1 Prevents the activation of all exceptions except for NMI. 0 No effect. 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 80.

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 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. Value Description 0x0 All exceptions are unmasked. 0x1 All exceptions with priority level 1-7 are masked. 0x2 All exceptions with priority level 2-7 are masked. 0x3 All exceptions with priority level 3-7 are masked. 0x4 All exceptions with priority level 4-7 are masked. 0x5 All exceptions with priority level 5-7 are masked. 0x6 All exceptions with priority level 6-7 are masked. 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)

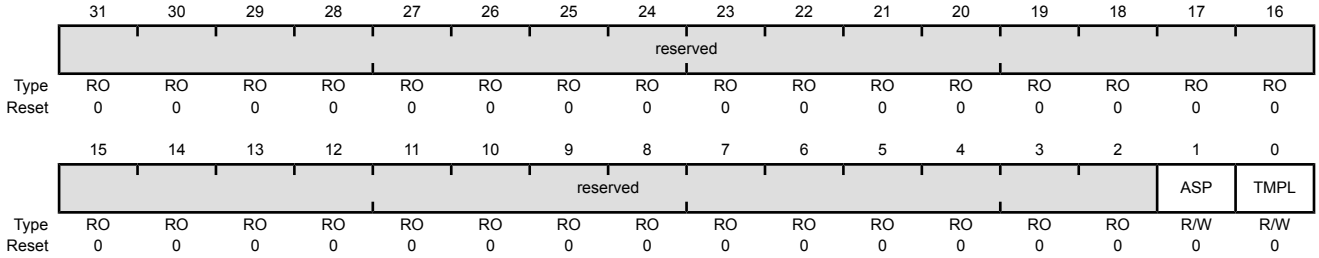
The **CONTROL** register controls the stack used and the privilege level for software execution when the processor is in Thread mode. This register is only accessible in privileged mode.

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 87). 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/M4 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 87.

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/M4 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 Value Description 1 **PSP** is the current stack pointer. 0 **MSP** is the current stack pointer 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 Value Description 1 Unprivileged software can be executed in Thread mode. 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 85 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” on page 95 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 73 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 LM3S8938 controller is provided in Table 2-4 on page 71. 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 75).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see “Cortex-M3 Peripherals” on page 94).

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	0x0003.FFFF	On-chip Flash	265
0x0004.0000	0x1FFF.FFFF	Reserved	-
0x2000.0000	0x2000.FFFF	Bit-banded on-chip SRAM	259
0x2001.0000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x221F.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000	259
0x2220.0000	0x3FFF.FFFF	Reserved	-
FiRM Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	373
0x4000.1000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	298
0x4000.5000	0x4000.5FFF	GPIO Port B	298
0x4000.6000	0x4000.6FFF	GPIO Port C	298
0x4000.7000	0x4000.7FFF	GPIO Port D	298
0x4000.8000	0x4000.8FFF	SSI0	485

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

Start	End	Description	For details, see page ...
0x4000.9000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	439
0x4000.D000	0x4000.DFFF	UART1	439
0x4000.E000	0x4000.EFFF	UART2	439
0x4000.F000	0x4001.FFFF	Reserved	-
Peripherals			
0x4002.0000	0x4002.0FFF	I^2C 0	526
0x4002.1000	0x4002.1FFF	I^2C 1	526
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	298
0x4002.5000	0x4002.5FFF	GPIO Port F	298
0x4002.6000	0x4002.6FFF	GPIO Port G	298
0x4002.7000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer 0	345
0x4003.1000	0x4003.1FFF	Timer 1	345
0x4003.2000	0x4003.2FFF	Timer 2	345
0x4003.3000	0x4003.3FFF	Timer 3	345
0x4003.4000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC0	403
0x4003.9000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	644
0x4003.D000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	569
0x4004.1000	0x400F.BFFF	Reserved	-
0x400F.C000	0x400F.CFFF	Hibernation Module	246
0x400F.D000	0x400F.DFFF	Flash memory control	265
0x400F.E000	0x400F.EFFF	System control	185
0x400F.F000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0xDFFF.FFFF	Reserved	-
Private Peripheral Bus			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	54
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	54
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	54
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Cortex-M3 Peripherals (SysTick, NVIC, MPU and SCB)	102
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	55
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 74).

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 73 shows the behavior of accesses to each region in the memory map. See “Memory Regions, Types and Attributes” on page 73 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 71 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 75).
0x4000.0000 - 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see Table 2-7 on page 76).
0x6000.0000 - 0x9FFF.FFFF	External RAM	Normal	-	This executable region is for data.

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

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

The Code, SRAM, and external RAM regions can hold programs. However, it is recommended that programs always use the Code region because the Cortex-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 97.

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 73 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/M4 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 75. 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 76. For the specific address range of the bit-band regions, see Table 2-4 on page 71.

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

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

Table 2-6. SRAM Memory Bit-Banding Regions

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

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

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x2200.0000	0x221F.FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 2-7. Peripheral Memory Bit-Banding Regions

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

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

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

```
bit_word_addr = bit_band_base + bit_word_offset
```

where:

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

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

bit_band_base
The starting address of the alias region.

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

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

Figure 2-4 on page 77 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)
  ```

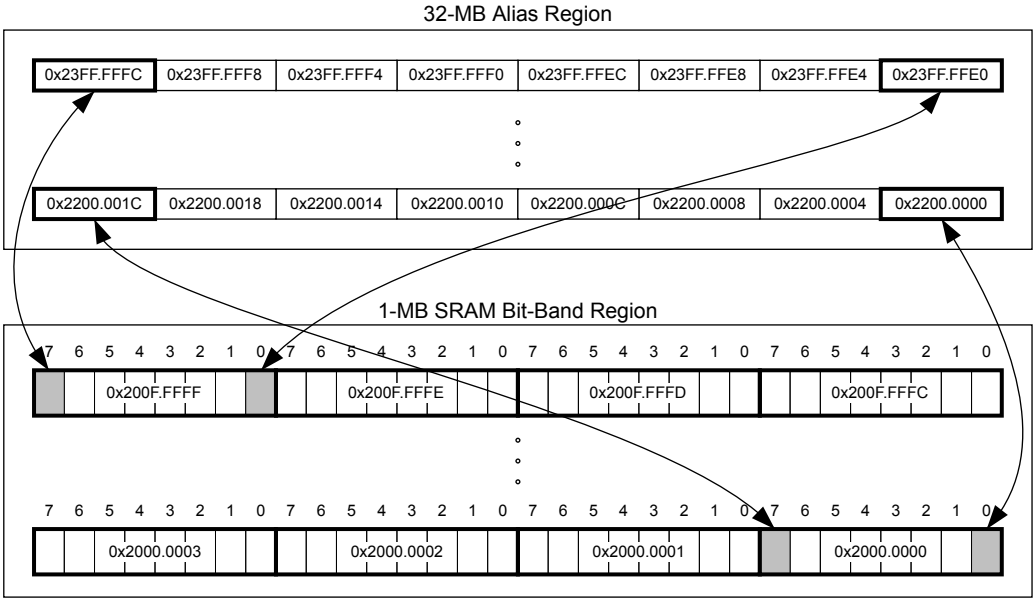
- The alias word at 0x2200.0000 maps to bit 0 of the bit-band byte at 0x2000.0000:

```
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.

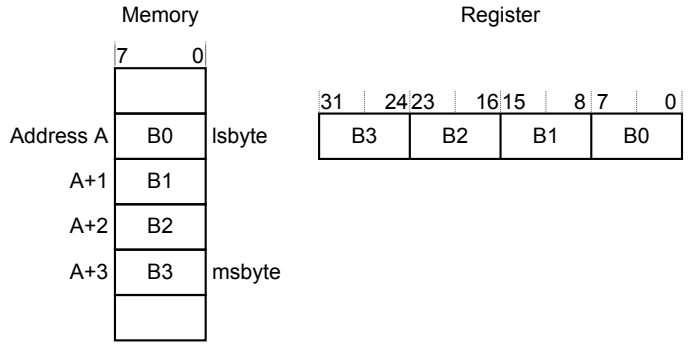
2.4.5.2 Directly Accessing a Bit-Band Region

"Behavior of Memory Accesses" on page 73 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 78 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 was performed.

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

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

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

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

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

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

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

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

The Cortex-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/M4 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 81 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 34 interrupts (listed in Table 2-9 on page 82).

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 95.

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 95 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 82 lists the interrupts on the LM3S8938 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 81 shows as having configurable priority (see the **SYSHNDCTRL** register on page 136 and the **DIS0** register on page 111).

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” on page 87.

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
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

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

Exception Type	Vector Number	Priority[a]	Vector Address or Offset[b]	Activation
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 83.

c. See **SYSPRI1** on page 133.

d. See **PRIn** registers on page 119.

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 Timer 0
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	25	0x0000.00A4	Analog Comparator 0
42	26	0x0000.00A8	Analog Comparator 1
43	27	0x0000.00AC	Analog Comparator 2
44	28	0x0000.00B0	System Control
45	29	0x0000.00B4	Flash Memory Control
46	30	0x0000.00B8	GPIO Port F
47	31	0x0000.00BC	GPIO Port G
48	32	-	Reserved

Table 2-9. Interrupts *(continued)*

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
49	33	0x0000.00C4	UART2
50	34	-	Reserved
51	35	0x0000.00CC	Timer 3A
52	36	0x0000.00D0	Timer 3B
53	37	0x0000.00D4	I^2C1
54	38	-	Reserved
55	39	0x0000.00DC	CAN0
56-57	40-41	-	Reserved
58	42	0x0000.00E8	Ethernet Controller
59	43	0x0000.00EC	Hibernation Module

2.5.3 Exception Handlers

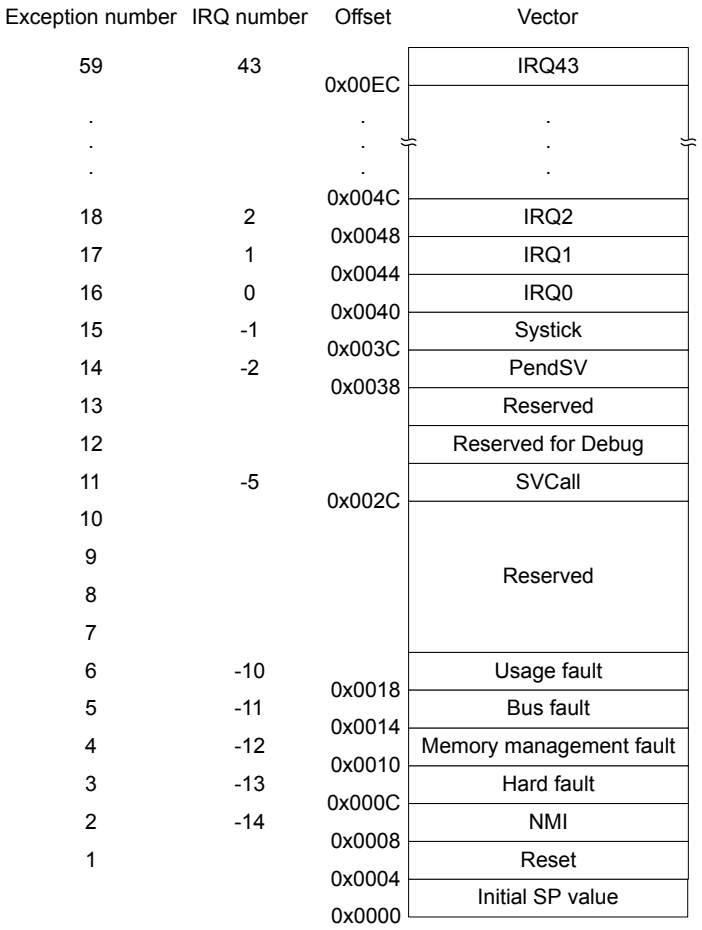
The processor handles exceptions using:

- **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 81. Figure 2-6 on page 84 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 0x0000.0100 to 0x3FFF.FF00 (see "Vector Table" on page 83). Note that when configuring the **VTABLE** register, the offset must be aligned on a 256-byte boundary.

2.5.5 Exception Priorities

As Table 2-8 on page 81 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 133 and page 119.

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 127.

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 85 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” on page 86 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 87 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

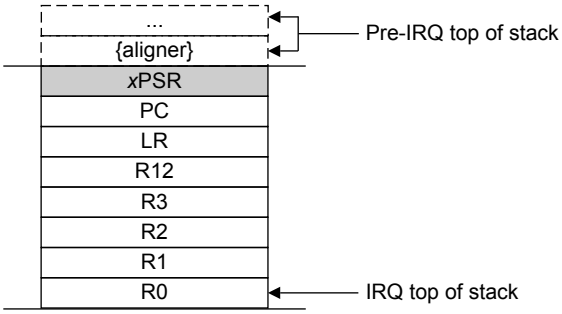
Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

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 67, **FAULTMASK** on page 68, and **BASEPRI** on page 69). 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. Unless stack alignment is disabled, the stack frame is aligned to a double-word address. If the STKALIGN bit of the **Configuration Control (CCR)** register is set, stack align adjustment is performed during stacking.

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 87 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. Exception return uses state from **MSP**. Execution uses **MSP** after return.
0xFFFF.FFF2 - 0xFFFF.FFF8	Reserved
0xFFFF.FFF9	Return to Thread mode. Exception return uses state from **MSP**. Execution uses **MSP** after return.
0xFFFF.FFFA - 0xFFFF.FFFC	Reserved
0xFFFF.FFFD	Return to Thread mode. Exception return uses state from **PSP**. 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 79). 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 88 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 140 for more information about the fault status registers.

Table 2-11. Faults

Fault	Handler	Fault Status Register	Bit Name
Bus error on a vector read	Hard fault	**Hard Fault Status (HFAULTSTAT)**	VECT
Fault escalated to a hard fault	Hard fault	**Hard Fault Status (HFAULTSTAT)**	FORCED
MPU or default memory mismatch on instruction access	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	IERR [a]
MPU or default memory mismatch on data access	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	DERR
MPU or default memory mismatch on exception stacking	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	MSTKE
MPU or default memory mismatch on exception unstacking	Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	MUSTKE
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 133). Software can disable execution of the handlers for these faults (see **SYSHNDCTRL** on page 136).

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 79.

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 89.

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 146
Memory management fault	**Memory Management Fault Status (MFAULTSTAT)**	**Memory Management Fault Address (MMADDR)**	page 140 page 147
Bus fault	**Bus Fault Status (BFAULTSTAT)**	**Bus Fault Address (FAULTADDR)**	page 140 page 148
Usage fault	**Usage Fault Status (UFAULTSTAT)**	-	page 140

2.6.4 Lockup

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

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

2.7 Power Management

The Cortex-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 129). 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 90). When the processor executes a WFI instruction, it stops executing instructions and enters sleep mode. See the *Cortex™-M3/M4 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/M4 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 all exception handlers, it returns to Thread mode and immediately enters sleep mode. This mechanism can be used in applications that only require the processor to run when an exception occurs.

2.7.2 Wake Up from Sleep Mode

The conditions for the processor to wake up depend on the mechanism that 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 the NVIC detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the PRIMASK bit and clearing the FAULTMASK bit. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears PRIMASK. For more information about **PRIMASK** and **FAULTMASK**, see page 67 and page 68.

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 129.

2.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 2-13 on page 91 lists the supported instructions.

Note: In Table 2-13 on page 91:

- 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/M4 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	-

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

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

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

Mnemonic	Operands	Brief Description	Flags
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	-
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	-
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 94)

  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 95)
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers

- System Control Block (SCB) (see page 97)

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

- Memory Protection Unit (MPU) (see page 97)

  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 94 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	94
0xE000.E100-0xE000.E4EF 0xE000.EF00-0xE000.EF03	Nested Vectored Interrupt Controller	95
0xE000.ED00-0xE000.ED3F	System Control Block	97
0xE000.ED90-0xE000.EDB8	Memory Protection Unit	97

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:

- 34 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 96 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 113 or **SWTRIG** on page 121.

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 73 for more information).

Table 3-2 on page 97 shows the possible MPU region attributes. See the section called “MPU Configuration for a Stellaris Microcontroller” on page 101 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 153) contains the required region number and has the VALID bit set. This method can be used when the data is statically packed, for example in a boot loader:

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

Subregions

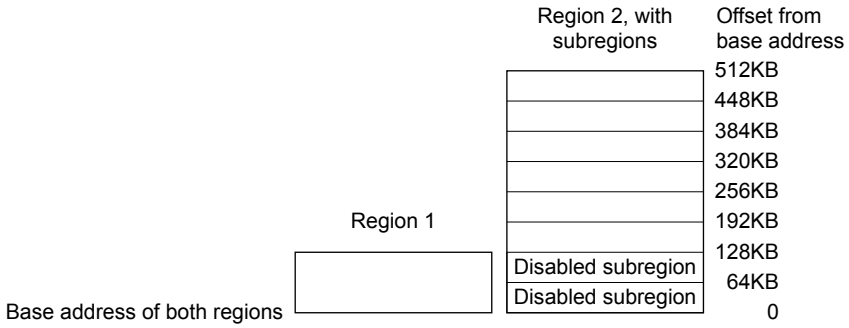
Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the SRD field of the **MPU Region Attribute and Size (MPUATTR)** register (see page 155) to disable a subregion. The least-significant bit of the SRD field controls the first subregion, and the most-significant 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 100 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 100 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 101 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	
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	-	-

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

TEX	S	C	B	Memory Type	Shareability	Other Attributes
1BB	0	A	A	Normal	Not shareable	Cached memory (BB = outer policy, AA = inner policy). 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 101 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 101 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 101.

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
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 71 for more information). The **MFAULTSTAT** register indicates the cause of the fault. See page 140 for more information.

3.2 Register Map

Table 3-7 on page 102 lists the Cortex-M3 Peripheral SysTick, NVIC, MPU and SCB registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000.E000.

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

Table 3-7. Peripherals Register Map

Offset	Name	Type	Reset	Description	See page
System Timer (SysTick) Registers					
0x010	STCTRL	R/W	0x0000.0000	SysTick Control and Status Register	105
0x014	STRELOAD	R/W	0x0000.0000	SysTick Reload Value Register	107
0x018	STCURRENT	R/WC	0x0000.0000	SysTick Current Value Register	108
Nested Vectored Interrupt Controller (NVIC) Registers					
0x100	EN0	R/W	0x0000.0000	Interrupt 0-31 Set Enable	109
0x104	EN1	R/W	0x0000.0000	Interrupt 32-43 Set Enable	110
0x180	DIS0	R/W	0x0000.0000	Interrupt 0-31 Clear Enable	111
0x184	DIS1	R/W	0x0000.0000	Interrupt 32-43 Clear Enable	112
0x200	PEND0	R/W	0x0000.0000	Interrupt 0-31 Set Pending	113
0x204	PEND1	R/W	0x0000.0000	Interrupt 32-43 Set Pending	114
0x280	UNPEND0	R/W	0x0000.0000	Interrupt 0-31 Clear Pending	115
0x284	UNPEND1	R/W	0x0000.0000	Interrupt 32-43 Clear Pending	116
0x300	ACTIVE0	RO	0x0000.0000	Interrupt 0-31 Active Bit	117
0x304	ACTIVE1	RO	0x0000.0000	Interrupt 32-43 Active Bit	118
0x400	PRI0	R/W	0x0000.0000	Interrupt 0-3 Priority	119
0x404	PRI1	R/W	0x0000.0000	Interrupt 4-7 Priority	119
0x408	PRI2	R/W	0x0000.0000	Interrupt 8-11 Priority	119
0x40C	PRI3	R/W	0x0000.0000	Interrupt 12-15 Priority	119
0x410	PRI4	R/W	0x0000.0000	Interrupt 16-19 Priority	119

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

Offset	Name	Type	Reset	Description	See page
0x414	PRI5	R/W	0x0000.0000	Interrupt 20-23 Priority	119
0x418	PRI6	R/W	0x0000.0000	Interrupt 24-27 Priority	119
0x41C	PRI7	R/W	0x0000.0000	Interrupt 28-31 Priority	119
0x420	PRI8	R/W	0x0000.0000	Interrupt 32-35 Priority	119
0x424	PRI9	R/W	0x0000.0000	Interrupt 36-39 Priority	119
0x428	PRI10	R/W	0x0000.0000	Interrupt 40-43 Priority	119
0xF00	SWTRIG	WO	0x0000.0000	Software Trigger Interrupt	121
System Control Block (SCB) Registers					
0xD00	CPUID	RO	0x411F.C231	CPU ID Base	122
0xD04	INTCTRL	R/W	0x0000.0000	Interrupt Control and State	123
0xD08	VTABLE	R/W	0x0000.0000	Vector Table Offset	126
0xD0C	APINT	R/W	0xFA05.0000	Application Interrupt and Reset Control	127
0xD10	SYSCTRL	R/W	0x0000.0000	System Control	129
0xD14	CFGCTRL	R/W	0x0000.0000	Configuration and Control	131
0xD18	SYSPRI1	R/W	0x0000.0000	System Handler Priority 1	133
0xD1C	SYSPRI2	R/W	0x0000.0000	System Handler Priority 2	134
0xD20	SYSPRI3	R/W	0x0000.0000	System Handler Priority 3	135
0xD24	SYSHNDCTRL	R/W	0x0000.0000	System Handler Control and State	136
0xD28	FAULTSTAT	R/W1C	0x0000.0000	Configurable Fault Status	140
0xD2C	HFAULTSTAT	R/W1C	0x0000.0000	Hard Fault Status	146
0xD34	MMADDR	R/W	-	Memory Management Fault Address	147
0xD38	FAULTADDR	R/W	-	Bus Fault Address	148
Memory Protection Unit (MPU) Registers					
0xD90	MPUTYPE	RO	0x0000.0800	MPU Type	149
0xD94	MPUCTRL	R/W	0x0000.0000	MPU Control	150
0xD98	MPUNUMBER	R/W	0x0000.0000	MPU Region Number	152
0xD9C	MPUBASE	R/W	0x0000.0000	MPU Region Base Address	153
0xDA0	MPUATTR	R/W	0x0000.0000	MPU Region Attribute and Size	155
0xDA4	MPUBASE1	R/W	0x0000.0000	MPU Region Base Address Alias 1	153
0xDA8	MPUATTR1	R/W	0x0000.0000	MPU Region Attribute and Size Alias 1	155
0xDAC	MPUBASE2	R/W	0x0000.0000	MPU Region Base Address Alias 2	153
0xDB0	MPUATTR2	R/W	0x0000.0000	MPU Region Attribute and Size Alias 2	155

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

Offset	Name	Type	Reset	Description	See page
0xDB4	MPUBASE3	R/W	0x0000.0000	MPU Region Base Address Alias 3	153
0xDB8	MPUATTR3	R/W	0x0000.0000	MPU Region Attribute and Size Alias 3	155

3.3 System Timer (SysTick) Register Descriptions

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

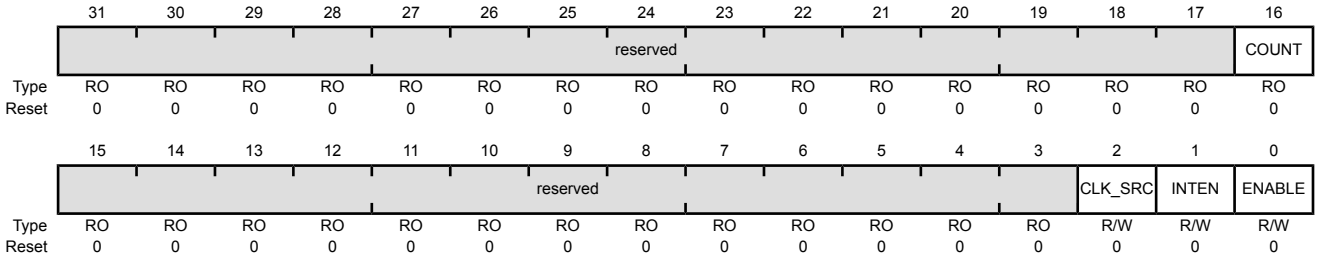
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010

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 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	COUNT	RO	0	Count Flag Value Description 0 The SysTick timer has not counted to 0 since the last time this bit was read. 1 The SysTick timer has counted to 0 since the last time this bit was read. This bit is cleared by a read of the register or if the **STCURRENT** register is written with any value. 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	0	Clock Source Value Description 0 External reference clock. (Not implemented for most Stellaris microcontrollers.) 1 System clock 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

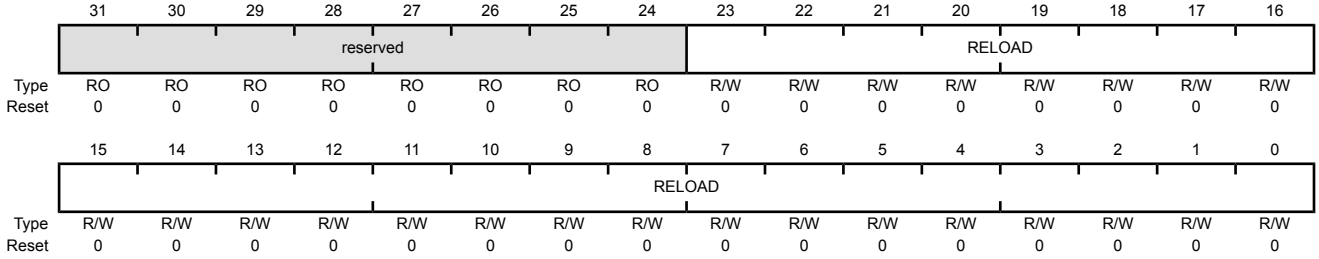
Note: This register can only be accessed from privileged mode.

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



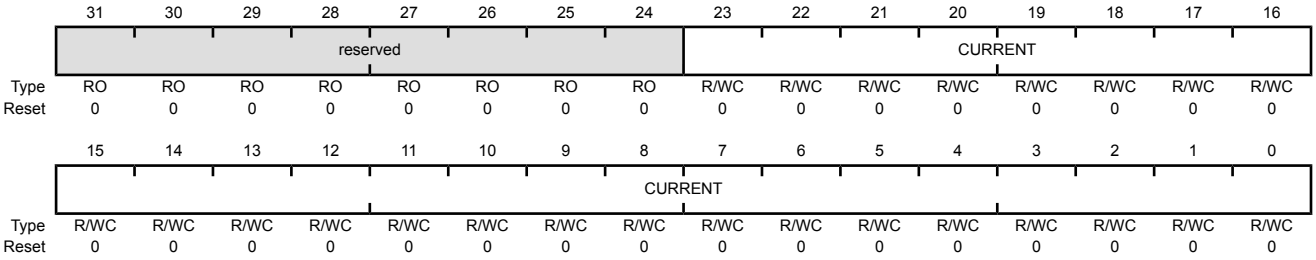
Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	RELOAD	R/W	0x00.0000	Reload Value Value to load 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 This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care. 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 126.

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

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

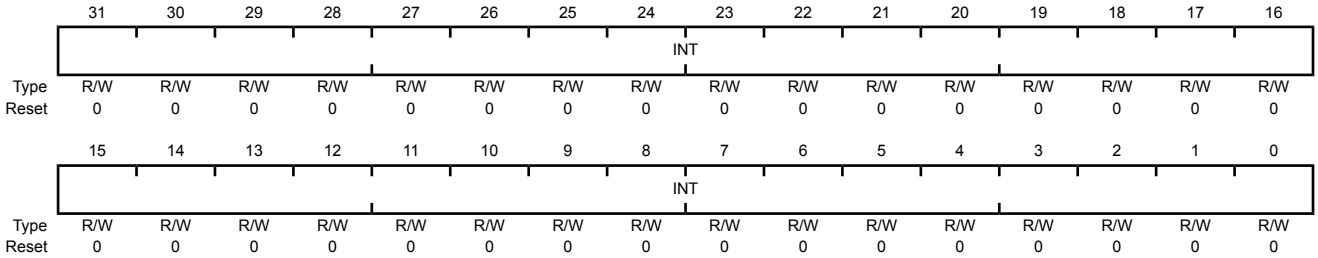
The **EN0** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 82 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. On a write, no effect.
1	On a read, indicates the interrupt is enabled. 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 32-43 Set Enable (EN1), offset 0x104

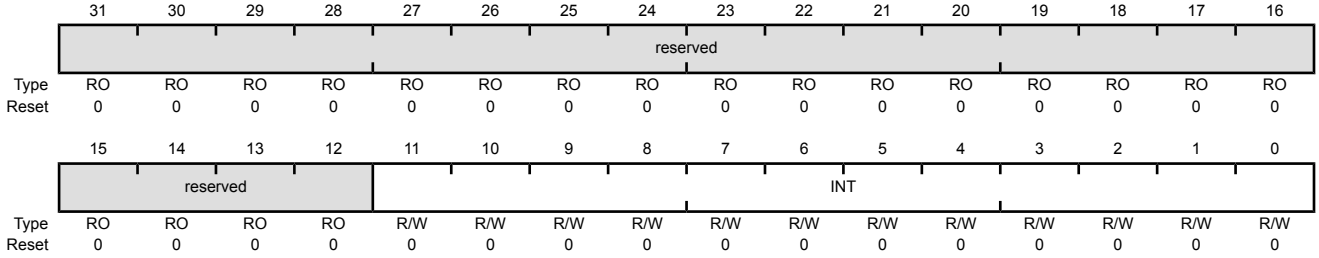
Note: This register can only be accessed from privileged mode.

The **EN1** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 32; bit 11 corresponds to Interrupt 43. See Table 2-9 on page 82 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 32-43 Set Enable (EN1)

Base 0xE000.E000
Offset 0x104
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.
11:0	INT	R/W	0x000	Interrupt Enable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. 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

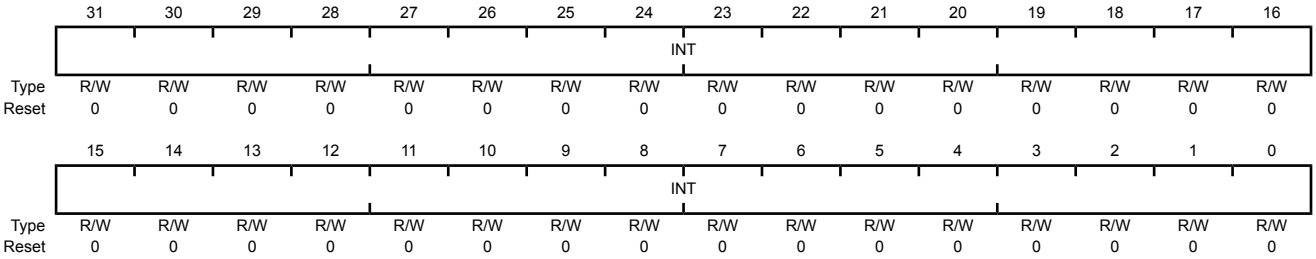
Note: This register can only be accessed from privileged mode.

The **DIS0** register disables interrupts. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 82 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. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding `INT[n]` bit in the **EN0** register, disabling interrupt [n].

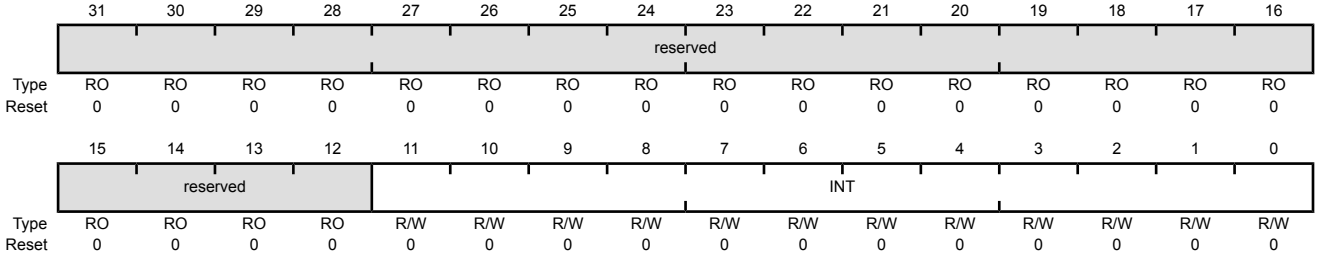
Register 7: Interrupt 32-43 Clear Enable (DIS1), offset 0x184

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

The **DIS1** register disables interrupts. Bit 0 corresponds to Interrupt 32; bit 11 corresponds to Interrupt 43. See Table 2-9 on page 82 for interrupt assignments.

Interrupt 32-43 Clear Enable (DIS1)

Base 0xE000.E000
Offset 0x184
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.
11:0	INT	R/W	0x000	Interrupt Disable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. 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

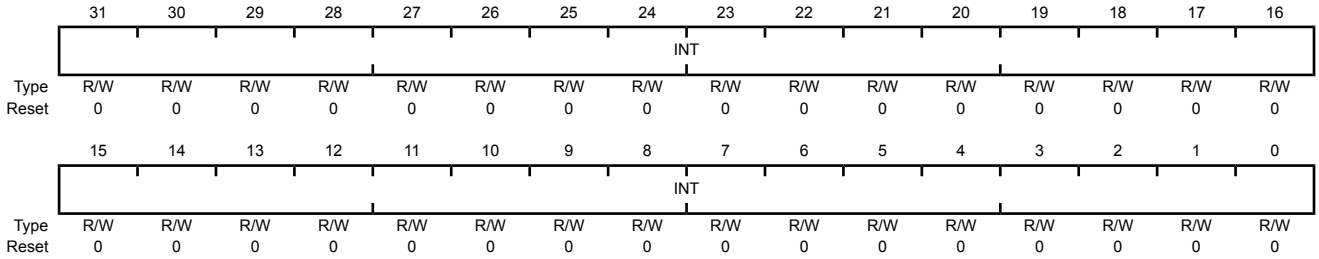
Note: This register can only be accessed from privileged mode.

The **PEND0** register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 82 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. On a write, no effect.
1	On a read, indicates that the interrupt is pending. 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.

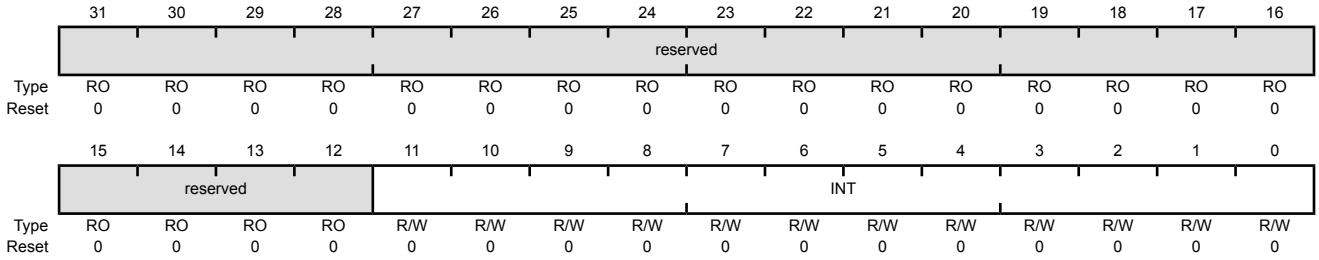
Register 9: Interrupt 32-43 Set Pending (PEND1), offset 0x204

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 11 corresponds to Interrupt 43. See Table 2-9 on page 82 for interrupt assignments.

Interrupt 32-43 Set Pending (PEND1)

Base 0xE000.E000
Offset 0x204
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.
11:0	INT	R/W	0x000	Interrupt Set Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. 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.

The **UNPEND0** register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 82 for interrupt assignments.

Interrupt 0-31 Clear Pending (UNPEND0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INT															
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
	INT															
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	INT	R/W	0x0000.0000	Interrupt Clear Pending

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

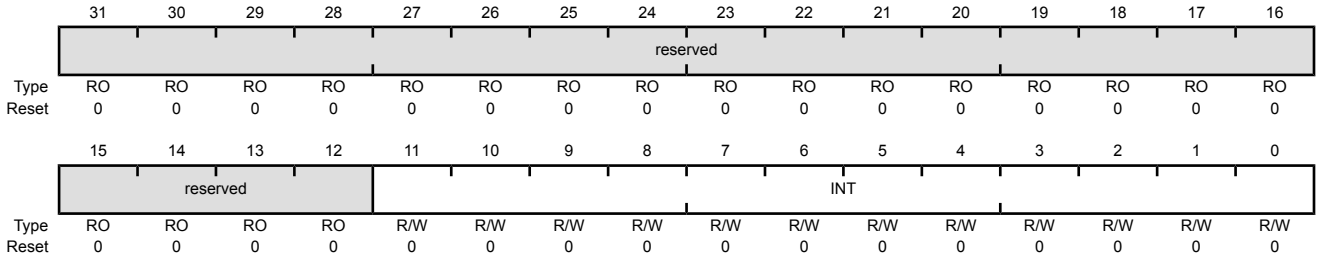
Register 11: Interrupt 32-43 Clear Pending (UNPEND1), offset 0x284

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 11 corresponds to Interrupt 43. See Table 2-9 on page 82 for interrupt assignments.

Interrupt 32-43 Clear Pending (UNPEND1)

Base 0xE000.E000
Offset 0x284
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.
11:0	INT	R/W	0x000	Interrupt Clear Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. On a write, clears the corresponding `INT[n]` bit in the **PEND1** register, so that interrupt [n] is no longer pending. 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.

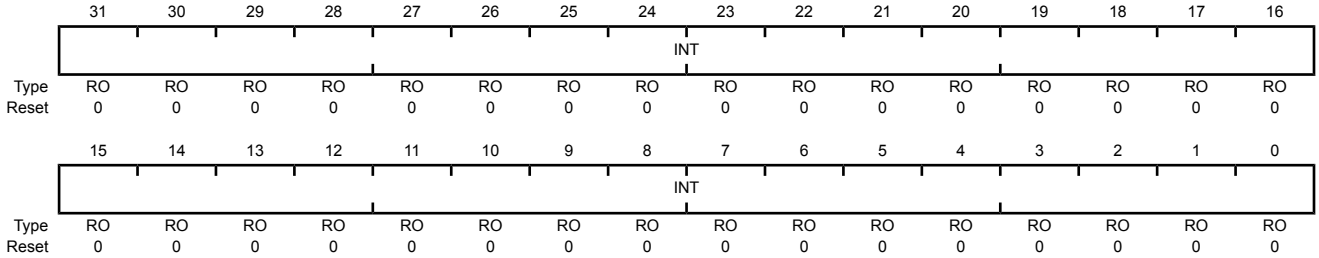
The **ACTIVE0** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 82 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 32-43 Active Bit (ACTIVE1), offset 0x304

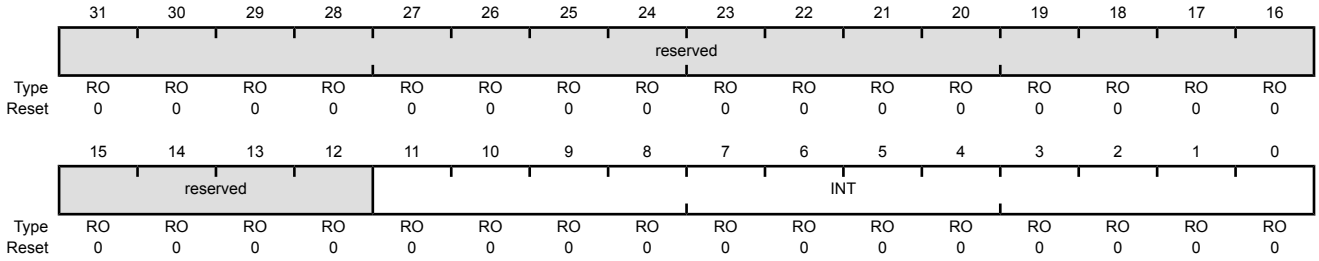
Note: This register can only be accessed from privileged mode.

The **ACTIVE1** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 32; bit 11 corresponds to Interrupt 43. See Table 2-9 on page 82 for interrupt assignments.

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

Interrupt 32-43 Active Bit (ACTIVE1)

Base 0xE000.E000
Offset 0x304
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.
11:0	INT	RO	0x000	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

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 82 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 127) 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	INTD			reserved					INTC			reserved				
Type	R/W	R/W	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W	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
	INTB			reserved					INTA			reserved				
Type	R/W	R/W	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W	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
31:29	INTD	R/W	0x0	Interrupt Priority for Interrupt [4n+3] 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] 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] 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] 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 25: Software Trigger Interrupt (SWTRIG), offset 0xF00

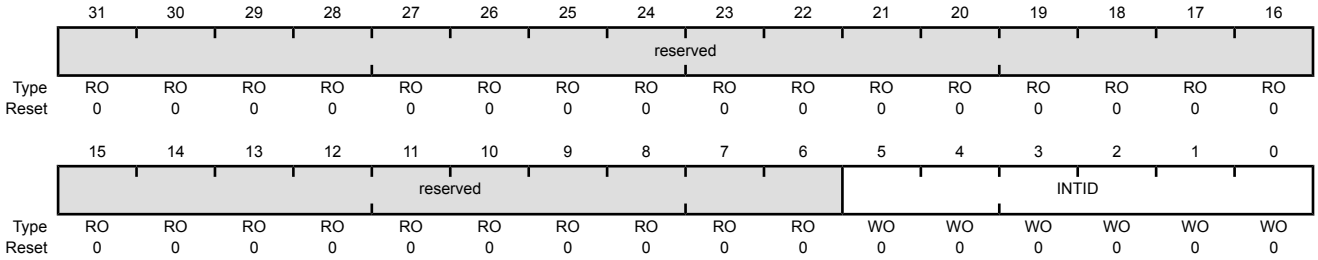
Note: Only privileged software can enable unprivileged access to the **SWTRIG** register.

Writing an interrupt number to the **SWTRIG** register generates a Software Generated Interrupt (SGI). See Table 2-9 on page 82 for interrupt assignments.

When the MAINPEND bit in the **Configuration and Control (CFGCTRL)** register (see page 131) 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 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.

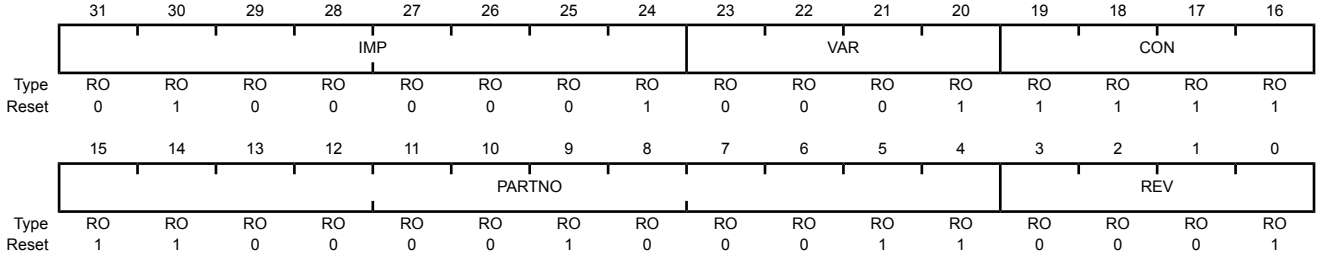
All registers must be accessed with aligned word accesses except for the **FAULTSTAT** and **SYSPRI1**-**SYSPRI3** registers, which can be accessed with byte or aligned halfword or word accesses. The processor does not support unaligned accesses to system control block registers.

Register 26: 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 0x411F.C231



Bit/Field	Name	Type	Reset	Description
31:24	IMP	RO	0x41	Implementer Code Value Description 0x41 ARM
23:20	VAR	RO	0x1	Variant Number Value Description 0x1 The rn value in the rnpn product revision identifier, for example, the 1 in r1p1.
19:16	CON	RO	0xF	Constant Value Description 0xF Always reads as 0xF.
15:4	PARTNO	RO	0xC23	Part Number Value Description 0xC23 Cortex-M3 processor.
3:0	REV	RO	0x1	Revision Number Value Description 0x1 The pn value in the rnpn product revision identifier, for example, the 1 in r1p1.

Register 27: Interrupt Control and State (INTCTRL), offset 0xD04

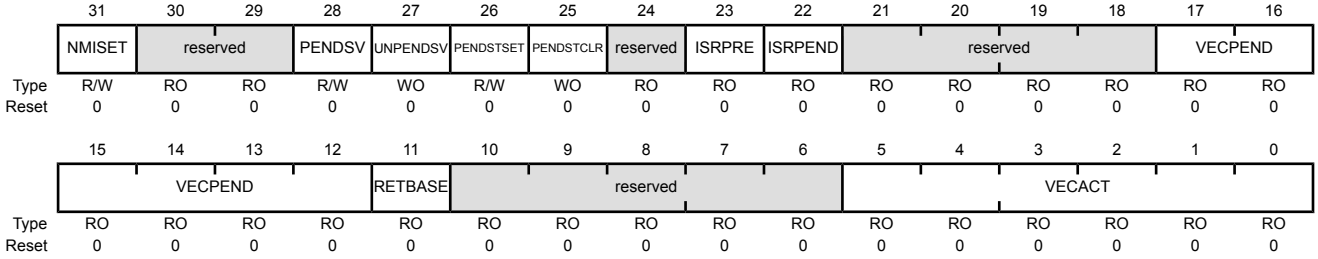
Note: This register can only be accessed from privileged mode.

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 Value Description 0 On a read, indicates an NMI exception is not pending. On a write, no effect. 1 On a read, indicates an NMI exception is pending. On a write, changes the NMI exception state to pending. 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 Value Description 0 On a read, indicates a PendSV exception is not pending. On a write, no effect. 1 On a read, indicates a PendSV exception is pending. On a write, changes the PendSV exception state to pending. 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 Value Description 0 On a write, no effect. 1 On a write, removes the pending state from the PendSV exception. This bit is write only; on a register read, its value is unknown.
26	PENDSTSET	R/W	0	SysTick Set Pending Value Description 0 On a read, indicates a SysTick exception is not pending. On a write, no effect. 1 On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending. This bit is cleared by writing a 1 to the PENDSTCLR bit.
25	PENDSTCLR	WO	0	SysTick Clear Pending Value Description 0 On a write, no effect. 1 On a write, removes the pending state from the SysTick exception. 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 Value Description 0 The release from halt does not take an interrupt. 1 The release from halt takes an interrupt. 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 Value Description 0 No interrupt is pending. 1 An interrupt is pending. This bit provides status for all interrupts excluding NMI and Faults.
21:18	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
17:12	VECPEND	RO	0x00	Interrupt Pending Vector Number 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. Value / Description 0x00 No exceptions are pending 0x01 Reserved 0x02 NMI 0x03 Hard fault 0x04 Memory management fault 0x05 Bus fault 0x06 Usage fault 0x07-0x0A Reserved 0x0B SVCall 0x0C Reserved for Debug 0x0D Reserved 0x0E PendSV 0x0F SysTick 0x10 Interrupt Vector 0 0x11 Interrupt Vector 1 ... ... 0x3B Interrupt Vector 43 0x3C-0x3F Reserved
11	RETBASE	RO	0	Return to Base Value / Description 0 There are preempted active exceptions to execute. 1 There are no active exceptions, or the currently executing exception is the only active exception. 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).
10:6	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.
5:0	VECACT	RO	0x00	Interrupt Pending Vector Number 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. 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 63).

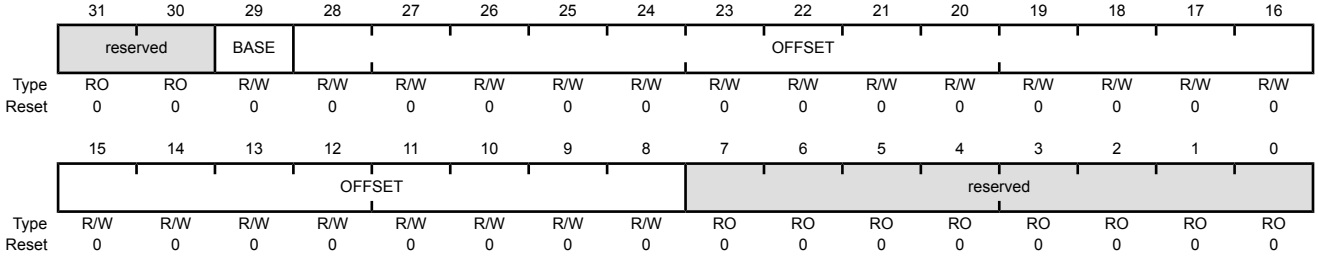
Register 28: Vector Table Offset (VTABLE), offset 0xD08

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 Value Description 0 The vector table is in the code memory region. 1 The vector table is in the SRAM memory region.
28:8	OFFSET	R/W	0x000.00	Vector Table Offset When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 43 interrupts, the offset must be aligned on a 256-byte boundary.
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 29: 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 127 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	VECTKEY															
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	0	1	0	0	0	0	0	0	1	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ENDIANESS	reserved				PRIGROUP			reserved					SYSRESREQ	VECTCLRACT	VECTRESET
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	RO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	VECTKEY	R/W	0xFA05	Register Key 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 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.

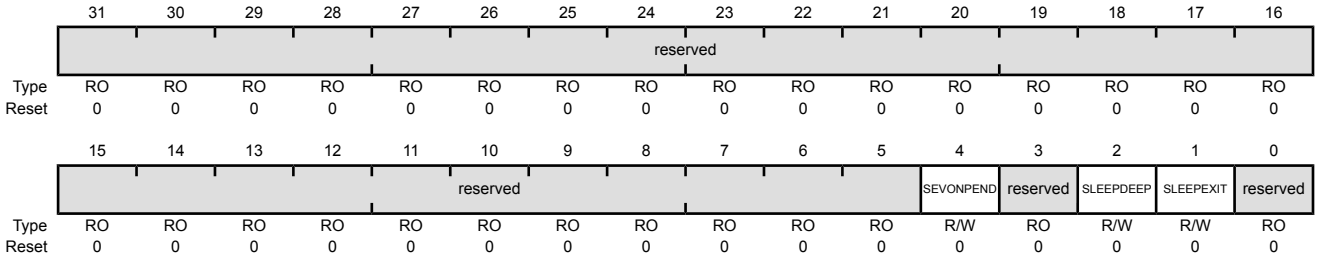
Bit/Field	Name	Type	Reset	Description
10:8	PRIGROUP	R/W	0x0	Interrupt Priority Grouping This field determines the split of group priority from subpriority (see Table 3-8 on page 127 for more information).
7:3	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility 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 Value Description 0 No effect. 1 Resets the core and all on-chip peripherals except the Debug interface. This bit is automatically cleared during the reset of the core and reads as 0.
1	VECTCLRACT	WO	0	Clear Active NMI / Fault 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 This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.

Register 30: 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 Value Description 0 Only enabled interrupts or events can wake up the processor; disabled interrupts are excluded. 1 Enabled events and all interrupts, including disabled interrupts, can wake up the processor. 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. 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 Value Description 0 Use Sleep mode as the low power mode. 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 Value Description 0 When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode. 1 When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR. 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.

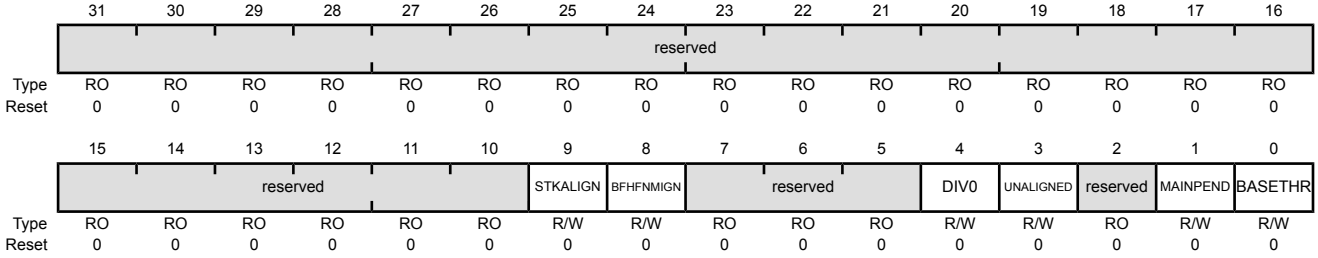
Register 31: Configuration and Control (CFGCTRL), offset 0xD14

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 121).

Configuration and Control (CFGCTRL)

Base 0xE000.E000
Offset 0xD14
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	STKALIGN	R/W	0	Stack Alignment on Exception Entry Value Description 0 The stack is 4-byte aligned. 1 The stack is 8-byte aligned. 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 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. Value Description 0 Data bus faults caused by load and store instructions cause a lock-up. 1 Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions. 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 This bit enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0. Value Description 0 Do not trap on divide by 0. A divide by zero returns a quotient of 0. 1 Trap on divide by 0.
3	UNALIGNED	R/W	0	Trap on Unaligned Access Value Description 0 Do not trap on unaligned halfword and word accesses. 1 Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault. 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 Value Description 0 Disables unprivileged software access to the **SWTRIG** register. 1 Enables unprivileged software access to the **SWTRIG** register (see page 121).
0	BASETHR	R/W	0	Thread State Control Value Description 0 The processor can enter Thread mode only when no exception is active. 1 The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see “Exception Return” on page 87 for more information).

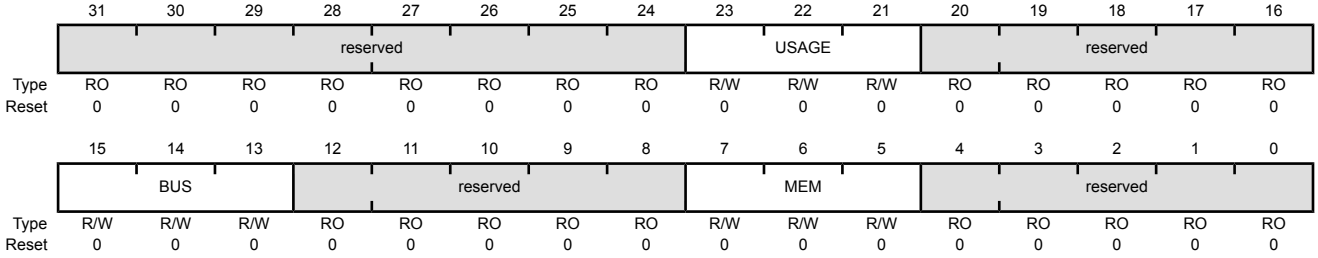
Register 32: System Handler Priority 1 (SYSPRI1), offset 0xD18

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 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 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 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.

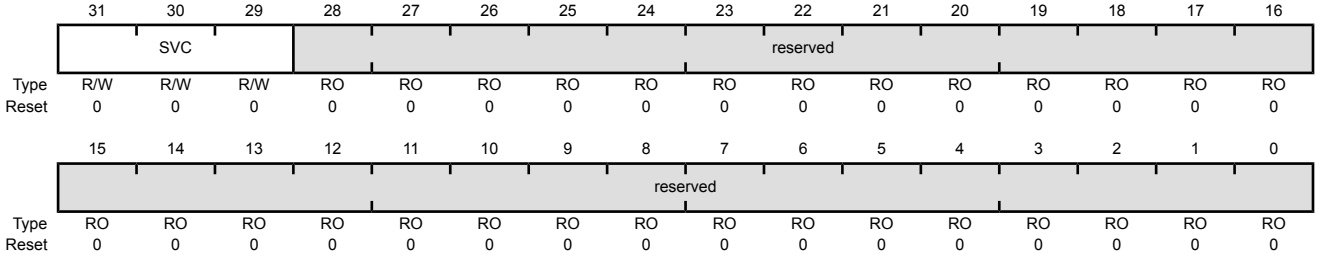
Register 33: System Handler Priority 2 (SYSPRI2), offset 0xD1C

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 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.

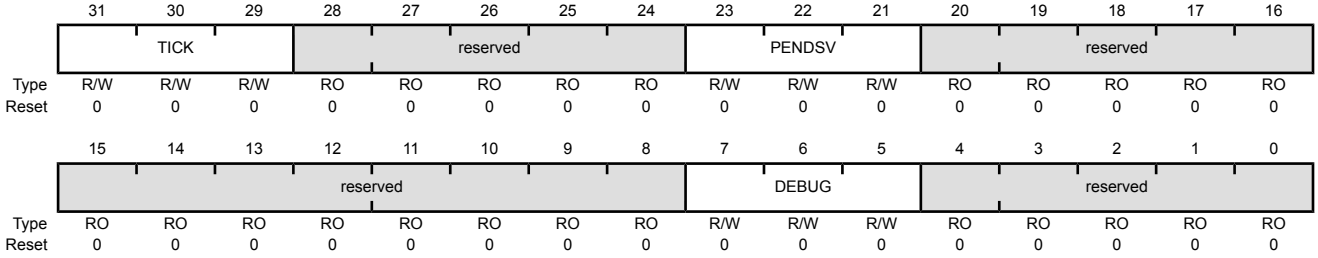
Register 34: System Handler Priority 3 (SYSPRI3), offset 0xD20

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 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 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 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 35: 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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved													USAGE	BUS	MEM
Type RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
SVC	BUSP	MEMP	USAGEP	TICK	PNDSV	reserved	MON	SVCA	reserved			USGA	reserved	BUSA	MEMA
Type R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	RO	RO	RO	R/W	RO	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
31:19	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 Value Description 0 Disables the usage fault exception. 1 Enables the usage fault exception.
17	BUS	R/W	0	Bus Fault Enable Value Description 0 Disables the bus fault exception. 1 Enables the bus fault exception.

Bit/Field	Name	Type	Reset	Description
16	MEM	R/W	0	Memory Management Fault Enable Value Description 0 Disables the memory management fault exception. 1 Enables the memory management fault exception.
15	SVC	R/W	0	SVC Call Pending Value Description 0 An SVC call exception is not pending. 1 An SVC call exception is pending. This bit can be modified to change the pending status of the SVC call exception.
14	BUSP	R/W	0	Bus Fault Pending Value Description 0 A bus fault exception is not pending. 1 A bus fault exception is pending. This bit can be modified to change the pending status of the bus fault exception.
13	MEMP	R/W	0	Memory Management Fault Pending Value Description 0 A memory management fault exception is not pending. 1 A memory management fault exception is pending. This bit can be modified to change the pending status of the memory management fault exception.
12	USAGEP	R/W	0	Usage Fault Pending Value Description 0 A usage fault exception is not pending. 1 A usage fault exception is pending. This bit can be modified to change the pending status of the usage fault exception.
11	TICK	R/W	0	SysTick Exception Active Value Description 0 A SysTick exception is not active. 1 A SysTick exception is active. 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 Value Description 0 A PendSV exception is not active. 1 A PendSV exception is active. 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 Value Description 0 The Debug monitor is not active. 1 The Debug monitor is active.
7	SVCA	R/W	0	SVC Call Active Value Description 0 SVC call is not active. 1 SVC call is active. 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 Value Description 0 Usage fault is not active. 1 Usage fault is active. 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 Value Description 0 Bus fault is not active. 1 Bus fault is active. 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 36: 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 Value Description 0 No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled. 1 The processor has executed an SDIV or UDIV instruction with a divisor of 0. When this bit is set, the **PC** value stacked for the exception return points to the instruction that performed the divide by zero. Trapping on divide-by-zero is enabled by setting the DIV0 bit in the **Configuration and Control (CFGCTRL)** register (see page 131). This bit is cleared by writing a 1 to it.
24	UNALIGN	R/W1C	0	Unaligned Access Usage Fault Value Description 0 No unaligned access fault has occurred, or unaligned access trapping is not enabled. 1 The processor has made an unaligned memory access. Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of the configuration of this bit. Trapping on unaligned access is enabled by setting the UNALIGNED bit in the **CFGCTRL** register (see page 131). 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 Value Description 0 A usage fault has not been caused by attempting to access a coprocessor. 1 The processor has attempted to access a coprocessor. This bit is cleared by writing a 1 to it.
18	INVPC	R/W1C	0	Invalid PC Load Usage Fault Value Description 0 A usage fault has not been caused by attempting to load an invalid **PC** value. 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. 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**. 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 Value Description 0 A usage fault has not been caused by an invalid state. 1 The processor has attempted to execute an instruction that makes illegal use of the **EPSR** register. 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. This bit is not set if an undefined instruction uses the **EPSR** register. This bit is cleared by writing a 1 to it.
16	UNDEF	R/W1C	0	Undefined Instruction Usage Fault Value Description 0 A usage fault has not been caused by an undefined instruction. 1 The processor has attempted to execute an undefined instruction. When this bit is set, the **PC** value stacked for the exception return points to the undefined instruction. An undefined instruction is an instruction that the processor cannot decode. This bit is cleared by writing a 1 to it.
15	BFARV	R/W1C	0	Bus Fault Address Register Valid Value Description 0 The value in the **Bus Fault Address (FAULTADDR)** register is not a valid fault address. 1 The **FAULTADDR** register is holding a valid fault address. 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. 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. 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 Value Description 0 No bus fault has occurred on stacking for exception entry. 1 Stacking for an exception entry has caused one or more bus faults. 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. This bit is cleared by writing a 1 to it.
11	BUSTKE	R/W1C	0	Unstack Bus Fault Value Description 0 No bus fault has occurred on unstacking for a return from exception. 1 Unstacking for a return from exception has caused one or more bus faults. 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. This bit is cleared by writing a 1 to it.
10	IMPRE	R/W1C	0	Imprecise Data Bus Error Value Description 0 An imprecise data bus error has not occurred. 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. When this bit is set, a fault address is not written to the **FAULTADDR** register. 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. This bit is cleared by writing a 1 to it.
9	PRECISE	R/W1C	0	Precise Data Bus Error Value Description 0 A precise data bus error has not occurred. 1 A data bus error has occurred, and the **PC** value stacked for the exception return points to the instruction that caused the fault. When this bit is set, the fault address is written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
8	IBUS	R/W1C	0	Instruction Bus Error Value Description 0 An instruction bus error has not occurred. 1 An instruction bus error has occurred. The processor detects the instruction bus error on prefetching an instruction, but sets this bit only if it attempts to issue the faulting instruction. When this bit is set, a fault address is not written to the **FAULTADDR** register. This bit is cleared by writing a 1 to it.
7	MMARV	R/W1C	0	Memory Management Fault Address Register Valid Value Description 0 The value in the **Memory Management Fault Address (MMADDR)** register is not a valid fault address. 1 The **MMADDR** register is holding a valid fault address. 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. 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 Value Description 0 No memory management fault has occurred on stacking for exception entry. 1 Stacking for an exception entry has caused one or more access violations. 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. This bit is cleared by writing a 1 to it.

Bit/Field	Name	Type	Reset	Description
3	MUSTKE	R/W1C	0	Unstack Access Violation Value Description 0 No memory management fault has occurred on unstacking for a return from exception. 1 Unstacking for a return from exception has caused one or more access violations. 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. 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 Value Description 0 A data access violation has not occurred. 1 The processor attempted a load or store at a location that does not permit the operation. 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. This bit is cleared by writing a 1 to it.
0	IERR	R/W1C	0	Instruction Access Violation Value Description 0 An instruction access violation has not occurred. 1 The processor attempted an instruction fetch from a location that does not permit execution. This fault occurs on any access to an XN region, even when the MPU is disabled or not present. 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. This bit is cleared by writing a 1 to it.

Register 37: Hard Fault Status (HFAULTSTAT), offset 0xD2C

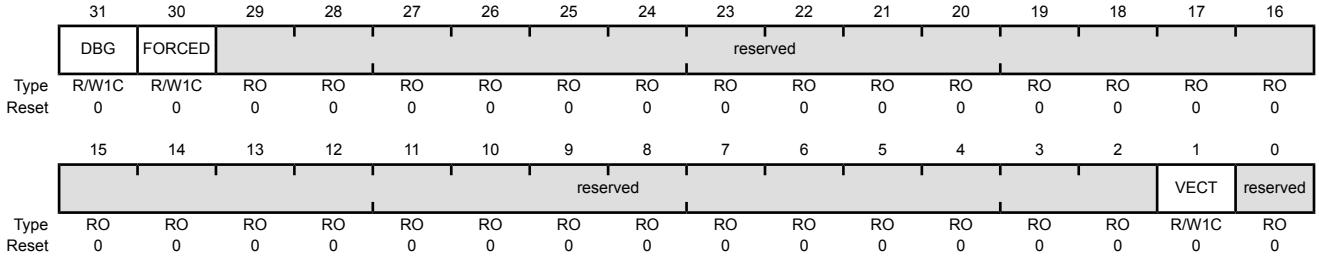
Note: This register can only be accessed from privileged mode.

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 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 Value Description 0 No forced hard fault has occurred. 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. When this bit is set, the hard fault handler must read the other fault status registers to find the cause of the fault. 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 Value Description 0 No bus fault has occurred on a vector table read. 1 A bus fault occurred on a vector table read. This error is always handled by the hard fault handler. When this bit is set, the **PC** value stacked for the exception return points to the instruction that was preempted by the exception. 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.

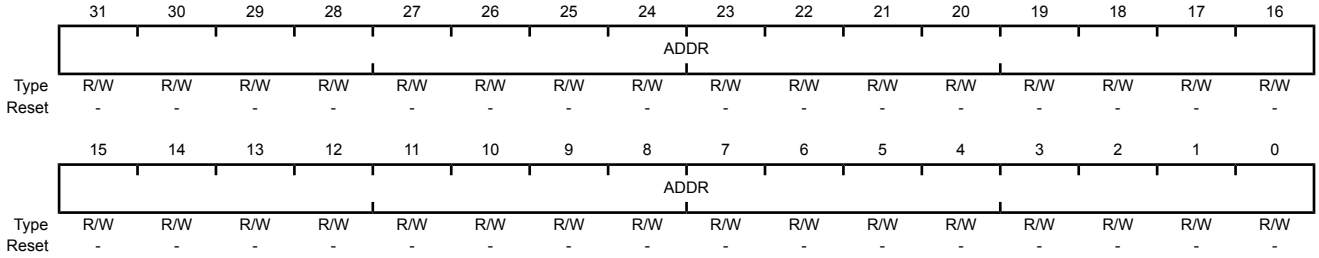
Register 38: Memory Management Fault Address (MMADDR), offset 0xD34

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 140).

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 When the MMARV bit of **MFAULTSTAT** is set, this field holds the address of the location that generated the memory management fault.

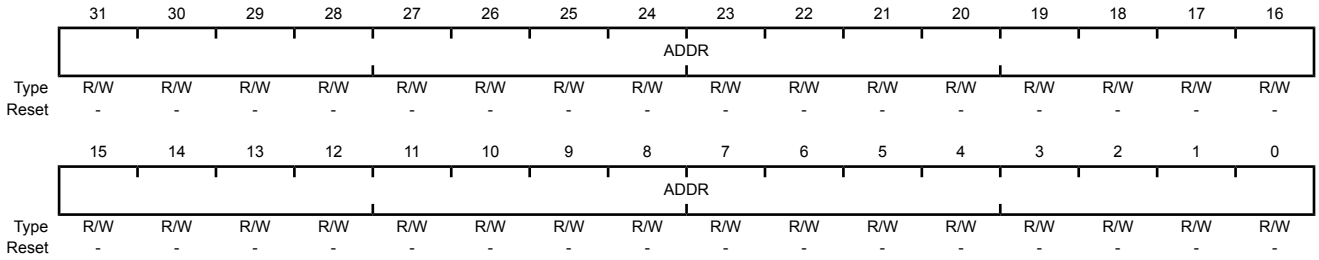
Register 39: Bus Fault Address (FAULTADDR), offset 0xD38

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 140).

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 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.

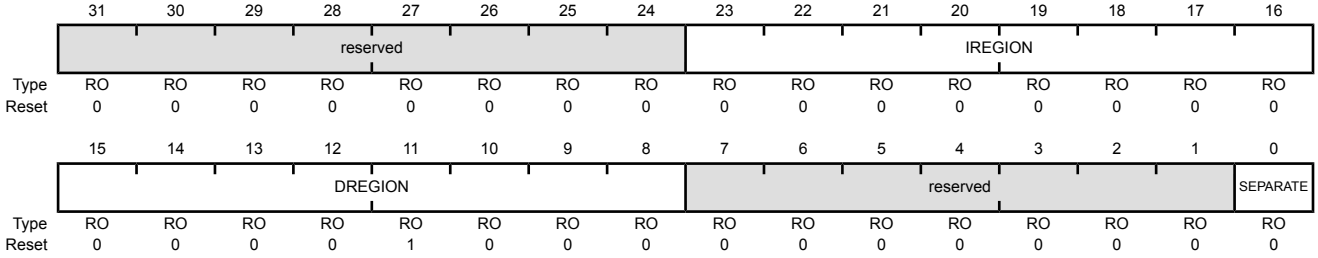
The MPU registers can only be accessed from privileged mode.

Register 40: 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 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 Value Description 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 Value Description 0 Indicates the MPU is unified.

Register 41: 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 71. 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.

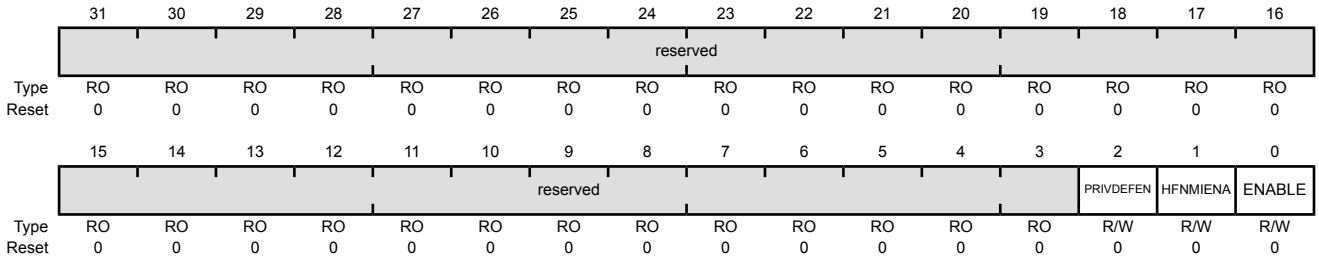
When the ENABLE bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see Table 2-5 on page 73 for more information). The default memory map applies to accesses from both privileged and unprivileged software.

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 This bit enables privileged software access to the default memory map. Value Description 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. 1 If the MPU is enabled, this bit enables use of the default memory map as a background region for privileged software accesses. 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. If the MPU is disabled, the processor ignores this bit.
1	HFNMIENA	R/W	0	MPU Enabled During Faults This bit controls the operation of the MPU during hard fault, NMI, and **FAULTMASK** handlers. Value Description 0 The MPU is disabled during hard fault, NMI, and **FAULTMASK** handlers, regardless of the value of the ENABLE bit. 1 The MPU is enabled during hard fault, NMI, and **FAULTMASK** handlers. When the MPU is disabled and this bit is set, the resulting behavior is unpredictable.
0	ENABLE	R/W	0	MPU Enable Value Description 0 The MPU is disabled. 1 The MPU is enabled. When the MPU is disabled and the HFNMIENA bit is set, the resulting behavior is unpredictable.

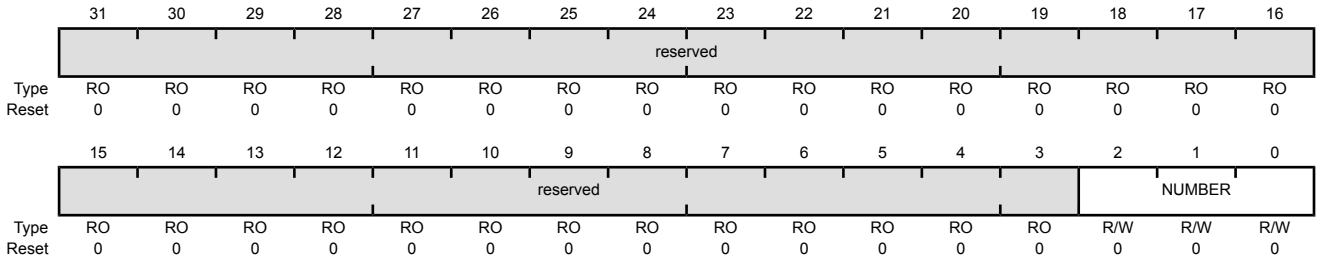
Register 42: MPU Region Number (MPUNUMBER), offset 0xD98

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 153). 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 This field indicates the MPU region referenced by the **MPUBASE** and **MPUATTR** registers. The MPU supports eight memory regions.

Register 43: MPU Region Base Address (MPUBASE), offset 0xD9C

Register 44: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4

Register 45: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC

Register 46: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4

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

The **MPUBASE** register defines the base address of the MPU region selected by the **MPU Region Number (MPUNUMBER)** register and can update the value of the **MPUNUMBER** register. To change the current region number and update the **MPUNUMBER** register, write the **MPUBASE** register with the VALID bit set.

The ADDR field is bits 31:*N* of the **MPUBASE** register. Bits (*N*-1):5 are reserved. The region size, as specified by the SIZE field in the **MPU Region Attribute and Size (MPUATTR)** register, defines the value of *N* where:

$N = Log_2(\text{Region size in bytes})$

If the region size is configured to 4 GB in the **MPUATTR** register, there is no valid ADDR field. In this case, the region occupies the complete memory map, and the base address is 0x0000.0000.

The base address is aligned to the size of the region. For example, a 64-KB region must be aligned on a multiple of 64 KB, for example, at 0x0001.0000 or 0x0002.0000.

MPU Region Base Address (MPUBASE)

Base 0xE000.E000
Offset 0xD9C
Type R/W, reset 0x0000.0000

Bit	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

Bit	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 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. Software should not rely on the value of 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 Value Description 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. 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. 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 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 47: MPU Region Attribute and Size (MPUATTR), offset 0xDA0

Register 48: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8

Register 49: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0

Register 50: 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 155 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 153).

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 Value Description 0 Instruction fetches are enabled. 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 For information on using this bit field, see Table 3-5 on page 101.
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 For information on using this bit field, see Table 3-3 on page 100.
18	S	R/W	0	Shareable For information on using this bit, see Table 3-3 on page 100.
17	C	R/W	0	Cacheable For information on using this bit, see Table 3-3 on page 100.
16	B	R/W	0	Bufferable For information on using this bit, see Table 3-3 on page 100.
15:8	SRD	R/W	0x00	Subregion Disable Bits Value Description 0 The corresponding subregion is enabled. 1 The corresponding subregion is disabled. 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 99 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 The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register. Refer to Table 3-9 on page 155 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 five pins: $\overline{\text{TRST}}$, 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. This is implemented 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 outputs. The multiplexer is controlled by the Stellaris JTAG controller, which has comprehensive programming for the ARM, Stellaris, and unimplemented JTAG instructions.

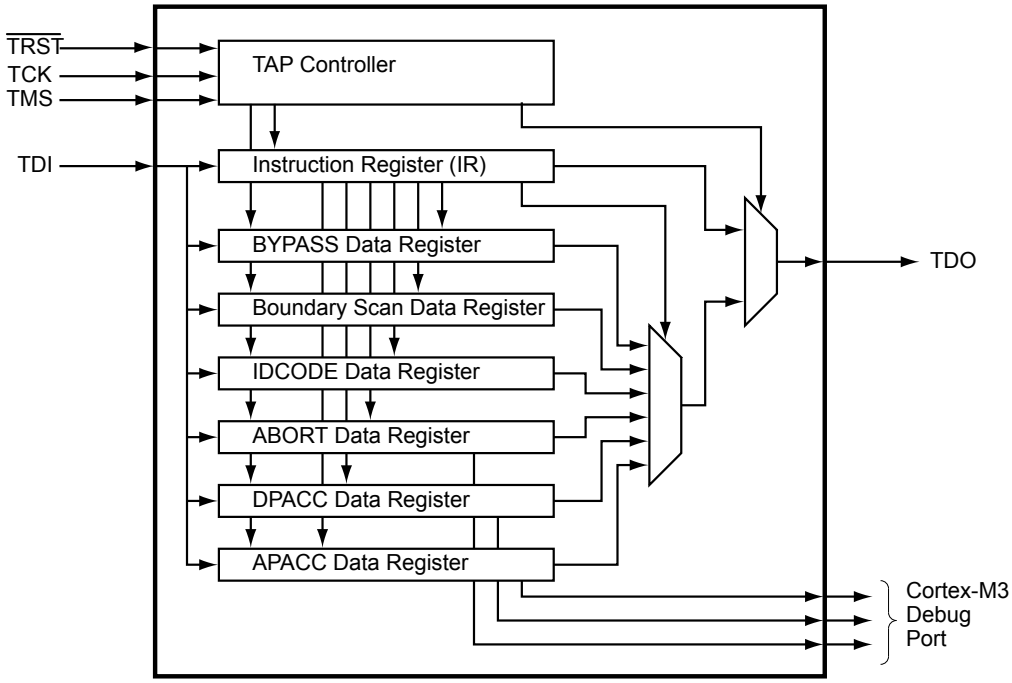
The Stellaris JTAG module has the following features:

- 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)

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

Table 4-1 on page 159 and Table 4-2 on page 160 list the external signals of the JTAG/SWD controller and describe 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 294. The column in the table below titled "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 308) is set to choose the JTAG/SWD function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286.

Table 4-1. JTAG_SWD_SWO Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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/O	TTL	JTAG TMS and SWDIO.
TRST	89	I	TTL	JTAG TRST.

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

Table 4-2. JTAG_SWD_SWO Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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/O	TTL	JTAG TMS and SWDIO.
$\overline{\text{TRST}}$	A8	I	TTL	JTAG $\overline{\text{TRST}}$.

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 159. 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 $\overline{\text{TRST}}$, TCK and TMS inputs. The current state of the TAP controller depends on the current value of $\overline{\text{TRST}}$ and 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 167 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 694 for JTAG timing diagrams.

4.3.1 JTAG Interface Pins

The JTAG interface consists of five standard pins: $\overline{\text{TRST}}$,TCK, TMS, TDI, and TDO. These pins and their associated reset state are given in Table 4-3 on page 160. Detailed information on each pin follows.

Table 4-3. JTAG Port Pins Reset State

Pin Name	Data Direction	Internal Pull-Up	Internal Pull-Down	Drive Strength	Drive Value
$\overline{\text{TRST}}$	Input	Enabled	Disabled	N/A	N/A
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 Reset Input (TRST)

The TRST pin is an asynchronous active Low input signal for initializing and resetting the JTAG TAP controller and associated JTAG circuitry. When TRST is asserted, the TAP controller resets to the Test-Logic-Reset state and remains there while TRST is asserted. When the TAP controller enters the Test-Logic-Reset state, the JTAG Instruction Register (IR) resets to the default instruction, IDCODE.

By default, the internal pull-up resistor on the TRST pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port B should ensure that the internal pull-up resistor remains enabled on PB7/TRST; otherwise JTAG communication could be lost.

4.3.1.2 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. In addition, it ensures 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. This assures 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.

4.3.1.3 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 is 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 Instruction Register (IR) resets to the default instruction, IDCODE. Therefore, this sequence can be used as a reset mechanism, similar to asserting TRST. The JTAG Test Access Port state machine can be seen in its entirety in Figure 4-2 on page 163.

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.

4.3.1.4 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, presents 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.

4.3.1.5 Test Data Output (TDO)

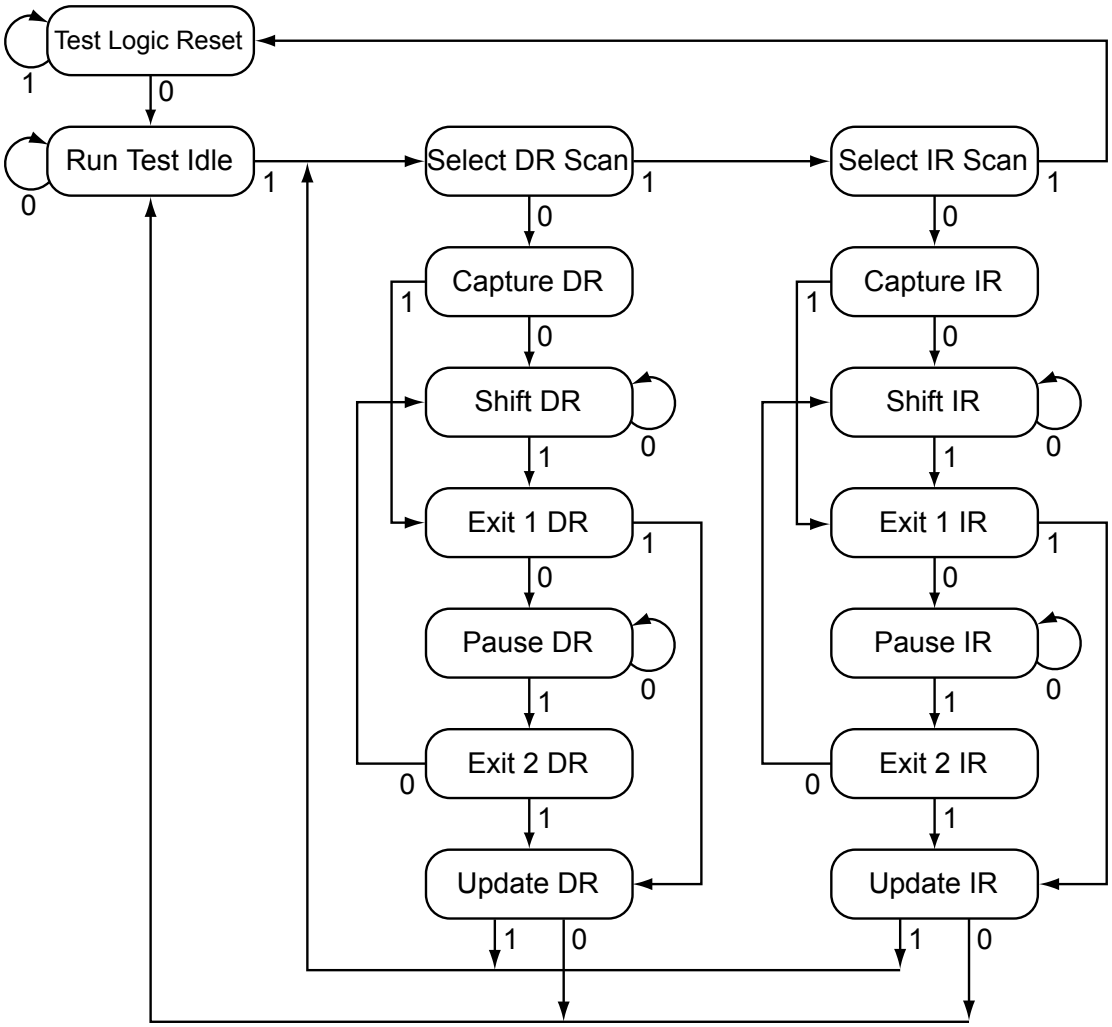
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an 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. This assures 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.

4.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 4-2 on page 163. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR) or the assertion of TRST. 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 of 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 166.

4.3.4 Operational Considerations

There are certain operational considerations 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 controller is reset with either a POR or RST, the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (setting **GPIODEN** to 1), enabling the pull-up resistors (setting **GPIOPUR** to 1), and enabling the alternate hardware function (setting **GPIOAFSEL** to 1) for the PB7 and PC[3:0] JTAG/SWD pins.

It is possible for software to configure these pins as GPIOs after reset by writing 0s to PB7 and PC[3:0] in the **GPIOAFSEL** register. If the user does not require the JTAG/SWD port for debugging or board-level testing, this provides five 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. This may lock the debugger out of the part. This 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 currently provided for the five JTAG/SWD pins (PB7 and PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 308) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 318) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 319) have been set to 1.

Recovering a "Locked" Device

Note: The mass erase of the flash memory caused by the below sequence erases the entire flash memory, regardless of the settings in the **Flash Memory Protection Program Enable n (FMPPEn)** registers. Performing the sequence below does not affect the nonvolatile registers discussed in “Nonvolatile Register Programming” on page 263.

If software configures any of the JTAG/SWD pins as GPIO and loses the ability to communicate with the debugger, there is a debug sequence that can be used to recover the device. Performing a total of ten JTAG-to-SWD and SWD-to-JTAG switch sequences while holding the device in reset mass erases the flash memory. The sequence to recover the device is:

1. Assert and hold the RST signal.
2. Apply power to the device.
3. Perform the JTAG-to-SWD switch sequence.
4. Perform the SWD-to-JTAG switch sequence.
5. Perform the JTAG-to-SWD switch sequence.
6. Perform the SWD-to-JTAG switch sequence.
7. Perform the JTAG-to-SWD switch sequence.
8. Perform the SWD-to-JTAG switch sequence.
9. Perform the JTAG-to-SWD switch sequence.
10. Perform the SWD-to-JTAG switch sequence.

11. Perform the JTAG-to-SWD switch sequence.

12. Perform the SWD-to-JTAG switch sequence.

13. Release the RST signal.

14. Wait 400 ms.

15. Power-cycle the device.

The JTAG-to-SWD and SWD-to-JTAG switch sequences are described in "ARM Serial Wire Debug (SWD)" on page 165. When performing switch sequences for the purpose of recovering the debug capabilities of the device, only steps 1 and 2 of the switch sequence in the section called "JTAG-to-SWD Switching" on page 165 must be performed.

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 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 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 sequences 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 is the only instance where the ARM JTAG TAP controller does not meet full compliance with the specification. Due to the low probability of this sequence occurring during normal operation of the TAP controller, it should not affect normal performance of the JTAG interface.

JTAG-to-SWD Switching

To switch the operating mode of the Debug Access Port (DAP) from JTAG to SWD mode, the external debug hardware must send the switching preamble to the microcontroller. The 16-bit TMS/SWDIO command for switching to SWD mode is defined as b1110.0111.1001.1110, transmitted LSB first. This command can also be represented as 0xE79E when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that both JTAG and SWD are in their reset states.
2. Send the 16-bit JTAG-to-SWD switch command, 0xE79E, on TMS/SWDIO.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in SWD mode before sending the switch sequence, the SWD goes into the line reset state.

To verify that the Debug Access Port (DAP) has switched to the Serial Wire Debug (SWD) operating mode, perform a SWD READID operation. The ID value can be compared against the device's known ID to verify the switch.

SWD-to-JTAG Switching

To switch the operating mode of the Debug Access Port (DAP) from SWD to JTAG mode, the external debug hardware must send a switch command to the microcontroller. The 16-bit TMS/SWDIO command for switching to JTAG mode is defined as b1110.0111.0011.1100, transmitted LSB first. This command can also be represented as 0xE73C when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

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

To verify that the Debug Access Port (DAP) has switched to the JTAG operating mode, set the JTAG Instruction Register (IR) to the IDCODE instruction and shift out the Data Register (DR). The DR value can be compared against the device's known IDCODE to verify the switch.

4.4 Initialization and Configuration

After a Power-On-Reset or an external reset ($\overline{\text{RST}}$), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. This is done by enabling the five JTAG pins (PB7 and 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 five JTAG pins (PB7 andPC[3:0]) should be reverted to their default settings.

4.5 Register Descriptions

There are no APB-accessible registers in the JTAG TAP Controller or Shift Register chains. The registers within the JTAG controller are all accessed serially through the TAP Controller. The registers can be broken down into two main categories: Instruction Registers and 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 Instruction Register. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the Instruction

Register bits is shown in Table 4-4 on page 167. 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
0000	EXTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction onto the pads.
0001	INTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction into the controller.
0010	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.
1000	ABORT	Shifts data into the ARM Debug Port Abort Register.
1010	DPACC	Shifts data into and out of the ARM DP Access Register.
1011	APACC	Shifts data into and out of the ARM AC Access Register.
1110	IDCODE	Loads manufacturing information defined by the *IEEE Standard 1149.1* into the IDCODE chain and shifts it out.
1111	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. 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. This allows tests to be developed that drive known values out of the controller, which 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. 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. This allows tests to be developed that drive known values into the controller, which can be used for testing. It is important to note that although the $\overline{\text{RST}}$ input pin is on the Boundary Scan Data Register chain, it is only observable. While the INTEXT 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 of 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. Please see “Boundary Scan Data Register” on page 169 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. Please see the “ABORT Data Register” on page 170 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. Please see “DPACC Data Register” on page 170 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. Please see “APACC Data Register” on page 170 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 their input and output data streams. IDCODE is the default instruction that is loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, TRST is asserted, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 169 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. Please see “BYPASS Data Register” on page 169 for more information.

4.5.2 Data Registers

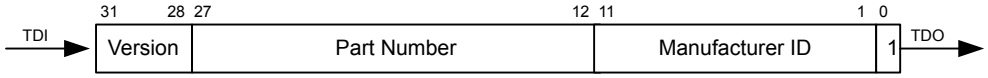
The JTAG module contains six Data Registers. These include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT serial Data Register chains. Each of these Data Registers is discussed in the following sections.

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 on page 169. The standard requires that every JTAG-compliant device 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 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 0x3BA0.0477. This allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

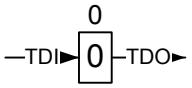
Figure 4-3. IDCODE Register Format



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 on page 169. The standard requires that every JTAG-compliant device 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 allows auto configuration test tools to determine which instruction is the default instruction.

Figure 4-4. BYPASS Register Format



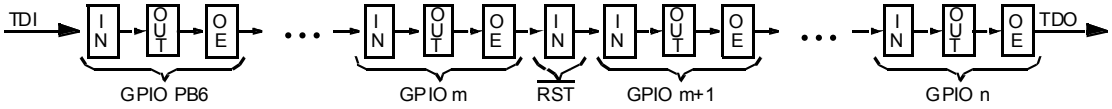
4.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 4-5 on page 170. 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 can be seen 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. These instructions either force data out of the controller, with the EXTEST instruction, or into the controller, with the INTEST instruction.

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 determines the overall operation of the device. It provides information about the device, controls the clocking to the core and individual peripherals, and handles reset detection and reporting.

5.1 Signal Description

Table 5-1 on page 171 and Table 5-2 on page 171 list the external signals of the System Control module and describe the function of each. The NMI signal is the alternate function for and functions as a GPIO after reset. under commit protection and require a special process to be configured as any alternate function or to subsequently return to the GPIO function, see “Commit Control” on page 294. The column in the table below titled "Pin Assignment" lists the GPIO pin placement for the NMI signal. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the NMI function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286. The remaining signals (with the word "fixed" in the Pin Assignment column) have a fixed pin assignment and function.

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

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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.
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.
RST	64	I	TTL	System reset input.

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

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

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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.
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.
RST	H11	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 172)

- Local control, such as reset (see “Reset Control” on page 172), power (see “Power Control” on page 176) and clock control (see “Clock Control” on page 177)
- 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 size, and other features. See the **DID0**, **DID1**, and **DC0**-**DC4** 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 controller has five sources of reset:

1. External reset input pin ($\overline{\text{RST}}$) assertion; see “External $\overline{\text{RST}}$ Pin” on page 173.
2. Power-on reset (POR); see “Power-On Reset (POR)” on page 173.
3. Internal brown-out (BOR) detector; see “Brown-Out Reset (BOR)” on page 174.
4. Software-initiated reset (with the software reset registers); see “Software Reset” on page 175.
5. A watchdog timer reset condition violation; see “Watchdog Timer Reset” on page 175.

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?
Power-On Reset	Yes	Yes	Yes
$\overline{\text{RST}}$	Yes	Pin Config Only	Yes
Brown-Out Reset	Yes	No	Yes
Software System Request Reset[a]	Yes	No	Yes
Software Peripheral Reset	No	No	Yes[b]
Watchdog Reset	Yes	No	Yes

a. By using the SYSRESREQ bit in the ARM Cortex-M3 **Application Interrupt and Reset Control (APINT)** register

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 or an external reset is the cause, and then all the other bits in the **RESC** register are cleared except for the POR or EXT indicator.

5.2.2.3 Power-On Reset (POR)

Note: The power-on reset also resets the JTAG controller. An external reset does not.

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. The 3.3-V power supply to the microcontroller must reach 3.0 V within 10 msec of V_{DD} crossing 2.0 V to guarantee proper operation. For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the $\overline{RST}$ input may be used as discussed in "External $\overline{RST}$ Pin" on page 173.

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.

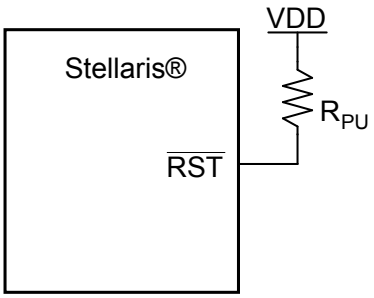
The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in Figure 21-6 on page 697.

5.2.2.4 External $\overline{RST}$ Pin

Note: It is recommended that the trace for the $\overline{RST}$ signal must be kept as short as possible. Be sure to place any components connected to the $\overline{RST}$ signal as close to the microcontroller as possible.

If the application only uses the internal POR circuit, the $\overline{RST}$ 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 173.

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



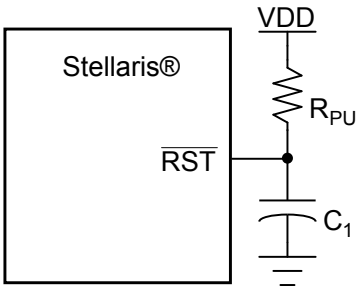
R_{PU} = 0 to 100 kΩ

The external reset pin ($\overline{RST}$) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see "JTAG Interface" on page 158). The external reset sequence is as follows:

1. The external reset pin ($\overline{RST}$) is asserted for the duration specified by T_{MIN} and then de-asserted (see "Reset" on page 696).
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 RST input may be connected to an RC network as shown in Figure 5-2 on page 174.

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

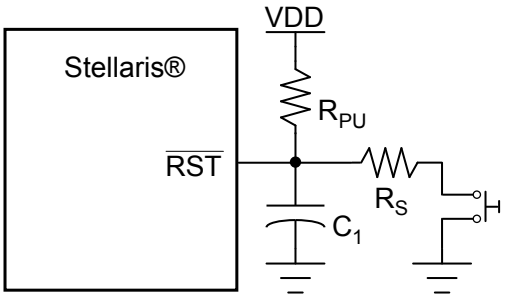


R_{PU} = 1 kΩ to 100 kΩ

C_1 = 1 nF to 10 µF

If the application requires the use of an external reset switch, Figure 5-3 on page 174 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 21-5 on page 696.

5.2.2.5 Brown-Out Reset (BOR)

A drop in the input voltage resulting in the assertion of the internal brown-out detector can be used to reset the controller. This is initially disabled and may be enabled by software.

The system 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 a controller interrupt or a system reset.

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.

The brown-out reset is equivalent to an assertion of the external 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 21-7 on page 697.

5.2.2.6 Software Reset

Software can reset a specific peripheral or generate a reset to the entire system .

Peripherals can be individually reset by software via three registers that control reset signals to each peripheral (see the **SRCRn** registers). 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). Note that all reset signals for all clocks of the specified unit are asserted as a result of a software-initiated reset.

The entire system can be reset by software by setting the SYSRESETREQ bit in the Cortex-M3 Application Interrupt and Reset Control register resets the entire system including the core. The software-initiated system reset sequence is as follows:

1. A software system reset is initiated by writing the SYSRESETREQ bit in the ARM Cortex-M3 Application Interrupt and Reset Control register.
2. An internal reset is asserted.
3. The internal reset is deasserted and the controller 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 21-8 on page 697.

5.2.2.7 Watchdog Timer Reset

The watchdog timer module's function is to prevent system hangs. 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.

After the first time-out event, the 32-bit counter is reloaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. 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, the watchdog timer asserts its reset signal to the system. 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 controller loads from memory the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

The watchdog reset timing is shown in Figure 21-9 on page 697.

5.2.3 Power Control

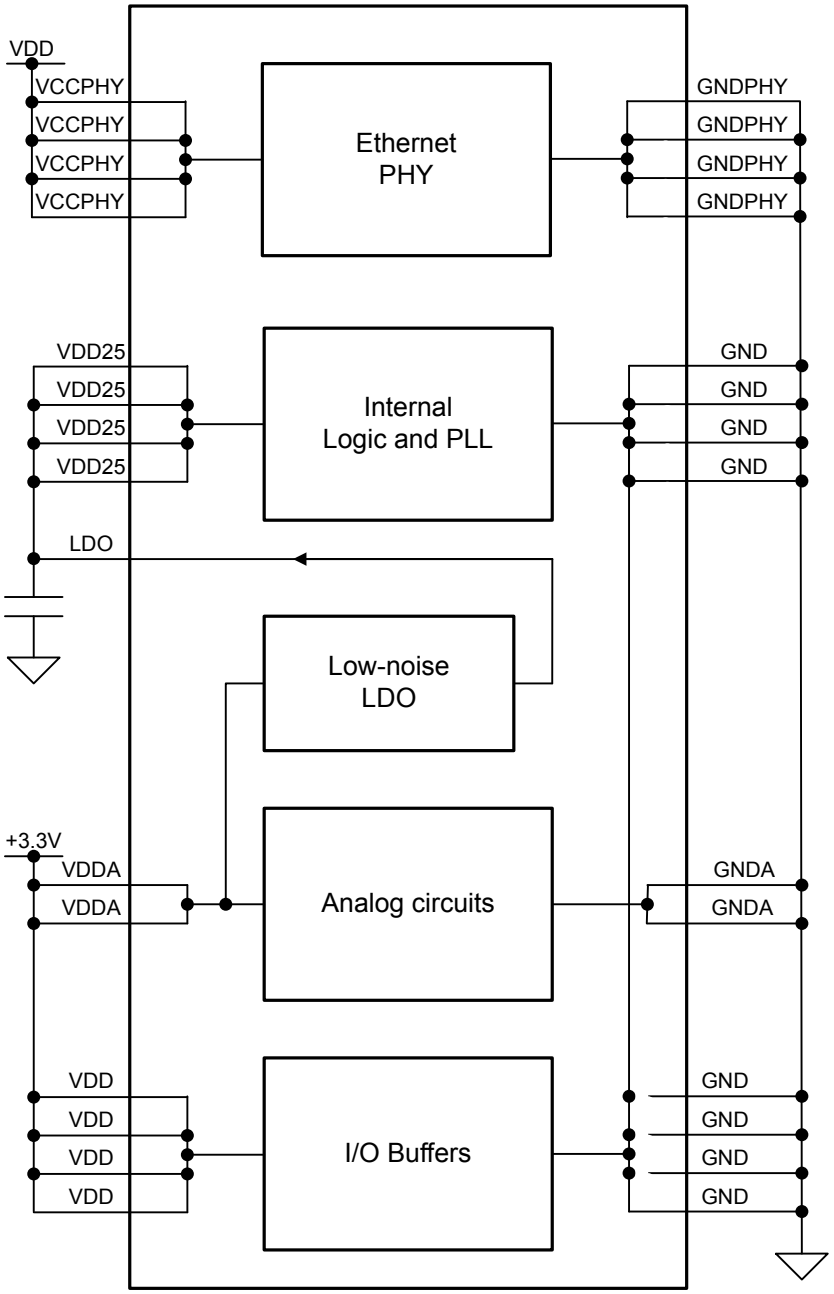
The Stellaris® microcontroller provides an integrated LDO regulator that is used to provide power to the majority of the controller's internal logic. For power reduction, the LDO regulator provides software a mechanism to adjust the regulated value, in small increments (VSTEP), over the range of 2.25 V to 2.75 V (inclusive)—or 2.5 V ± 10%. The adjustment is made by changing the value of the VADJ field in the **LDO Power Control (LDOPCTL)** register.

Figure 5-4 on page 177 shows the power architecture.

Note: On the printed circuit board, use the LDO output as the source of VDD25 input. Do not use an external regulator to supply the voltage to VDD25. In addition, the LDO requires decoupling capacitors. See “On-Chip Low Drop-Out (LDO) Regulator Characteristics” on page 690.

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 LDO and the clock circuitry.

Figure 5-4. Power Architecture



5.2.4 Clock Control

System control determines the control of clocks in this part.

5.2.4.1 Fundamental Clock Sources

There are multiple clock sources for use in the device:

- **Internal Oscillator (IOSC).** The internal oscillator is an on-chip clock source. It does not require the use of any external components. The frequency of the internal oscillator is 12 MHz ± 30%.

Applications that do not depend on accurate clock sources may use this clock source to reduce system cost. The internal oscillator is the clock source the device uses during and following POR. 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.

- **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 through 8.192 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 8.192 MHz. The single-ended clock source range is from DC through the specified speed of the device. The supported crystals are listed in the XTAL bit field in the **RCC** register (see page 194).

- **Internal 30-kHz Oscillator.** The internal 30-kHz oscillator is similar to the internal oscillator, except that it 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 main oscillator to be powered down.

- **External Real-Time Oscillator.** The external real-time oscillator provides a low-frequency, accurate clock reference. It is intended to provide the system with a real-time clock source. The real-time oscillator is part of the Hibernation Module (see “Hibernation Module” on page 238) 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 internal oscillator divided by four (3 MHz ± 30%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 8.192 MHz (inclusive). Table 5-4 on page 178 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?	
Internal Oscillator (12 MHz)	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC = 0x1
Internal Oscillator divide by 4 (3 MHz)	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC = 0x2
Main Oscillator	Yes	BYPASS = 0, OSCSRC = 0x0	Yes	BYPASS = 1, OSCSRC = 0x0
Internal 30-kHz Oscillator	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC = 0x3
External Real-Time Oscillator	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC2 = 0x7

5.2.4.2 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:

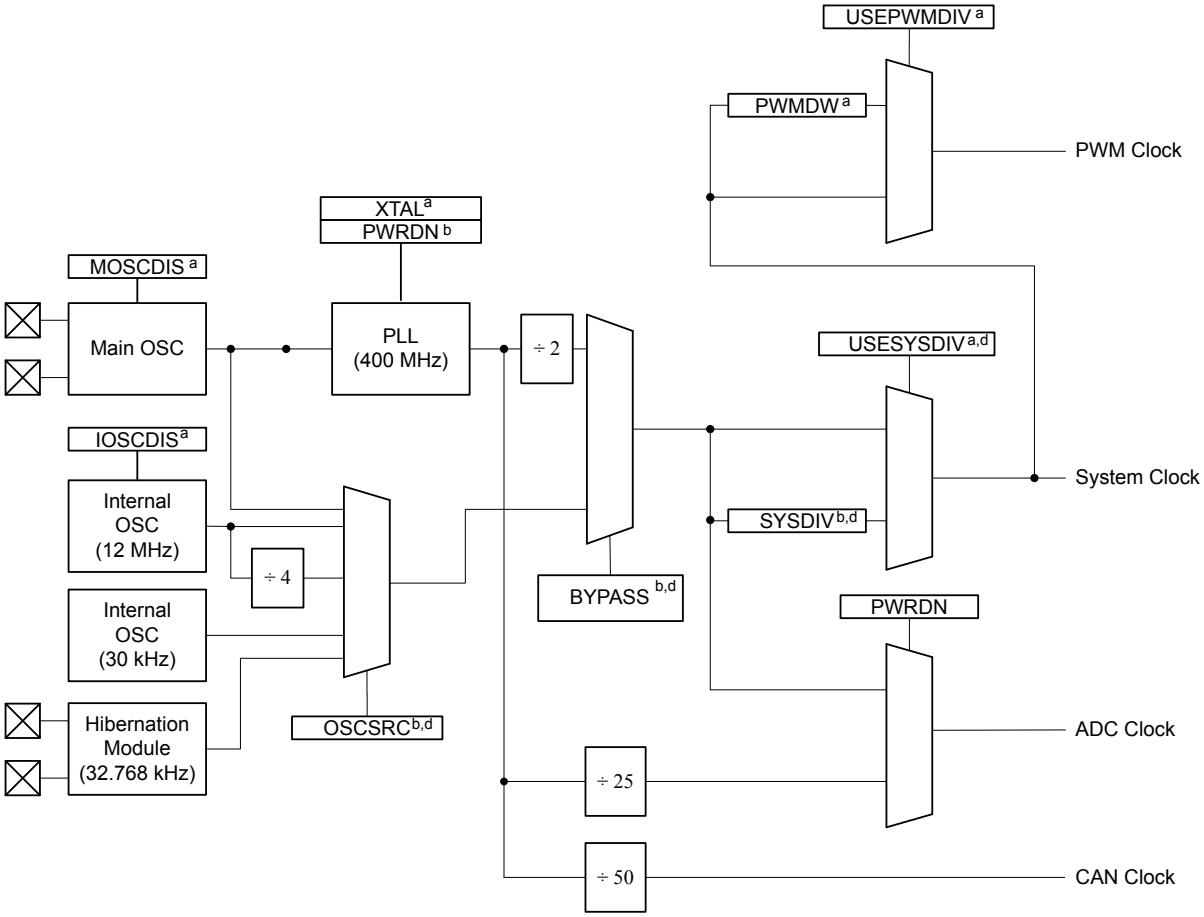
- Source of clocks in sleep and deep-sleep modes
- System clock derived from PLL or other clock source
- Enabling/disabling of oscillators and PLL

- Clock divisors
- Crystal input selection

Figure 5-5 on page 179 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled. The ADC clock signal is automatically divided down to 16 MHz 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.

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

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 178.

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	reserved	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 178.

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	reserved	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

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]
0x05	/6	33.33 MHz	Clock source frequency/6	SYSCTL_SYSDIV_6
0x06	/7	28.57 MHz	Clock source frequency/7	SYSCTL_SYSDIV_7
0x07	/8	25 MHz	Clock source frequency/8	SYSCTL_SYSDIV_8
0x08	/9	22.22 MHz	Clock source frequency/9	SYSCTL_SYSDIV_9
0x09	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
...	...	...	...	...
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.

5.2.4.3 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 8.192 MHz, otherwise, the range of supported crystals is 1 to 8.192 MHz.

The XTAL bit in the **RCC** register (see page 194) 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.

5.2.4.4 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.

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 198). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency. Table 21-10 on page 693 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 194) 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.

To configure the external 32-kHz real-time oscillator as the PLL input reference, program the OSCRC2 field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x7.

5.2.4.5 PLL Modes

The PLL has two modes of operation: Normal and Power-Down

- 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 194 and page 199).

5.2.4.6 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 21-9 on page 693). During the relock time, the affected PLL is not usable as a clock reference.

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 is defined to measure the T_{READY} requirement. The counter is clocked by the main oscillator. The range of the main oscillator has been taken into account and the down counter is set to 0x1200 (that is, ~600 µs at an 8.192 MHz external oscillator clock). 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 controller 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 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 controller is in Run, Sleep, and Deep-Sleep mode, respectively.

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

- **Run Mode.** In Run mode, the controller 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.
- **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 will bring the processor back into Run mode. See “Power Management” on page 89 for more details.

  Peripherals are clocked that are enabled in the **SCGCn** register when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** register when the auto-clock gating is disabled. The system clock has the same source and frequency as that during Run mode.
- **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 device 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 writing the Deep Sleep Enable bit in the ARM Cortex-M3 NVIC system control register and then executing

a WFI instruction. Any properly configured interrupt event in the system will bring the processor back into Run mode. See “Power Management” on page 89 for more details.

The Cortex-M3 processor core and the memory subsystem are not clocked. Peripherals are clocked that are enabled in the **DCGCn** register when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** register when auto-clock gating is disabled. The system clock source is the main oscillator by default or the internal oscillator specified in the **DSLPCLKCFG** register if one is enabled. When the **DSLPCLKCFG** register is used, the internal oscillator is powered up, if necessary, and the main oscillator is powered down. If the PLL is running at the time of the WFI instruction, hardware will power the PLL down and override 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.

- **Hibernate Mode.** In this mode, the power supplies are turned off to the main part of the device and only the Hibernation module's circuitry is active. An external wake event or RTC event is required to bring the device 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. It can determine that it has been restarted from Hibernate mode by inspecting the Hibernation module registers.

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.

5.3 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. This configures the system to run off a “raw” clock source and allows 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 5-7 on page 184 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.

Table 5-7. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	186
0x004	DID1	RO	-	Device Identification 1	202
0x008	DC0	RO	0x00FF.007F	Device Capabilities 0	204
0x010	DC1	RO	0x0101.33FF	Device Capabilities 1	205
0x014	DC2	RO	0x070F.5017	Device Capabilities 2	207
0x018	DC3	RO	0xBFFF.7FC0	Device Capabilities 3	209
0x01C	DC4	RO	0x5100.007F	Device Capabilities 4	211
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	188
0x034	LDOPCTL	R/W	0x0000.0000	LDO Power Control	189
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	234
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	235
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	237
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	190
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	191
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	192
0x05C	RESC	R/W	-	Reset Cause	193
0x060	RCC	R/W	0x0780.3AD1	Run-Mode Clock Configuration	194
0x064	PLLCFG	RO	-	XTAL to PLL Translation	198
0x070	RCC2	R/W	0x0780.2810	Run-Mode Clock Configuration 2	199
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	213
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	219
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	228
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	215
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	222
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	230

Table 5-7. System Control Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	217
0x124	DCGC1	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 1	225
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	232
0x144	DSLPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	201

5.5 Register Descriptions

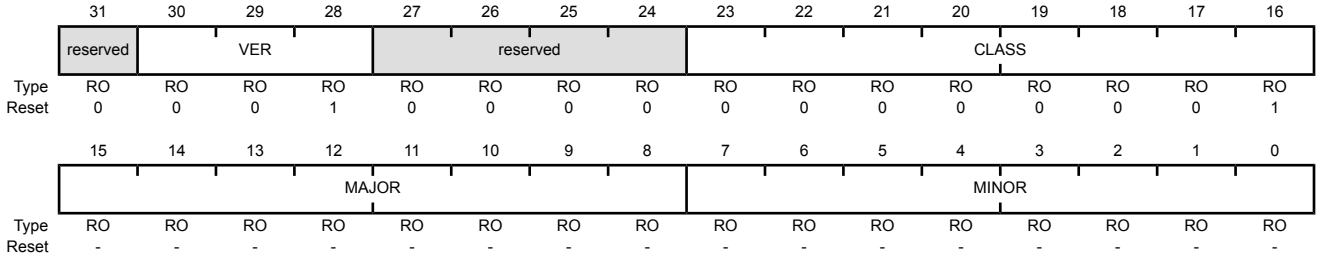
All addresses given are relative to the System Control base address of 0x400F.E000.

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 This field defines the **DID0** register format version. The version number is numeric. The value of the VER field is encoded as follows: Value Description 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	0x1	Device Class The CLASS field value identifies the internal design from which all mask sets are generated for all devices 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 devices. The value of the CLASS field is encoded as follows (all other encodings are reserved): Value Description 0x1 Stellaris® Fury-class devices.

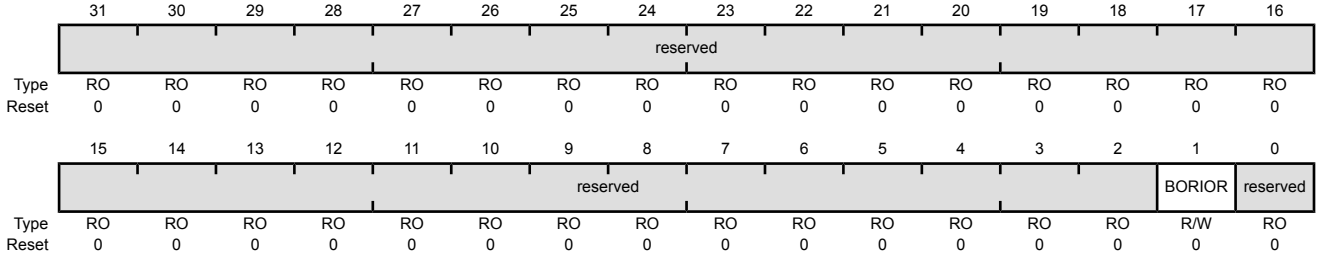
Bit/Field	Name	Type	Reset	Description
15:8	MAJOR	RO	-	Major Revision This field specifies the major revision number of the device. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, and so on). This field is encoded as follows: Value Description 0x0 Revision A (initial device) 0x1 Revision B (first base layer revision) 0x2 Revision C (second base layer revision) and so on.
7:0	MINOR	RO	-	Minor Revision This field specifies the minor revision number of the device. 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: Value Description 0x0 Initial device, or a major revision update. 0x1 First metal layer change. 0x2 Second metal layer change. 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 0x0000.7FFD



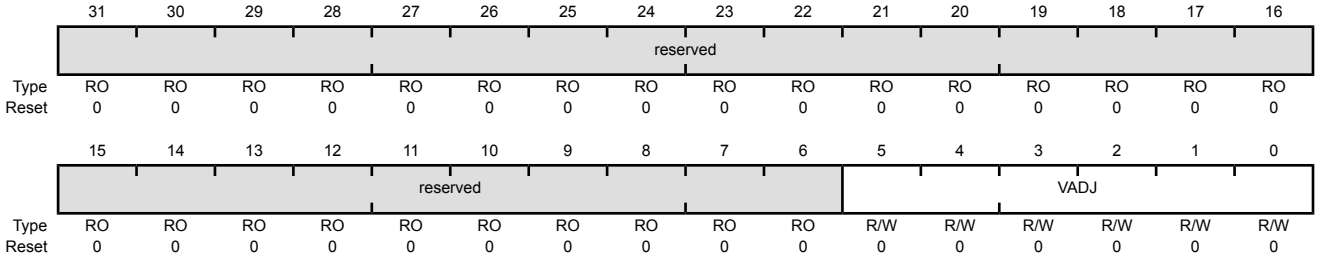
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIOR	R/W	0	BOR Interrupt or Reset This bit controls how a BOR event is signaled to the controller. If set, a reset is signaled. Otherwise, an interrupt is signaled.
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: LDO Power Control (LDOPCTL), offset 0x034

The VADJ field in this register adjusts the on-chip output voltage (V_{OUT}).

LDO Power Control (LDOPCTL)

Base 0x400F.E000
Offset 0x034
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	VADJ	R/W	0x0	LDO Output Voltage This field sets the on-chip output voltage. The programming values for the VADJ field are provided below.

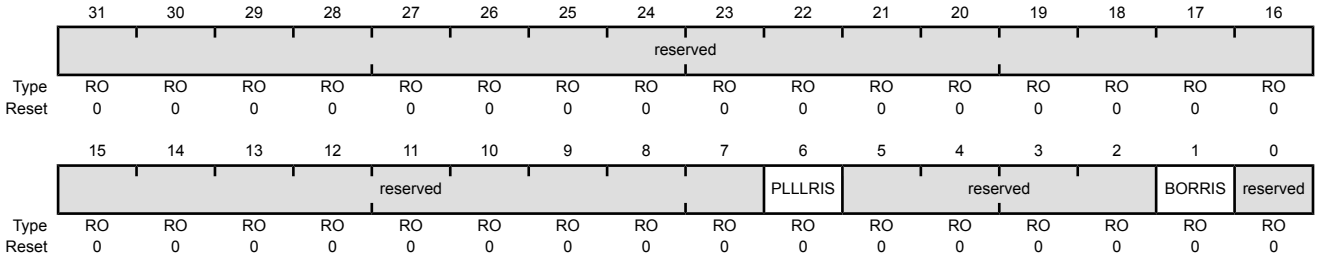
Value	V_{OUT} (V)
0x00	2.50
0x01	2.45
0x02	2.40
0x03	2.35
0x04	2.30
0x05	2.25
0x06-0x3F	Reserved
0x1B	2.75
0x1C	2.70
0x1D	2.65
0x1E	2.60
0x1F	2.55

Register 4: Raw Interrupt Status (RIS), offset 0x050

Central location for system control raw interrupts. These are set and cleared by hardware.

Raw Interrupt Status (RIS)

Base 0x400F.E000
Offset 0x050
Type RO, reset 0x0000.0000



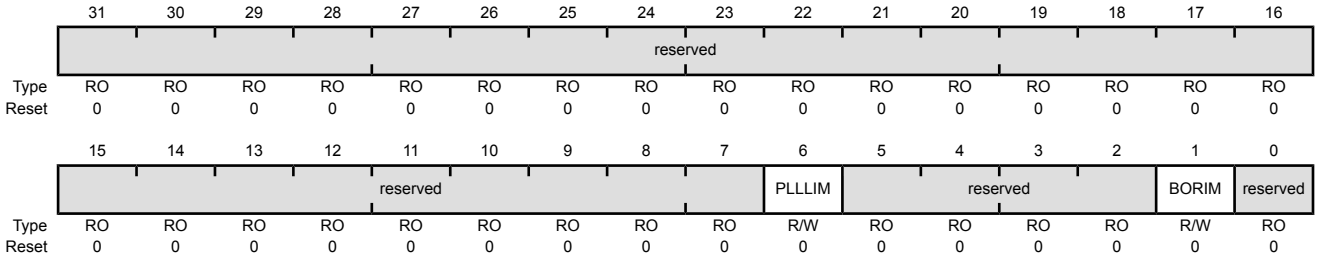
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status This bit is set when the PLL T_{READY} Timer asserts.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORRIS	RO	0	Brown-Out Reset Raw Interrupt Status This bit is the raw interrupt status for any brown-out conditions. If set, a brown-out condition is currently active. This is an unregistered signal from the brown-out detection circuit. An interrupt is reported if the `BORIM` bit in the **IMC** register is set and the `BORIOR` bit in the **PBORCTL** register is cleared.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 5: Interrupt Mask Control (IMC), offset 0x054

Central location for system control interrupt masks.

Interrupt Mask Control (IMC)

Base 0x400F.E000
Offset 0x054
Type R/W, reset 0x0000.0000



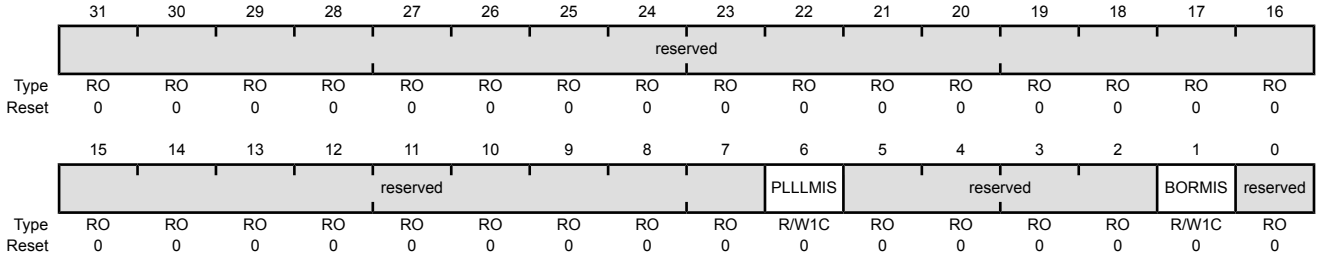
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask This bit specifies whether a PLL Lock interrupt is promoted to a controller interrupt. If set, an interrupt is generated if PLLLRIS in **RIS** is set; otherwise, an interrupt is not generated.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIM	R/W	0	Brown-Out Reset Interrupt Mask This bit specifies whether a brown-out condition is promoted to a controller interrupt. If set, an interrupt is generated if BORRIS is set; otherwise, an interrupt is not generated.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058

On a read, this register gives the current masked status value of the corresponding interrupt. All of the bits are R/W1C and this action also 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



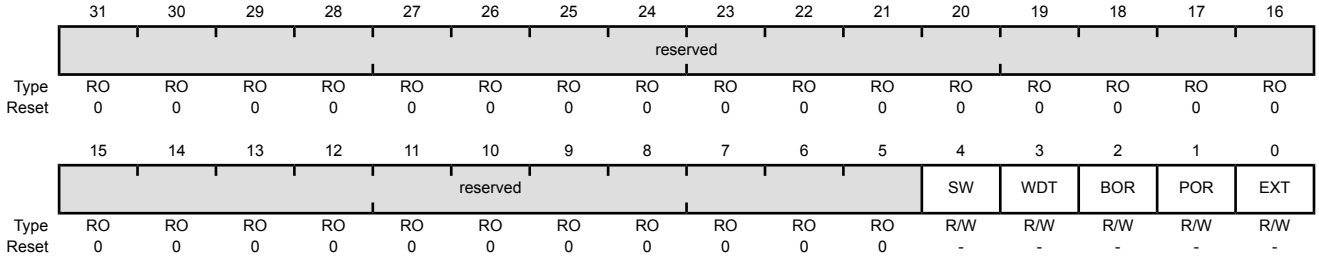
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLMIS	R/W1C	0	PLL Lock Masked Interrupt Status This bit is set when the PLL T_{READY} timer asserts. The interrupt is cleared by writing a 1 to this bit.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORMIS	R/W1C	0	BOR Masked Interrupt Status The `BORMIS` is simply the `BORRIS` ANDed with the mask value, `BORIM`.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: Reset Cause (RESC), offset 0x05C

This register is set with the reset cause after reset. The bits in this register are sticky and maintain their state across multiple reset sequences, except when a power- on reset or an external reset is the cause, in which case, all bits other than POR or EXT in the **RESC** register are cleared.

Reset Cause (RESC)

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



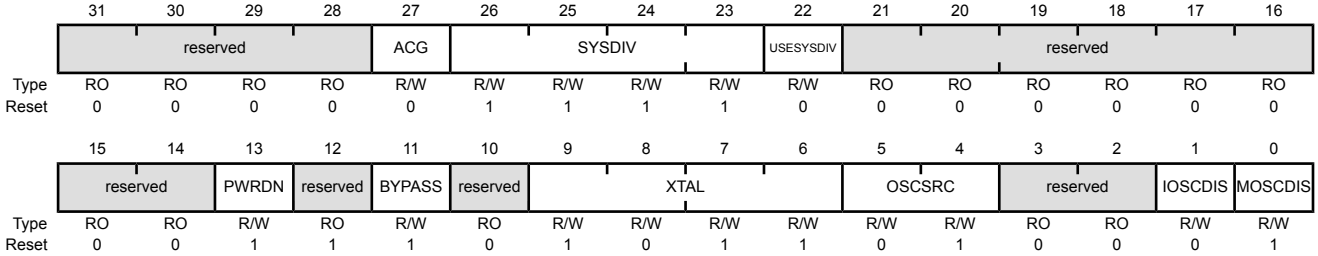
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SW	R/W	-	Software Reset When set, indicates a software reset is the cause of the reset event.
3	WDT	R/W	-	Watchdog Timer Reset When set, indicates a watchdog reset is the cause of the reset event.
2	BOR	R/W	-	Brown-Out Reset When set, indicates a brown-out reset is the cause of the reset event.
1	POR	R/W	-	Power-On Reset When set, indicates a power-on reset is the cause of the reset event.
0	EXT	R/W	-	External Reset When set, indicates an external reset ($\overline{\text{RST}}$ assertion) is the cause of the reset event.

Register 8: Run-Mode Clock Configuration (RCC), offset 0x060

This register is defined to provide source control and frequency speed.

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 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 controller enters a Sleep or Deep-Sleep mode (respectively). If set, the **SCGCn** or **DCGCn** registers are used to control the clocks distributed to the peripherals when the controller is in a sleep mode. Otherwise, the **Run-Mode Clock Gating Control (RCGCn)** registers are used when the controller enters a sleep mode. The **RCGCn** registers are always used to control the clocks in Run mode. This allows peripherals to consume less power when the controller is in a sleep mode and the peripheral is unused.
26:23	SYSDIV	R/W	0xF	System Clock Divisor 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 180 for bit encodings. If the SYSDIV value is less than MINSYSDIV (see page 205), and the PLL is being used, then the MINSYSDIV value is used as the divisor. If the PLL is not being used, the SYSDIV value can be less than MINSYSDIV.
22	USESYSDIV	R/W	0	Enable System Clock Divider Use the system clock divider as the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source. 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.

Bit/Field	Name	Type	Reset	Description
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 This bit connects to the PLL PWRDN input. The reset value of 1 powers down the PLL.
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 Chooses whether the system clock is derived from the PLL output or the OSC source. If set, the clock that drives the system is the OSC source. Otherwise, the clock that drives the system is the PLL output clock divided by the system divider. See Table 5-5 on page 180 for programming guidelines. **Note:** The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly. While the ADC works in a 14-18 MHz range, to maintain a 1 M sample/second rate, the ADC must be provided a 16-MHz clock source.
10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
9:6	XTAL	R/W	0xB	Crystal Value 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 21-10 on page 693 for more information). Value / Crystal Frequency (MHz) Not Using the PLL / Crystal Frequency (MHz) Using the PLL 0x0 — 1.000 — reserved 0x1 — 1.8432 — reserved 0x2 — 2.000 — reserved 0x3 — 2.4576 — reserved 0x4 — 3.579545 MHz 0x5 — 3.6864 MHz 0x6 — 4 MHz 0x7 — 4.096 MHz 0x8 — 4.9152 MHz 0x9 — 5 MHz 0xA — 5.12 MHz 0xB — 6 MHz (reset value) 0xC — 6.144 MHz 0xD — 7.3728 MHz 0xE — 8 MHz 0xF — 8.192 MHz
5:4	OSCSRC	R/W	0x1	Oscillator Source Selects the input source for the OSC. The values are: Value / Input Source 0x0 — MOSC — Main oscillator 0x1 — IOSC — Internal oscillator (default) 0x2 — IOSC/4 — Internal oscillator / 4 0x3 — 30 kHz — 30-KHz internal oscillator 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	Internal Oscillator Disable 0: Internal oscillator (IOSC) is enabled. 1: Internal oscillator is disabled.

Bit/Field	Name	Type	Reset	Description
0	MOSCDIS	R/W	1	Main Oscillator Disable 0: Main oscillator is enabled . 1: Main oscillator is disabled (default).

Register 9: XTAL to PLL Translation (PLLCFG), offset 0x064

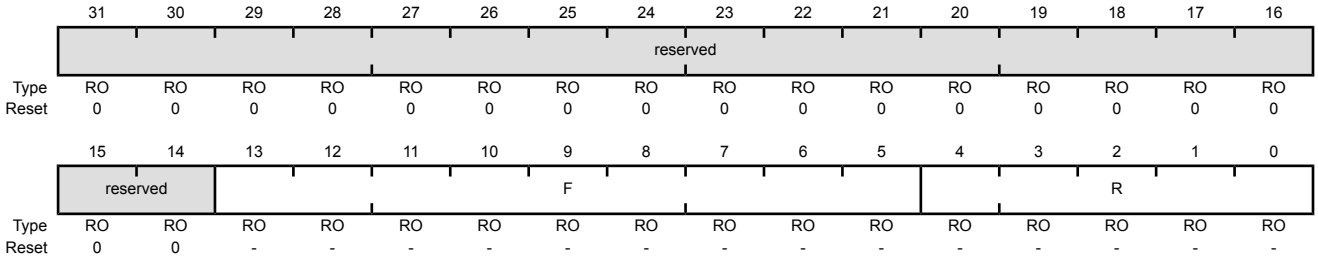
This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 194).

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	0x0	Software should not rely on the value of 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 This field specifies the value supplied to the PLL's F input.
4:0	R	RO	-	PLL R Value This field specifies the value supplied to the PLL's R input.

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

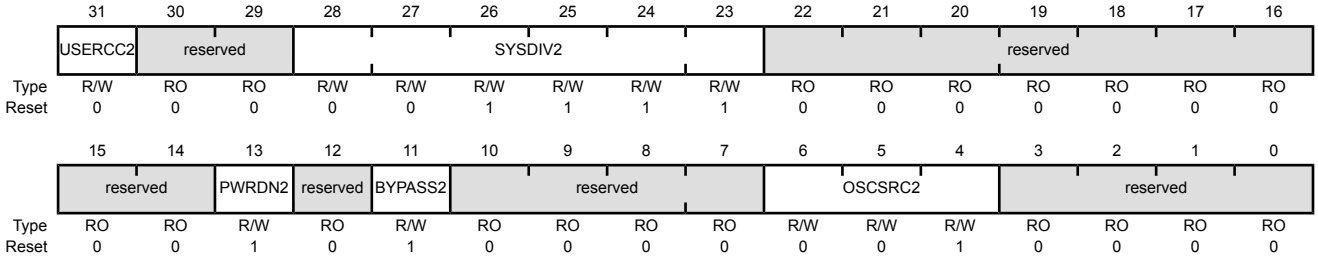
This register overrides the **RCC** equivalent register fields, as shown in Table 5-8, when the USERCC2 bit is set, allowing the extended capabilities of the **RCC2** register to be used while also providing a means to be backward-compatible to previous parts. Each **RCC2** field that supersedes an **RCC** field is located at the same LSB bit position; however, some **RCC2** fields are larger than the corresponding **RCC** field.

Table 5-8. 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 0x0780.2810



Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	Use RCC2 When set, overrides the **RCC** register fields.
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.
28:23	SYSDIV2	R/W	0x0F	System Clock Divisor 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 180 for programming guidelines.
22:14	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 When set, powers down the PLL.
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.

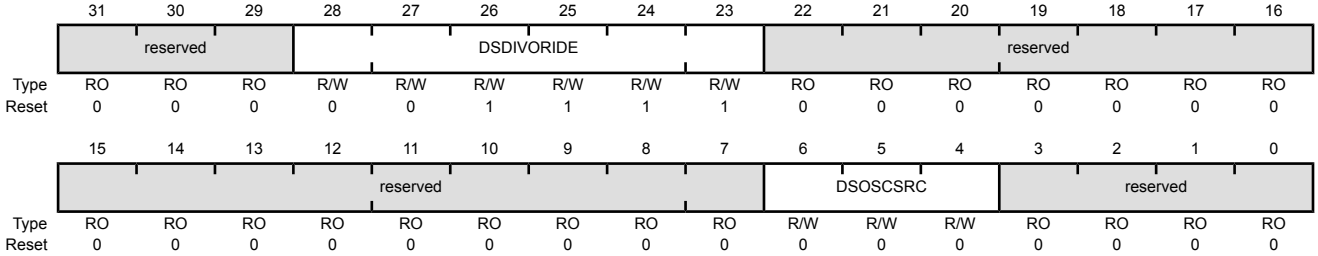
Bit/Field	Name	Type	Reset	Description
11	BYPASS2	R/W	1	Bypass PLL When set, bypasses the PLL for the clock source. See Table 5-6 on page 180 for programming guidelines.
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	OSCSRC2	R/W	0x1	Oscillator Source Selects the input source for the OSC. The values are: Value Description 0x0 MOSC Main oscillator 0x1 IOSC Internal oscillator 0x2 IOSC/4 Internal oscillator / 4 0x3 30 kHz 30-kHz internal oscillator 0x4 Reserved 0x5 Reserved 0x6 Reserved 0x7 32 kHz 32.768-kHz external oscillator
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 11: 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



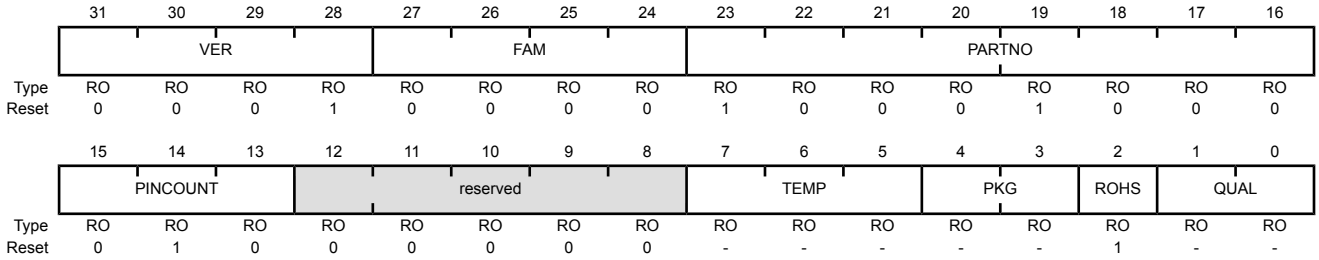
Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:23	DSDIVORIDE	R/W	0x0F	Divider Field Override 6-bit system divider field to override when Deep-Sleep occurs with PLL running.
22:7	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.
6:4	DSOSCSRC	R/W	0x0	Clock Source Specifies the clock source during Deep-Sleep mode. Value Description 0x0 MOSC Use main oscillator as source. 0x1 IOSC Use internal 12-MHz oscillator as source. 0x2 Reserved 0x3 30 kHz Use 30-kHz internal oscillator as source. 0x4 Reserved 0x5 Reserved 0x6 Reserved 0x7 32 kHz Use 32.768-kHz external oscillator as 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 12: 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 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): Value Description 0x1 Second version of the **DID1** register format.
27:24	FAM	RO	0x0	Family 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): Value Description 0x0 Stellaris family of microcontollers, that is, all devices with external part numbers starting with LM3S.
23:16	PARTNO	RO	0x88	Part Number This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved): Value Description 0x88 LM3S8938
15:13	PINCOUNT	RO	0x2	Package Pin Count This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved): Value Description 0x2 100-pin or 108-ball package

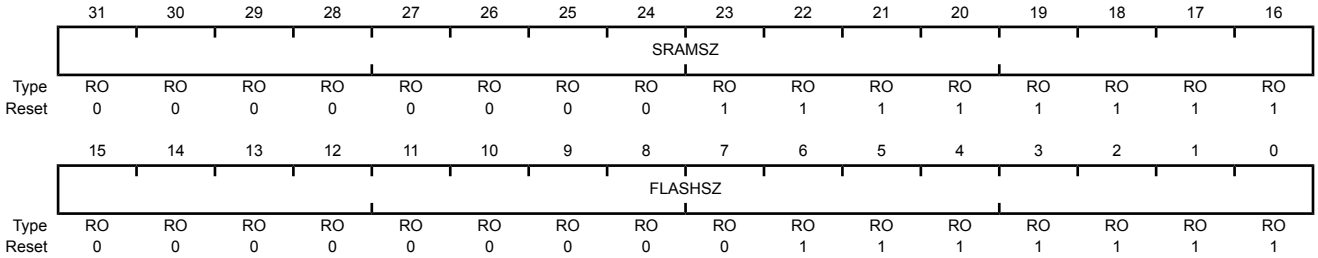
Bit/Field	Name	Type	Reset	Description
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	TEMP	RO	-	Temperature Range This field specifies the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

Register 13: 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.007F



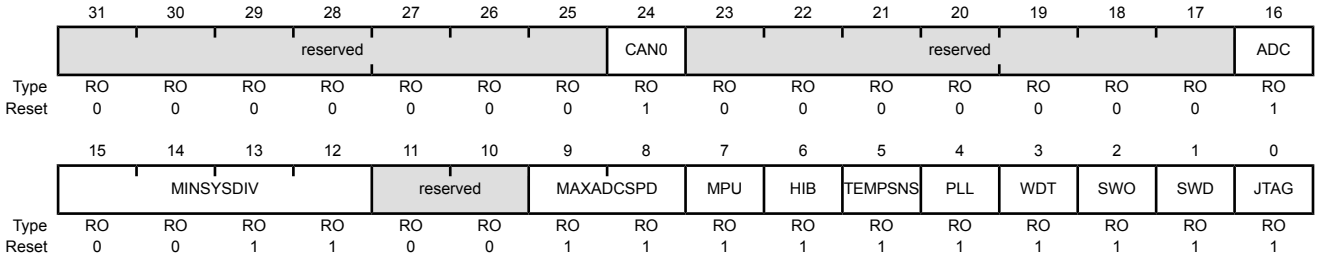
Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x00FF	SRAM Size Indicates the size of the on-chip SRAM memory. Value Description 0x00FF 64 KB of SRAM
15:0	FLASHSZ	RO	0x007F	Flash Size Indicates the size of the on-chip flash memory. Value Description 0x007F 256 KB of Flash

Register 14: Device Capabilities 1 (DC1), offset 0x010

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: CANs, PWM, ADC, Watchdog timer, Hibernation module, and debug capabilities. This register also indicates the maximum clock frequency and maximum ADC sample rate. The format of this register is consistent with the **RCGC0**, **SCGC0**, and **DCGC0** clock control registers and the **SRCR0** software reset control register.

Device Capabilities 1 (DC1)

Base 0x400F.E000
Offset 0x010
Type RO, reset 0x0101.33FF



Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of 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	CAN0	RO	1	CAN Module 0 Present When set, indicates that CAN unit 0 is present.
23:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	RO	1	ADC Module Present When set, indicates that the ADC module is present.
15:12	MINSYSDIV	RO	0x3	System Clock Divider 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. Value Description 0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4.
11:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	MAXADCSPD	RO	0x3	Max ADC Speed Indicates the maximum rate at which the ADC samples data. Value Description 0x3 1M samples/second

Bit/Field	Name	Type	Reset	Description
7	MPU	RO	1	MPU Present When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the "Cortex-M3 Peripherals" chapter in the Stellaris Data Sheet for details on the MPU.
6	HIB	RO	1	Hibernation Module Present When set, indicates that the Hibernation module is present.
5	TEMPSNS	RO	1	Temp Sensor Present When set, indicates that the on-chip temperature sensor is present.
4	PLL	RO	1	PLL Present When set, indicates that the on-chip Phase Locked Loop (PLL) is present.
3	WDT	RO	1	Watchdog Timer Present When set, indicates that a watchdog timer is present.
2	SWO	RO	1	SWO Trace Port Present When set, indicates that the Serial Wire Output (SWO) trace port is present.
1	SWD	RO	1	SWD Present When set, indicates that the Serial Wire Debugger (SWD) is present.
0	JTAG	RO	1	JTAG Present When set, indicates that the JTAG debugger interface is present.

Register 15: Device Capabilities 2 (DC2), offset 0x014

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: Analog Comparators, General-Purpose Timers, I2Cs, QEIs, SSIs, and UARTs. The format of this register is consistent with the **RCGC1**, **SCGC1**, and **DCGC1** clock control registers and the **SRCR1** software reset control register.

Device Capabilities 2 (DC2)

Base 0x400F.E000
Offset 0x014
Type RO, reset 0x070F.5017

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					COMP2	COMP1	COMP0	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	1	1	1	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							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	0	1	0	1	1	1

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	RO	1	Analog Comparator 2 Present When set, indicates that analog comparator 2 is present.
25	COMP1	RO	1	Analog Comparator 1 Present When set, indicates that analog comparator 1 is present.
24	COMP0	RO	1	Analog Comparator 0 Present When set, indicates that analog comparator 0 is present.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	RO	1	Timer 3 Present When set, indicates that General-Purpose Timer module 3 is present.
18	TIMER2	RO	1	Timer 2 Present When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	1	Timer 1 Present When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	Timer 0 Present 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.

Bit/Field	Name	Type	Reset	Description
14	I2C1	RO	1	I2C Module 1 Present 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 When set, indicates that I2C module 0 is present.
11:5	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	SSI0	RO	1	SSI0 Present 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	UART2 Present When set, indicates that UART module 2 is present.
1	UART1	RO	1	UART1 Present When set, indicates that UART module 1 is present.
0	UART0	RO	1	UART0 Present When set, indicates that UART module 0 is present.

Register 16: Device Capabilities 3 (DC3), offset 0x018

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: Analog Comparator I/Os, CCP I/Os, ADC I/Os, and PWM I/Os.

Device Capabilities 3 (DC3)

Base 0x400F.E000
Offset 0x018
Type RO, reset 0xBFFF.7FC0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	32KHZ	reserved	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
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	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	reserved					
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	1	32KHz Input Clock Available 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 When set, indicates that Capture/Compare/PWM pin 5 is present.
28	CCP4	RO	1	CCP4 Pin Present When set, indicates that Capture/Compare/PWM pin 4 is present.
27	CCP3	RO	1	CCP3 Pin Present When set, indicates that Capture/Compare/PWM pin 3 is present.
26	CCP2	RO	1	CCP2 Pin Present When set, indicates that Capture/Compare/PWM pin 2 is present.
25	CCP1	RO	1	CCP1 Pin Present When set, indicates that Capture/Compare/PWM pin 1 is present.
24	CCP0	RO	1	CCP0 Pin Present When set, indicates that Capture/Compare/PWM pin 0 is present.
23	ADC7	RO	1	ADC7 Pin Present When set, indicates that ADC pin 7 is present.
22	ADC6	RO	1	ADC6 Pin Present When set, indicates that ADC pin 6 is present.
21	ADC5	RO	1	ADC5 Pin Present When set, indicates that ADC pin 5 is present.

Bit/Field	Name	Type	Reset	Description
20	ADC4	RO	1	ADC4 Pin Present When set, indicates that ADC pin 4 is present.
19	ADC3	RO	1	ADC3 Pin Present When set, indicates that ADC pin 3 is present.
18	ADC2	RO	1	ADC2 Pin Present When set, indicates that ADC pin 2 is present.
17	ADC1	RO	1	ADC1 Pin Present When set, indicates that ADC pin 1 is present.
16	ADC0	RO	1	ADC0 Pin Present When set, indicates that ADC pin 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	C2O	RO	1	C2o Pin Present When set, indicates that the analog comparator 2 output pin is present.
13	C2PLUS	RO	1	C2+ Pin Present When set, indicates that the analog comparator 2 (+) input pin is present.
12	C2MINUS	RO	1	C2- Pin Present When set, indicates that the analog comparator 2 (-) input pin is present.
11	C1O	RO	1	C1o Pin Present When set, indicates that the analog comparator 1 output pin is present.
10	C1PLUS	RO	1	C1+ Pin Present When set, indicates that the analog comparator 1 (+) input pin is present.
9	C1MINUS	RO	1	C1- Pin Present When set, indicates that the analog comparator 1 (-) input pin is present.
8	C0O	RO	1	C0o Pin Present When set, indicates that the analog comparator 0 output pin is present.
7	C0PLUS	RO	1	C0+ Pin Present When set, indicates that the analog comparator 0 (+) input pin is present.
6	C0MINUS	RO	1	C0- Pin Present When set, indicates that the analog comparator 0 (-) input pin is present.
5:0	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 17: Device Capabilities 4 (DC4), offset 0x01C

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: Ethernet MAC and PHY, GPIOs, and CCP I/Os. The format of this register is consistent with the **RCGC2**, **SCGC2**, and **DCGC2** clock control registers and the **SRCR2** software reset control register.

Device Capabilities 4 (DC4)

Base 0x400F.E000
Offset 0x01C
Type RO, reset 0x5100.007F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPHY0	reserved	EMAC0	reserved			E1588	reserved							
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	1	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
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	1	1	1	1	1	1	1

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	EPHY0	RO	1	Ethernet PHY0 Present When set, indicates that Ethernet PHY module 0 is present.
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	EMAC0	RO	1	Ethernet MAC0 Present When set, indicates that Ethernet MAC module 0 is present.
27:25	reserved	RO	0	Software should not rely on the value of 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	E1588	RO	1	1588 Capable When set, indicates that that EMAC0 is 1588-capable.
23:7	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	GPIOG	RO	1	GPIO Port G Present When set, indicates that GPIO Port G is present.
5	GPIOF	RO	1	GPIO Port F Present When set, indicates that GPIO Port F is present.
4	GPIOE	RO	1	GPIO Port E Present When set, indicates that GPIO Port E is present.

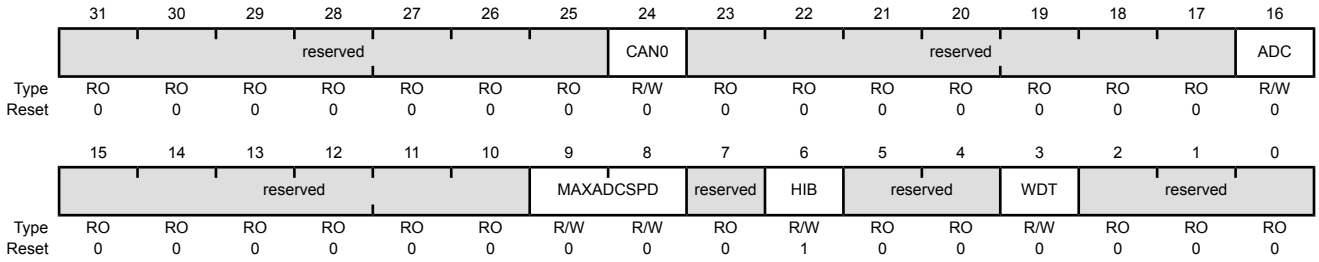
Bit/Field	Name	Type	Reset	Description
3	GPIOD	RO	1	GPIO Port D Present When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	GPIO Port C Present When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	GPIO Port B Present When set, indicates that GPIO Port B is present.
0	GPIOA	RO	1	GPIO Port A Present When set, indicates that GPIO Port A is present.

Register 18: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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



Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of 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	CAN0	R/W	0	CAN0 Clock Gating Control This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
23:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

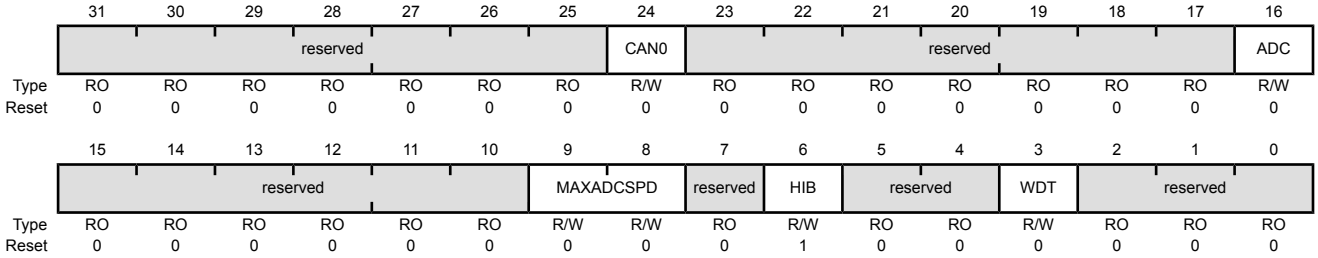
Bit/Field	Name	Type	Reset	Description
9:8	MAXADCSPD	R/W	0	ADC Sample Speed This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows: Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 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 This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
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	WDT	R/W	0	WDT Clock Gating Control This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit 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 19: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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



Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of 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	CAN0	R/W	0	CAN0 Clock Gating Control This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
23:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

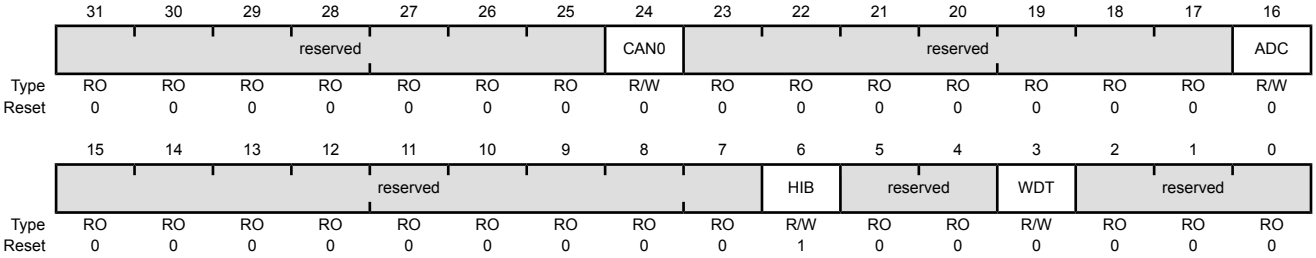
Bit/Field	Name	Type	Reset	Description
9:8	MAXADCSPD	R/W	0	ADC Sample Speed This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows: Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 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 This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
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	WDT	R/W	0	WDT Clock Gating Control This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit 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 20: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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



Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of 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	CAN0	R/W	0	CAN0 Clock Gating Control This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
23:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
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 This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.

Bit/Field	Name	Type	Reset	Description
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	WDT	R/W	0	WDT Clock Gating Control This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit 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 21: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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					COMP2	COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	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							SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	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
31:27	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	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
19	TIMER3	R/W	0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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 This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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 This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
11:5	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	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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.

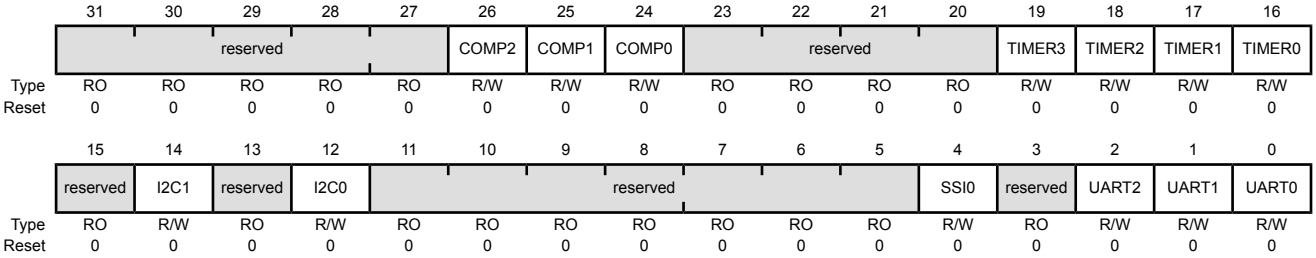
Bit/Field	Name	Type	Reset	Description
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 22: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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
31:27	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	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
19	TIMER3	R/W	0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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 This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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 This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
11:5	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	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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.

Bit/Field	Name	Type	Reset	Description
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 23: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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					COMP2	COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
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							SSI0	reserved	UART2	UART1	UART0
RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	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
31:27	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	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
19	TIMER3	R/W	0	Timer 3 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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 This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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 This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
11:5	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	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate 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.

Bit/Field	Name	Type	Reset	Description
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 24: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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	EPHY0	reserved	EMAC0	reserved											
RO	R/W	RO	R/W	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									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
RO	RO	RO	RO	RO	RO	RO	RO	RO	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	reserved	RO	0	Software should not rely on the value of 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	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
27:7	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	GPIOG	R/W	0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

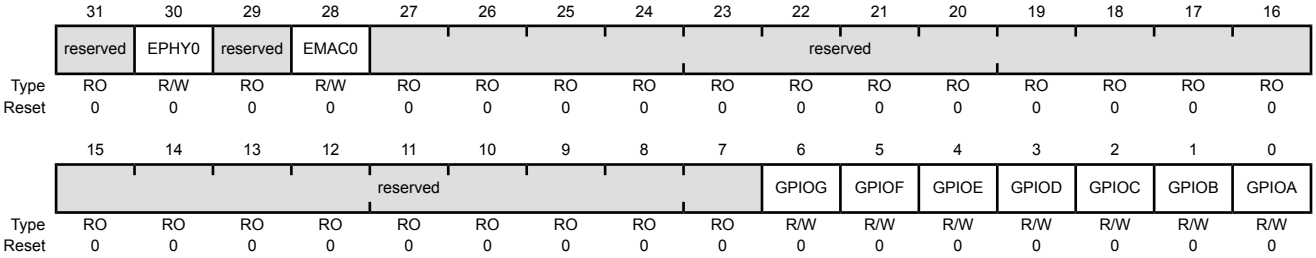
Bit/Field	Name	Type	Reset	Description
5	GPIOF	R/W	0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
4	GPIOE	R/W	0	Port E Clock Gating Control This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 25: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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	reserved	RO	0	Software should not rely on the value of 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	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
27:7	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	GPIOG	R/W	0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
5	GPIOF	R/W	0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
4	GPIOE	R/W	0	Port E Clock Gating Control This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 26: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units 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 units to control. This is 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	EPHY0	reserved	EMAC0	reserved											
Type	RO	R/W	RO	R/W	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									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	reserved	RO	0	Software should not rely on the value of 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	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
27:7	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	GPIOG	R/W	0	Port G Clock Gating Control This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

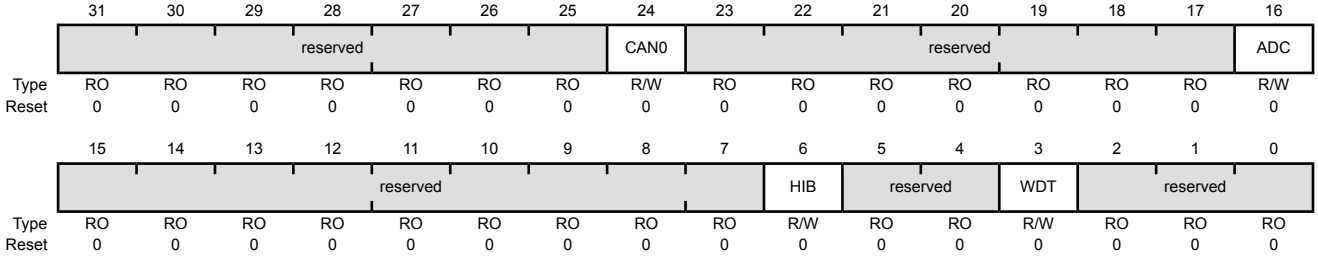
Bit/Field	Name	Type	Reset	Description
5	GPIOF	R/W	0	Port F Clock Gating Control This bit controls the clock gating for Port F. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
4	GPIOE	R/W	0	Port E Clock Gating Control This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 27: Software Reset Control 0 (SRCR0), offset 0x040

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
31:25	reserved	RO	0	Software should not rely on the value of 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	CAN0	R/W	0	CAN0 Reset Control Reset control for CAN unit 0.
23:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	R/W	0	ADC0 Reset Control Reset control for SAR ADC module 0.
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 Reset control for the Hibernation module.
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	WDT	R/W	0	WDT Reset Control Reset control for Watchdog unit.
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: Software Reset Control 1 (SRCR1), offset 0x044

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					COMP2	COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	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							SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	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
31:27	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	COMP2	R/W	0	Analog Comp 2 Reset Control Reset control for analog comparator 2.
25	COMP1	R/W	0	Analog Comp 1 Reset Control Reset control for analog comparator 1.
24	COMP0	R/W	0	Analog Comp 0 Reset Control Reset control for analog comparator 0.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Reset Control Reset control for General-Purpose Timer module 3.
18	TIMER2	R/W	0	Timer 2 Reset Control Reset control for General-Purpose Timer module 2.
17	TIMER1	R/W	0	Timer 1 Reset Control Reset control for General-Purpose Timer module 1.
16	TIMER0	R/W	0	Timer 0 Reset Control Reset control for General-Purpose Timer module 0.
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 Reset control for I2C unit 1.

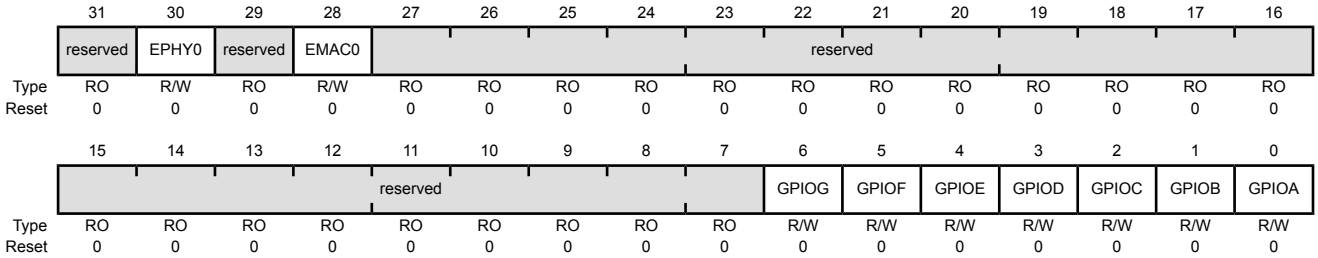
Bit/Field	Name	Type	Reset	Description
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Reset Control Reset control for I2C unit 0.
11:5	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	SSI0	R/W	0	SSI0 Reset Control Reset control for SSI unit 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	UART2	R/W	0	UART2 Reset Control Reset control for UART unit 2.
1	UART1	R/W	0	UART1 Reset Control Reset control for UART unit 1.
0	UART0	R/W	0	UART0 Reset Control Reset control for UART unit 0.

Register 29: Software Reset Control 2 (SRCR2), offset 0x048

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	reserved	RO	0	Software should not rely on the value of 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	EPHY0	R/W	0	PHY0 Reset Control Reset control for Ethernet PHY unit 0.
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	EMAC0	R/W	0	MAC0 Reset Control Reset control for Ethernet MAC unit 0.
27:7	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	GPIOG	R/W	0	Port G Reset Control Reset control for GPIO Port G.
5	GPIOF	R/W	0	Port F Reset Control Reset control for GPIO Port F.
4	GPIOE	R/W	0	Port E Reset Control Reset control for GPIO Port E.
3	GPIOD	R/W	0	Port D Reset Control Reset control for GPIO Port D.
2	GPIOC	R/W	0	Port C Reset Control Reset control for GPIO Port C.
1	GPIOB	R/W	0	Port B Reset Control Reset control for GPIO Port B.
0	GPIOA	R/W	0	Port A Reset Control Reset control for GPIO Port A.

6 Hibernation Module

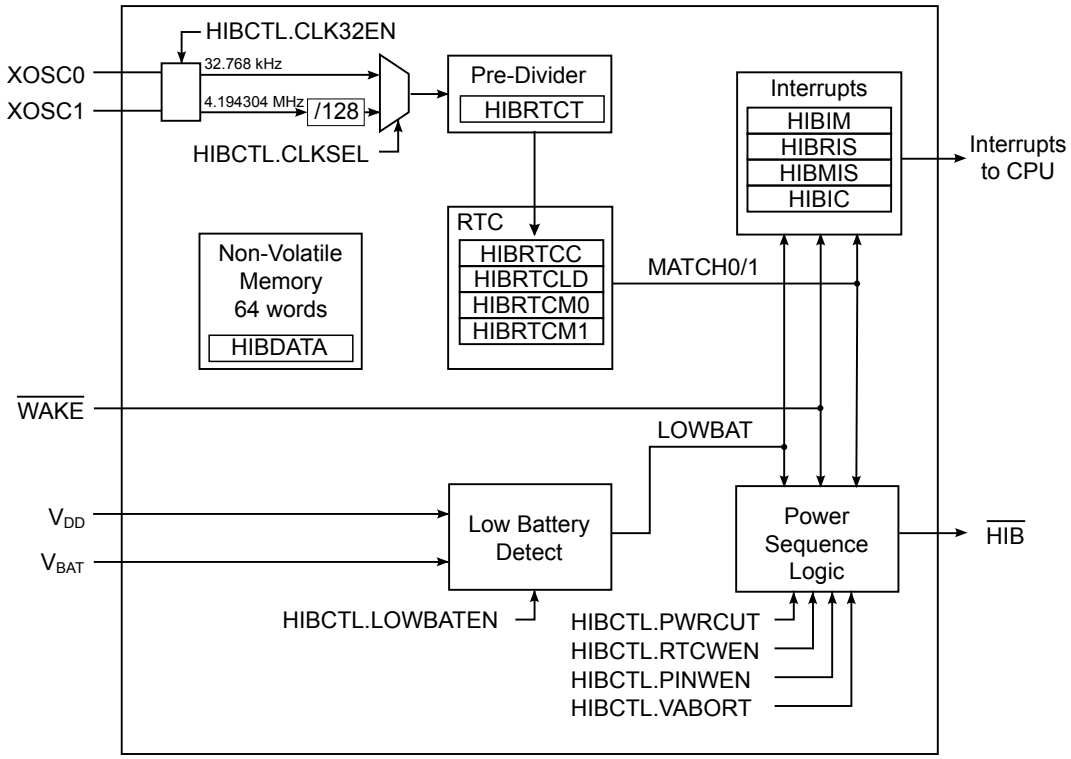
The Hibernation Module manages removal and restoration of power to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal, or at 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:

- System power control using discrete external regulator
- Dedicated pin for waking from an external signal
- Low-battery detection, signaling, and interrupt generation
- 32-bit real-time clock (RTC)
- Two 32-bit RTC match registers for timed wake-up and interrupt generation
- Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal
- RTC predivider trim for making fine adjustments to the clock rate
- 64 32-bit words of non-volatile memory
- 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

Table 6-1 on page 239 and Table 6-2 on page 240 list the external signals of the Hibernation module and describe the function of each. These signals have dedicated functions and are not alternate functions for any GPIO signals.

Table 6-1. Hibernate Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
HIB	51	O	OD	An open-drain output with internal pull-up 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 crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
XOSC1	53	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. Hibernate Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
HIB	M12	O	OD	An open-drain output with internal pull-up 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 crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
XOSC1	K12	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

The Hibernation module controls the power to the processor with an enable signal (HIB) that signals an external voltage regulator to turn off.

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}). A voting circuit indicates the larger and an internal power switch selects the appropriate voltage source. The Hibernation module also has a separate clock source to maintain a real-time clock (RTC). Once in hibernation, the module signals an external voltage regulator to turn back on the power 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 $t_{HIB_REG_WRITE}$, therefore software must guarantee that a delay of $t_{HIB_REG_WRITE}$ is inserted between back-to-back writes to certain Hibernation registers, or between a write followed by a read to those same registers. There is no restriction on timing for back-to-back reads from the Hibernation module. The following registers are subject to this timing restriction:

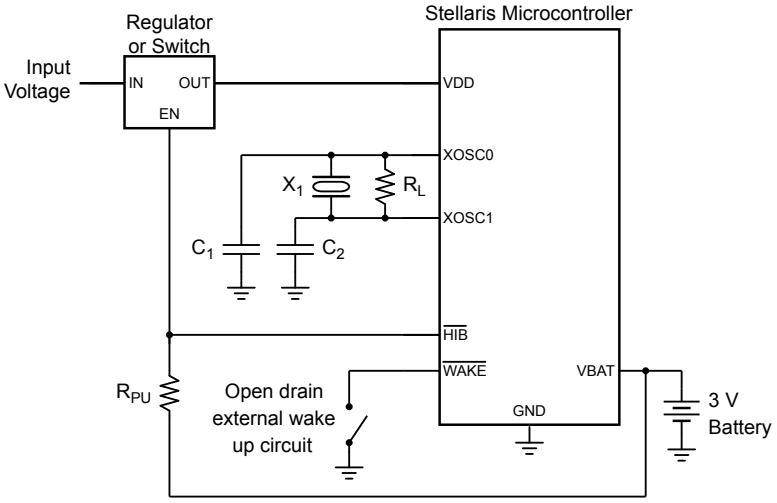
- **Hibernation RTC Counter (HIBRTCC)**
- **Hibernation RTC Match 0 (HIBRTCM0)**
- **Hibernation RTC Match 1 (HIBRTCM1)**
- **Hibernation RTC Load (HIBRTCLD)**
- **Hibernation RTC Trim (HIBRTCT)**
- **Hibernation Data (HIBDATA)**

6.3.2 Clock Source

The Hibernation module must be clocked by an external source, even if the RTC feature is not used. An external oscillator or crystal can be 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 the 32.768-kHz clock reference. For an alternate clock source, a 32.768-kHz oscillator can be connected to the XOSC0 pin. See Figure 6-2 on page 241 and Figure 6-3 on page 242. Note that these diagrams only show the connection to the Hibernation pins and not to the full system. See "Hibernation Module" on page 698 for specific values.

The clock source is enabled by setting the CLK32EN bit of the **HIBCTL** register. The type of clock source is selected by setting the CLKSEL bit to 0 for a 4.194304-MHz clock source, and to 1 for a 32.768-kHz clock source. If the bit is set to 0, the 4.194304-MHz input clock is divided by 128, resulting in a 32.768-kHz clock source. If a crystal is used for the clock source, the software must leave a delay of t_{XOSC_SETTLE} after setting 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. Clock Source Using Crystal



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

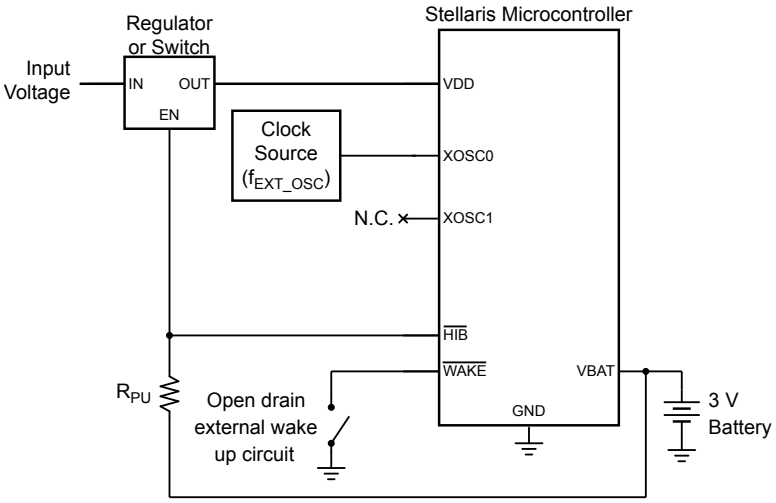
$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

R_L = Load resistor is R_{XOSC_LOAD}.

R_{PU} = Pull-up resistor (1 M½).

See "Hibernation Module" on page 698 for specific parameter values.

Figure 6-3. Clock Source Using Dedicated Oscillator



Note: R_{PU} = Pull-up resistor (1 M½).

6.3.3 Battery Management

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}. When this happens, an interrupt can be generated. The module can also be configured so that it will not go into Hibernate mode if the battery voltage drops below this threshold. Battery voltage is not measured while in Hibernate mode.

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.

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.

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 **HIBRIS** register will be set when the battery level is low. If the VABORT bit 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 244).

6.3.4 Real-Time Clock

The Hibernation module includes a 32-bit counter that increments once per second with a proper clock source and configuration (see “Clock Source” on page 241). The 32.768-kHz clock signal is fed into a predivider register which 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 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 244). As long as the RTC is enabled and a valid V_{BAT} is present, the RTC continues counting, regardless of whether V_{DD} is present or if the part is in hibernation.

6.3.5 Battery-Backed Memory

The Hibernation module contains 64 32-bit words of memory which are retained during hibernation. This memory is powered from the battery or auxiliary power supply during hibernation. The processor software can save state information in this memory prior to hibernation, and can then recover the state upon waking. The battery-backed memory can be accessed through the **HIBDATA** registers.

6.3.6 Power Control

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 its host device. 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 698 for more details.

The Hibernation module controls power to the microcontroller through the use of the $\overline{\text{HIB}}$ pin. This pin is intended to be connected to the enable signal of the external regulator(s) providing 3.3 V and/or 2.5 V to the microcontroller. When the $\overline{\text{HIB}}$ signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the system. 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 device 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 Initiating Hibernate

Hibernation mode is initiated by the microcontroller setting the HIBREQ bit of the **HIBCTL** register. Prior to doing this, a wake-up condition must be configured, either from the external WAKE pin, or by using an RTC match.

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 can be set prior to going into hibernation. The $\overline{\text{WAKE}}$ pin includes a weak internal pull-up. Note that both the HIB and $\overline{\text{WAKE}}$ pins use the Hibernation module's internal power supply as the logic 1 reference.

When the Hibernation module wakes, the microcontroller will see 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 244) and by looking for state data in the battery-backed memory (see "Battery-Backed Memory" on page 243).

When the $\overline{\text{HIB}}$ signal deasserts, enabling the external regulator, the external regulator must reach the operating voltage within $t_{HIB_TO_VDD}$.

6.3.8 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 **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 at power-on to see if a wake condition is pending, which indicates to the software that a hibernation wake occurred.

The events that can trigger an interrupt are configured by setting the appropriate bits in the **HIBIM** register. Pending interrupts can be cleared by writing the corresponding bit in the **HIBIC** register.

6.4 Initialization and Configuration

The Hibernation module can be set in 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 show bit 2 (CLKSEL) of the **HIBCTL** register set to 1. 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 system, software must allow a delay of $t_{HIB_REG_WRITE}$ after writes to certain registers (see “Register Access Timing” on page 240). The registers that require a delay are listed in a note in “Register Map” on page 245 as well as in each register description.

6.4.1 Initialization

The Hibernation module clock source must be enabled first, even if the RTC feature is not used. If a 4.194304-MHz crystal is used, 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 for a time of t_{XOSC_SETTLE} for the crystal to power up and stabilize before performing any other operations with the Hibernation module.

If a 32.678-kHz oscillator is used, then perform the following steps:

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

The above is only necessary when the entire system is initialized for the first time. If the processor is powered due to a wake from 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.

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.

6.4.5 RTC/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.

6.5 Register Map

Table 6-3 on page 246 lists the Hibernation registers. All addresses given are relative to the Hibernation Module base address at 0x400F.C000. Note that the Hibernation module clock must be enabled before the registers can be programmed (see page 213). There must be a delay of 3 system clocks after the Hibernation module clock is enabled before any Hibernation module registers are accessed.

Important: The Hibernation module registers are reset under two conditions:

1. A system reset when the RTCEN and the PINWEN bits in the **HIBCTL** register are both cleared.

2. A cold POR, when both the V_{DD} and V_{BAT} supplies are removed.

Any other reset condition is ignored by the Hibernation module.

Table 6-3. Hibernation Module Register Map

Offset	Name	Type	Reset	Description	See page
0x000	HIBRTCC	RO	0x0000.0000	Hibernation RTC Counter	247
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	Hibernation RTC Match 0	248
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	Hibernation RTC Match 1	249
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	Hibernation RTC Load	250
0x010	HIBCTL	R/W	0x8000.0000	Hibernation Control	251
0x014	HIBIM	R/W	0x0000.0000	Hibernation Interrupt Mask	253
0x018	HIBRIS	RO	0x0000.0000	Hibernation Raw Interrupt Status	254
0x01C	HIBMIS	RO	0x0000.0000	Hibernation Masked Interrupt Status	255
0x020	HIBIC	R/W1C	0x0000.0000	Hibernation Interrupt Clear	256
0x024	HIBRTCT	R/W	0x0000.7FFF	Hibernation RTC Trim	257
0x030-0x12C	HIBDATA	R/W	-	Hibernation Data	258

6.6 Register Descriptions

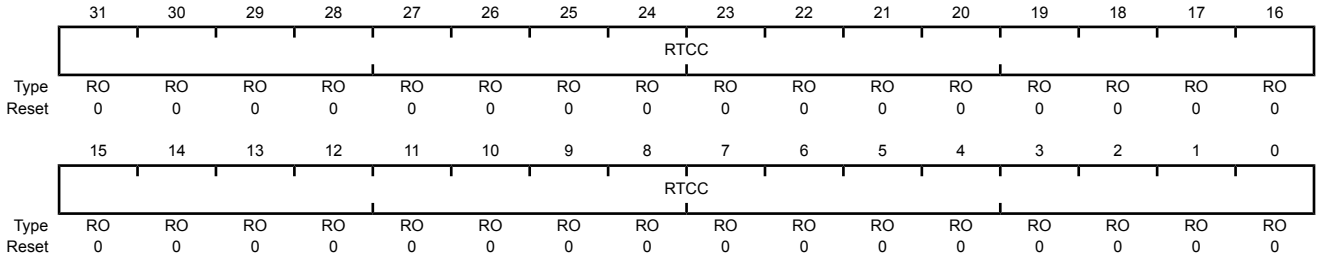
The remainder of this section lists and describes the Hibernation module registers, in numerical order by address offset.

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

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

Hibernation RTC Counter (HIBRTCC)

Base 0x400F.C000
Offset 0x000
Type RO, reset 0x0000.0000



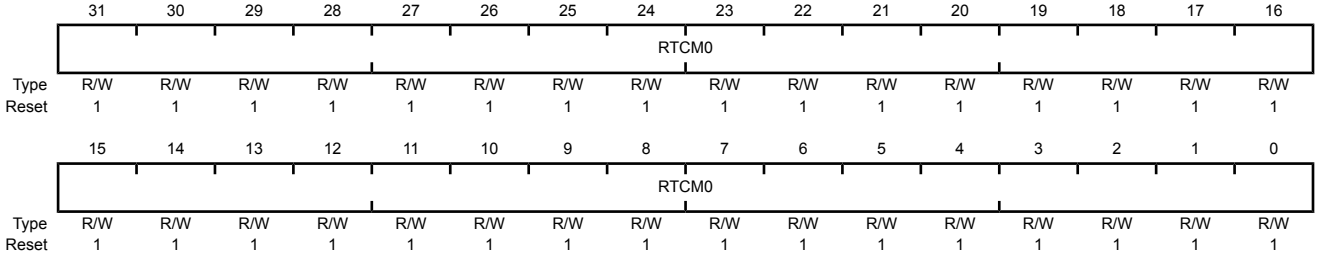
Bit/Field	Name	Type	Reset	Description
31:0	RTCC	RO	0x0000.0000	RTC Counter A read returns the 32-bit counter value. This register is read-only. To change the value, use the **HIBRTCLD** register.

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

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

Hibernation RTC Match 0 (HIBRTCM0)

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



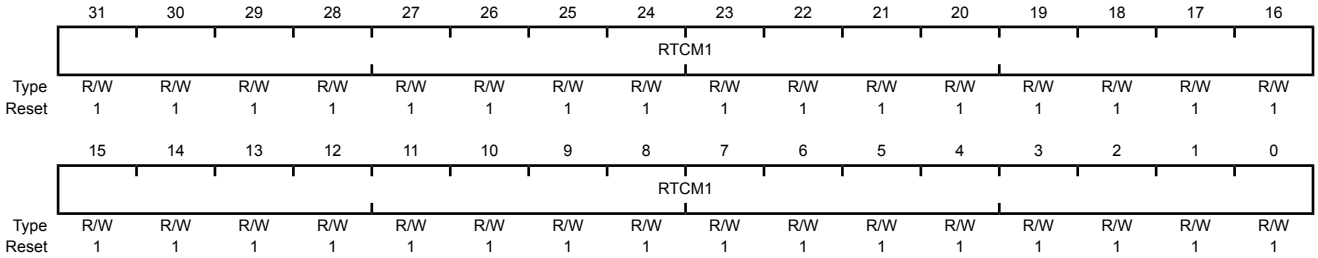
Bit/Field	Name	Type	Reset	Description
31:0	RTCM0	R/W	0xFFFF.FFFF	RTC Match 0 A write loads the value into the RTC match register. A read returns the current match value.

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

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

Hibernation RTC Match 1 (HIBRTCM1)

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



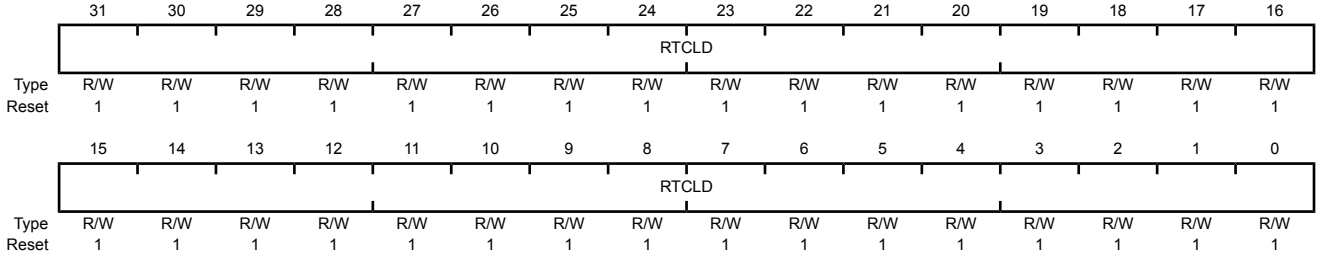
Bit/Field	Name	Type	Reset	Description
31:0	RTCM1	R/W	0xFFFF.FFFF	RTC Match 1 A write loads the value into the RTC match register. A read returns the current match value.

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

This register is the 32-bit value loaded into the RTC counter.

Hibernation RTC Load (HIBRTCLD)

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

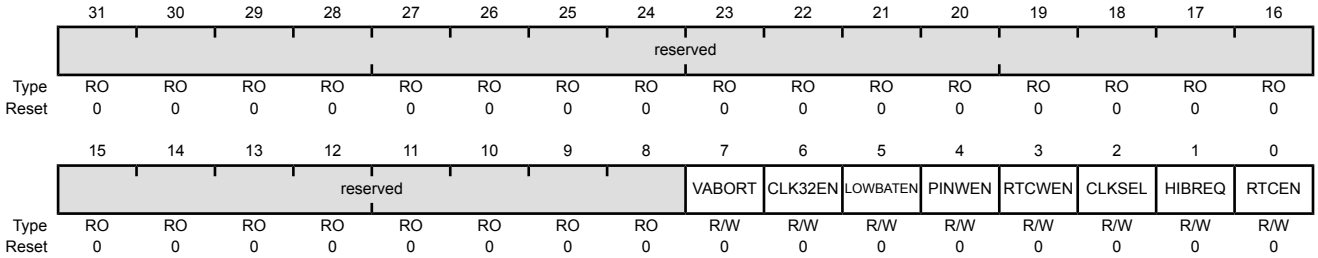


Bit/Field	Name	Type	Reset	Description
31:0	RTCLD	R/W	0xFFFF.FFFF	RTC Load A write loads the current value into the RTC counter (RTCC). A read returns the 32-bit load value.

Register 5: Hibernation Control (HIBCTL), offset 0x010

This register is the control register for the Hibernation module.

Hibernation Control (HIBCTL)
Base 0x400F.C000
Offset 0x010
Type R/W, reset 0x8000.0000



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	VABORT	R/W	0	Power Cut Abort Enable Value Description 0 Power cut occurs during a low-battery alert. 1 Power cut is aborted.
6	CLK32EN	R/W	0	Clocking Enable Value Description 0 Disabled 1 Enabled This bit must be enabled to use the Hibernation module. If a crystal is used, then software should wait 20 ms after setting this bit to allow the crystal to power up and stabilize.
5	LOWBATEN	R/W	0	Low Battery Monitoring Enable Value Description 0 Disabled 1 Enabled When set, low battery voltage detection is enabled ($V_{BAT} < V_{LOWBAT}$).
4	PINWEN	R/W	0	External $\overline{WAKE}$ Pin Enable Value Description 0 Disabled 1 Enabled When set, an external event on the $\overline{WAKE}$ pin will re-power the device.

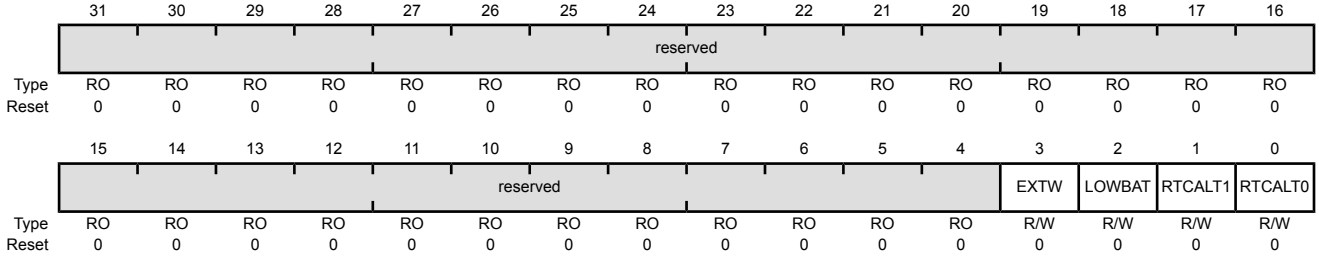
Bit/Field	Name	Type	Reset	Description
3	RTCWEN	R/W	0	RTC Wake-up Enable Value: 0 = Disabled; 1 = Enabled When set, an RTC match event (RTCM0 or RTCM1) will re-power the device based on the RTC counter value matching the corresponding match register 0 or 1.
2	CLKSEL	R/W	0	Hibernation Module Clock Select Value: 0 = Use Divide by 128 output. Use this value for a 4.194304-MHz crystal.; 1 = Use raw output. Use this value for a 32.768-kHz oscillator.
1	HIBREQ	R/W	0	Hibernation Request Value: 0 = Disabled; 1 = Hibernation initiated After a wake-up event, this bit is cleared by hardware.
0	RTCEN	R/W	0	RTC Timer Enable Value: 0 = Disabled; 1 = Enabled

Register 6: Hibernation Interrupt Mask (HIBIM), offset 0x014

This register is the interrupt mask register for the Hibernation module interrupt sources.

Hibernation Interrupt Mask (HIBIM)

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



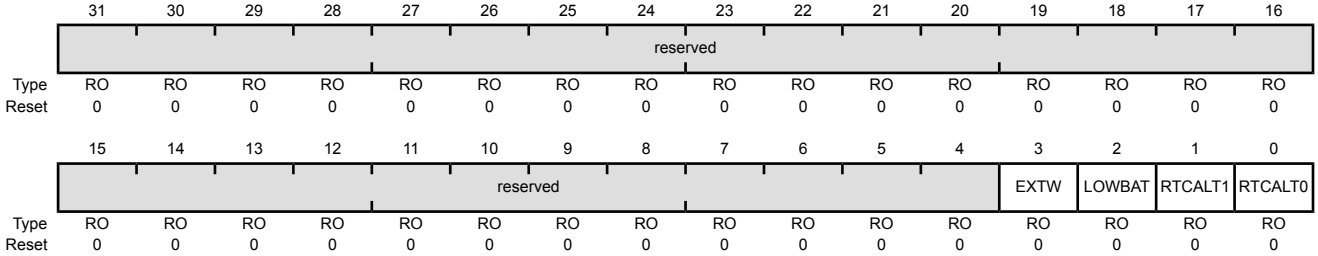
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W	0	External Wake-Up Interrupt Mask Value: 0 = Masked; 1 = Unmasked
2	LOWBAT	R/W	0	Low Battery Voltage Interrupt Mask Value: 0 = Masked; 1 = Unmasked
1	RTCALT1	R/W	0	RTC Alert1 Interrupt Mask Value: 0 = Masked; 1 = Unmasked
0	RTCALT0	R/W	0	RTC Alert0 Interrupt Mask Value: 0 = Masked; 1 = Unmasked

Register 7: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018

This register is the raw interrupt status for the Hibernation module interrupt sources.

Hibernation Raw Interrupt Status (HIBRIS)

Base 0x400F.C000
Offset 0x018
Type RO, reset 0x0000.0000



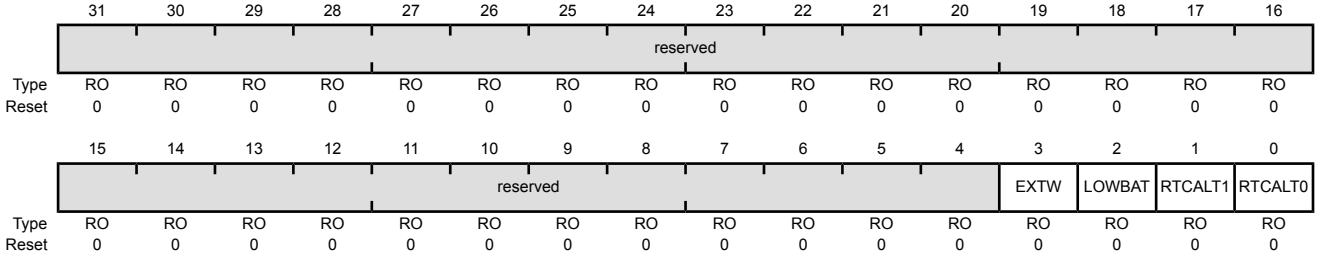
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Raw Interrupt Status
2	LOWBAT	RO	0	Low Battery Voltage Raw Interrupt Status
1	RTCALT1	RO	0	RTC Alert1 Raw Interrupt Status
0	RTCALT0	RO	0	RTC Alert0 Raw Interrupt Status

Register 8: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C

This register is the masked interrupt status for the Hibernation module interrupt sources.

Hibernation Masked Interrupt Status (HIBMIS)

Base 0x400F.C000
Offset 0x01C
Type RO, reset 0x0000.0000



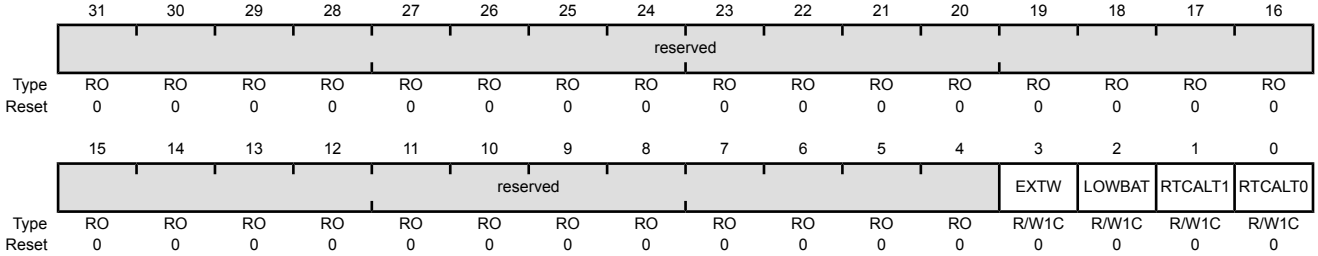
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Masked Interrupt Status
2	LOWBAT	RO	0	Low Battery Voltage Masked Interrupt Status
1	RTCALT1	RO	0	RTC Alert1 Masked Interrupt Status
0	RTCALT0	RO	0	RTC Alert0 Masked Interrupt Status

Register 9: Hibernation Interrupt Clear (HIBIC), offset 0x020

This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources.

Hibernation Interrupt Clear (HIBIC)

Base 0x400F.C000
Offset 0x020
Type R/W1C, reset 0x0000.0000



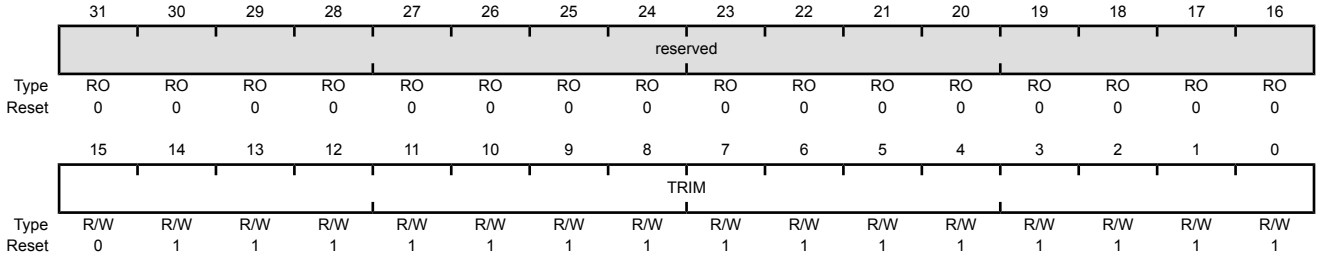
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W1C	0	External Wake-Up Masked Interrupt Clear Reads return an indeterminate value.
2	LOWBAT	R/W1C	0	Low Battery Voltage Masked Interrupt Clear Reads return an indeterminate value.
1	RTCALT1	R/W1C	0	RTC Alert1 Masked Interrupt Clear Reads return an indeterminate value.
0	RTCALT0	R/W1C	0	RTC Alert0 Masked Interrupt Clear 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.

Hibernation RTC Trim (HIBRTCT)

Base 0x400F.C000
Offset 0x024
Type R/W, reset 0x0000.7FFF



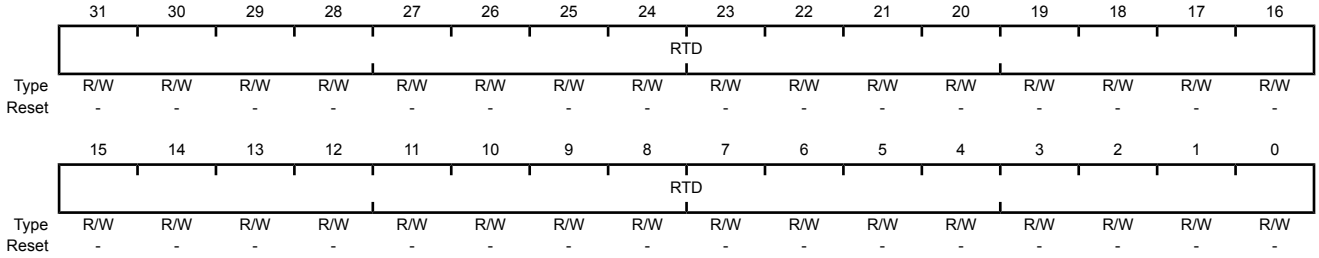
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TRIM	R/W	0x7FFF	RTC Trim Value This value is 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. The compensation is made by software by adjusting the default value of 0x7FFF up or down.

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 state information and does not lose power during a power-cut operation as long as a battery is present.

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 Registers[63:0]

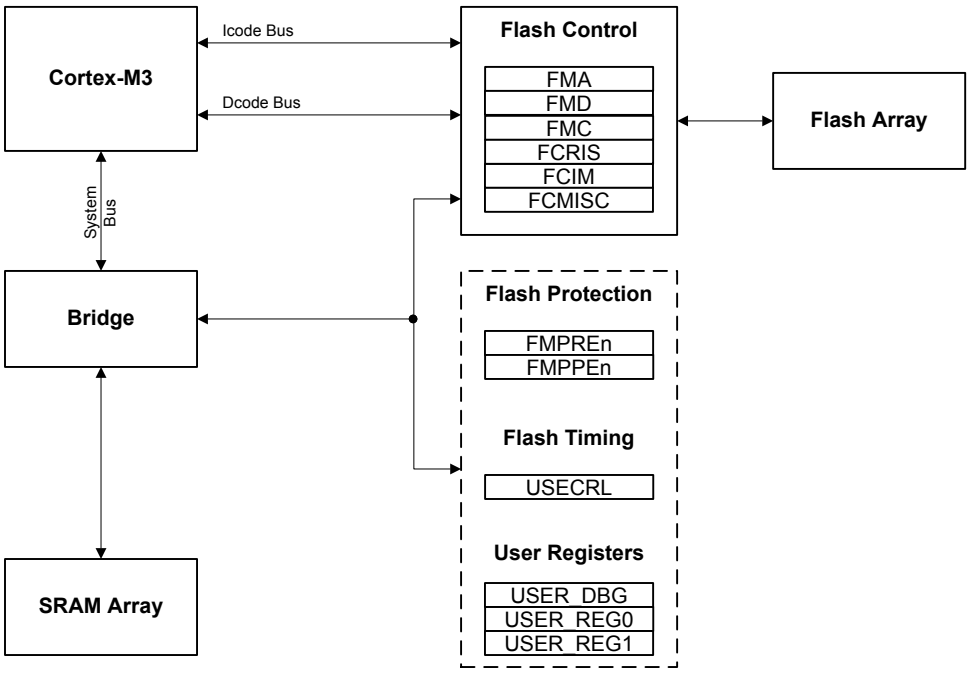
7 Internal Memory

The LM3S8938 microcontroller comes with 64 KB of bit-banded SRAM and 256 KB of flash memory. The flash controller provides a user-friendly interface, making flash programming a simple task. Flash protection can be applied to the flash memory on a 2-KB block basis.

7.1 Block Diagram

Figure 7-1 on page 259 illustrates the Flash functions. The dashed boxes in the figure indicate registers residing in the System Control module rather than the Flash Control module.

Figure 7-1. Flash Block Diagram



7.2 Functional Description

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

7.2.1 SRAM Memory

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 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.

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 75.

7.2.2 Flash Memory

The flash 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. An individual 32-bit word can be programmed to change bits that are currently 1 to a 0. These blocks are paired into a set 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.

See also “Serial Flash Loader” on page 707 for a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface.

7.2.2.1 Flash Memory Timing

The timing for the flash is automatically handled by the flash controller. However, in order to do so, it must know the clock rate of the system in order to time its internal signals properly. The number of clock cycles per microsecond must be provided to the flash controller for it to accomplish this timing. It is software's responsibility to keep the flash controller updated with this information via the **USec Reload (USECRL)** register.

On reset, the **USECRL** register is loaded with a value that configures the flash timing so that it works with the maximum clock rate of the part. If software changes the system operating frequency, the new operating frequency minus 1 (in MHz) must be loaded into **USECRL** before any flash modifications are attempted. For example, if the device is operating at a speed of 20 MHz, a value of 0x13 (20-1) must be written to the **USECRL** register.

7.2.2.2 Flash Memory Protection

The user is provided two forms of flash protection per 2-KB flash blocks in four pairs of 32-bit wide registers. The protection policy for each form is controlled by individual bits (per policy per block) in the **FMPPEn** and **FMPREn** registers.

- **Flash Memory Protection Program Enable (FMPPEn)**: If set, the block may be programmed (written) or erased. If cleared, the 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 260.

Table 7-1. Flash 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.

Table 7-1. Flash Protection Policy Combinations *(continued)*

FMPPEn	FMPREn	Protection
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 263.

7.2.2.3 Execute-Only Protection

Execute-only protection prevents both modification and visibility to a protected flash block. This mode is intended to be used in situations where a device requires debug capability, yet portions of the application space must be protected from external access. An example of this is a company who wishes to sell Stellaris devices with their proprietary software pre-programmed, yet allow the end user to add custom code to an unprotected region of the flash (such as a motor control module with a customizable motor configuration section in flash).

Literal data introduces a complication to the protection mechanism. When C code is compiled and linked, literal data (constants, and so on) is typically placed in the text section, between functions, by the compiler. The literal data is accessed at run time through the use of the LDR instruction, which loads the data from memory using a PC-relative memory address. The execution of the LDR instruction generates a read transaction across the Cortex-M3's DCode bus, which is subject to the execute-only protection mechanism. If the accessed block is marked as execute only, the transaction is blocked, and the processor is prevented from loading the constant data and, therefore, inhibiting correct execution. Therefore, using execute-only protection requires that literal data be handled differently. There are three ways to address this:

1. Use a compiler that allows literal data to be collected into a separate section that is put into one or more read-enabled flash blocks. Note that the LDR instruction may use a PC-relative address—in which case the literal pool cannot be located outside the span of the offset—or the software may reserve a register to point to the base address of the literal pool and the LDR offset is relative to the beginning of the pool.

2. Use a compiler that generates literal data from arithmetic instruction immediate data and subsequent computation.

3. Use method 1 or 2, but in assembly language, if the compiler does not support either method.

7.2.2.4 Read-Only Protection

Read-only protection prevents the contents of the flash block from being re-programmed, while still allowing the content to be read by processor or the debug interface. Note that if a **FMPREn** bit is cleared, all read accesses to the Flash memory block are disallowed, including any data accesses. Care must be taken not to store required data in a Flash memory block that has the associated **FMPREn** bit cleared.

The read-only mode does not prevent read access to the stored program, but it does provide protection against accidental (or malicious) erasure or programming. Read-only is especially useful for utilities like the boot loader when the debug interface is permanently disabled. In such combinations, the boot loader, which provides access control to the Flash memory, is protected from being erased or modified.

7.2.2.5 Permanently Disabling Debug

For extremely sensitive applications, the debug interface to the processor and peripherals can be permanently disabled, blocking all accesses to the device through the JTAG or SWD interfaces. With the debug interface disabled, it is still possible to perform standard IEEE instructions (such as boundary scan operations), but access to the processor and peripherals is blocked.

The DBG0 and DBG1 bits of the **User Debug (USER_DBG)** register control whether the debug interface is turned on or off.

The debug interface should not be permanently disabled without providing some mechanism—such as the boot loader—to provide customer-installable updates or bug fixes. Disabling the debug interface is permanent and cannot be reversed.

7.2.2.6 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 271) 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 270).

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 272).

7.3 Flash Memory Initialization and Configuration

7.3.1 Flash Programming

The Stellaris devices provide a user-friendly interface for flash programming. All erase/program operations are handled via three registers: **FMA**, **FMD**, and **FMC**.

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.

7.3.1.1 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 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.

7.3.1.2 To perform an erase of a 1-KB page

1. Write the page address to the **FMA** register.
2. Write the flash 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.

7.3.1.3 To perform a mass erase of the flash

1. Write the flash 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.

7.3.2 Nonvolatile Register Programming

Note: The **USER_DBG** register requires a POR before the committed changes take effect.

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. Prior to being committed, the register contents are unaffected by any reset condition except power-on reset, which returns the register contents to the original value. 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 contents are permanent, and they cannot be restored to their factory default values.

With the exception of the **USER_DBG** register, the settings in these registers can be tested before committing them to Flash memory. For the **USER_DBG** 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 **USER_DBG** operation to be tried before committing it to nonvolatile memory.

Important: The Flash memory registers can only have bits changed from 1 to 0 by user programming and can only be committed once. After being committed, these registers cannot be restored to their factory default values.

In addition, the **USER_REG0**, **USER_REG1**, **USER_REG2**, **USER_REG3**, and **USER_DBG** 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. These five registers can only be committed once whereas the Flash memory protection registers may be committed multiple times. Table 7-2 on page 264 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
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1
FMPPE2	0x0000.0005	FMPPE2
FMPPE3	0x0000.0007	FMPPE3
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1
USER_REG2	0x8000.0002	USER_REG2
USER_REG3	0x8000.0003	USER_REG3
USER_DBG	0x7510.0000	FMD

7.4 Register Map

Table 7-3 on page 264 lists the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, and **FCMISC** register offsets are relative to the Flash memory control base address of 0x400F.D000. The 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 Control Registers (Flash Control Offset)					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	266
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	267
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	268
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	270
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	271
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	272
Flash Memory Protection Registers (System Control Offset)					
0x130	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	275
0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	275
0x134	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	276
0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	276
0x140	USECRL	R/W	0x31	USec Reload	274
0x1D0	USER_DBG	R/W	0xFFFF.FFFE	User Debug	277
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	278

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

Offset	Name	Type	Reset	Description	See page
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	279
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	280
0x208	FMPRE2	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 2	281
0x20C	FMPRE3	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 3	282
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	283
0x408	FMPPE2	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 2	284
0x40C	FMPPE3	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 3	285

7.5 Flash Register Descriptions (Flash Control Offset)

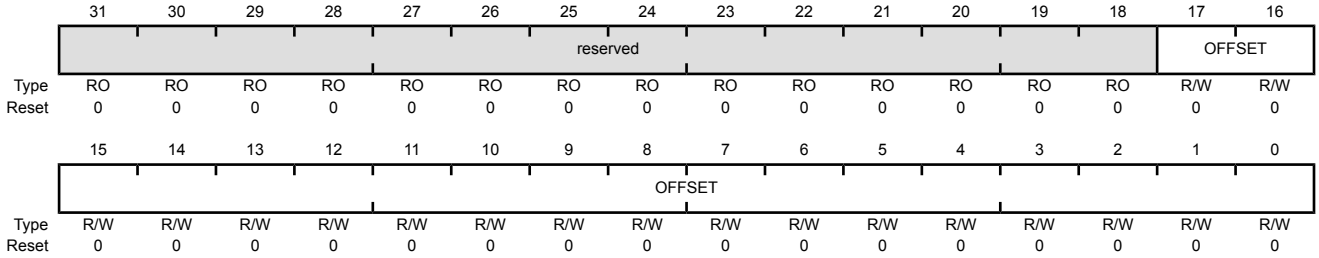
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

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 address and specifies which page 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



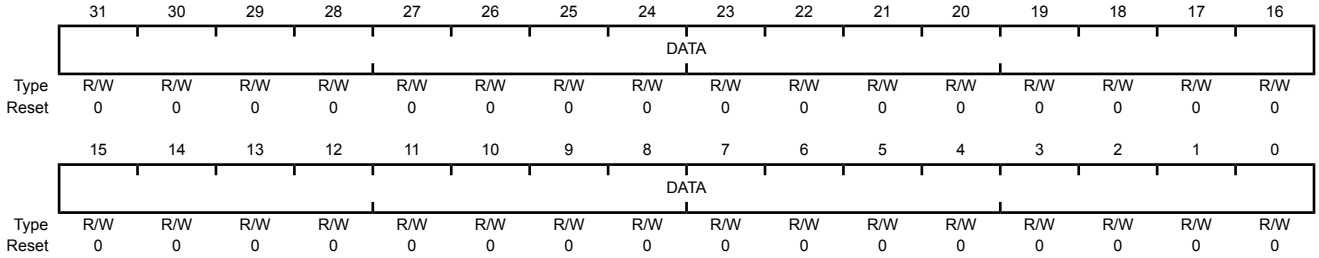
Bit/Field	Name	Type	Reset	Description
31:18	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
17:0	OFFSET	R/W	0x0	Address Offset Address offset in flash where operation is performed, except for nonvolatile registers (see “Nonvolatile Register Programming” on page 263 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 the 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	0x0	Data Value Data value for write operation.

Register 3: Flash Memory Control (FMC), offset 0x008

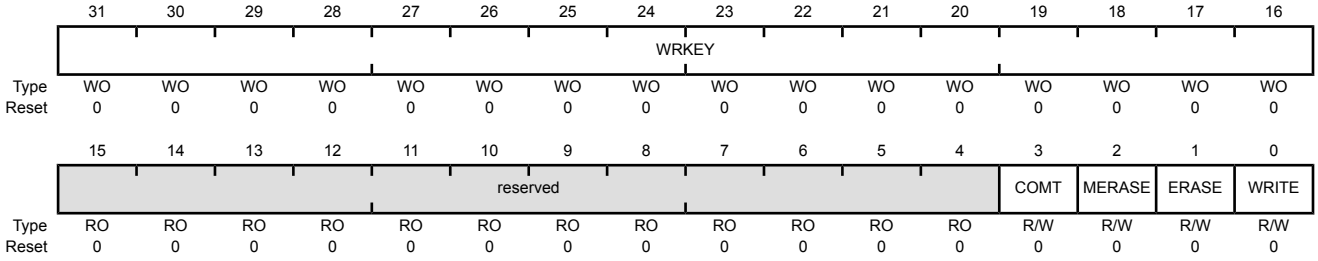
When this register is written, the flash controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 266). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 267) is written.

This is the final register written and initiates the memory operation. There are four control bits in the lower byte of this register that, when set, initiate the memory operation. The most used of these register bits are the ERASE and WRITE bits.

It is a programming error to write multiple control bits and the results of such an operation are unpredictable.

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	0x0	Flash Write Key This field contains a write key, which is used to minimize the incidence of accidental flash writes. The value 0xA442 must be written into this field for a 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	0x0	Software should not rely on the value of 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	COMT	R/W	0	Commit Register Value Commit (write) of register value to nonvolatile storage. A write of 0 has no effect on the state of this bit. If read, the state of the previous commit access is provided. If the previous commit access is complete, a 0 is returned; otherwise, if the commit access is not complete, a 1 is returned. This can take up to 50 µs.
2	MERASE	R/W	0	Mass Erase Flash Memory If this bit is set, the flash main memory of the device is all erased. A write of 0 has no effect on the state of this bit. If read, the state of the previous mass erase access is provided. If the previous mass erase access is complete, a 0 is returned; otherwise, if the previous mass erase access is not complete, a 1 is returned. This can take up to 250 ms.

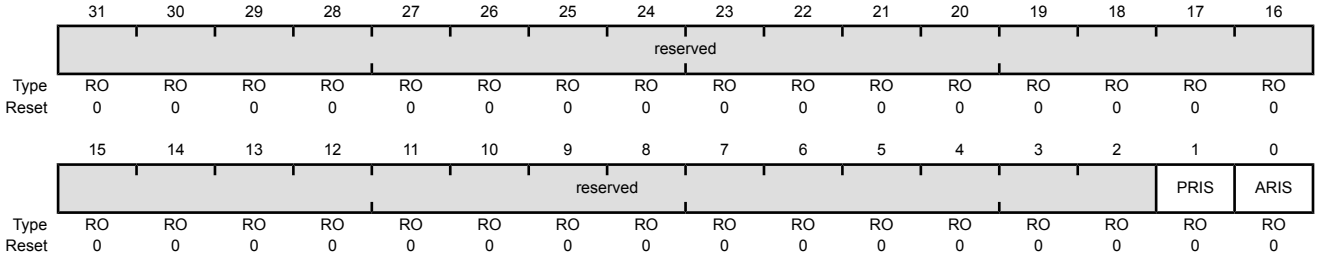
Bit/Field	Name	Type	Reset	Description
1	ERASE	R/W	0	Erase a Page of Flash Memory If this bit is set, the page of flash main memory as specified by the contents of **FMA** is erased. A write of 0 has no effect on the state of this bit. If read, the state of the previous erase access is provided. If the previous erase access is complete, a 0 is returned; otherwise, if the previous erase access is not complete, a 1 is returned. This can take up to 25 ms.
0	WRITE	R/W	0	Write a Word into Flash Memory If this bit is set, the data stored in **FMD** is written into the location as specified by the contents of **FMA**. A write of 0 has no effect on the state of this bit. If read, the state of the previous write update is provided. If the previous write access is complete, a 0 is returned; otherwise, if the write access is not complete, a 1 is returned. This can take up to 50 µs.

Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C

This register indicates that the flash controller has an interrupt condition. An interrupt is only signaled if the corresponding **FCIM** register bit is set.

Flash Controller Raw Interrupt Status (FCRIS)

Base 0x400F.D000
Offset 0x00C
Type RO, reset 0x0000.0000



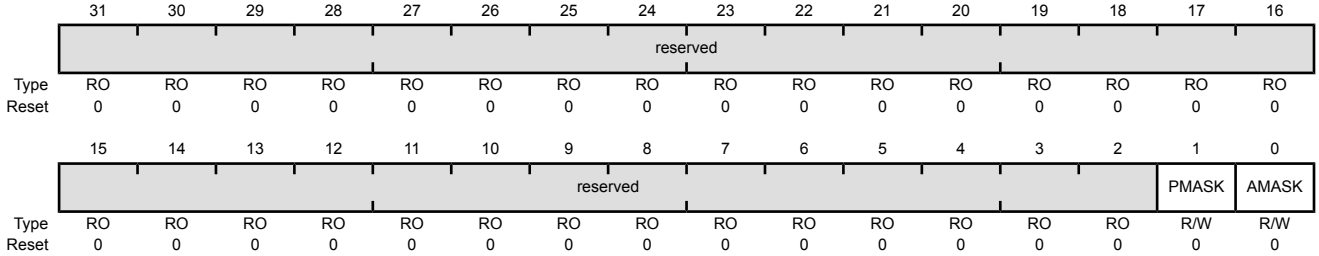
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	Programming Raw Interrupt Status This bit provides status on programming cycles which are write or erase actions generated through the **FMC** register bits (see page 268). Value Description 1 The programming cycle has completed. 0 The programming cycle has not completed. This status is sent to the interrupt controller when the PMASK bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the PMISC bit in the **FCMISC** register.
0	ARIS	RO	0	Access Raw Interrupt Status Value Description 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. 0 No access has tried to improperly program or erase the Flash memory. This status is sent to the interrupt controller when the AMASK bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the AMISC bit in the **FCMISC** register.

Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010

This register controls whether the flash 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	0x0	Software should not rely on the value of 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 This bit controls the reporting of the programming raw interrupt status to the interrupt controller. Value Description 1 An interrupt is sent to the interrupt controller when the PRIS bit is set. 0 The PRIS interrupt is suppressed and not sent to the interrupt controller.
0	AMASK	R/W	0	Access Interrupt Mask This bit controls the reporting of the access raw interrupt status to the interrupt controller. Value Description 1 An interrupt is sent to the interrupt controller when the ARIS bit is set. 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

	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														PMISC	AMISC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:2	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	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed. Writing a 1 to this bit clears `PMISC` and also the `PRIS` bit in the **FCRIS** register (see page 270). 0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred. A write of 0 has no effect on the state of this bit.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear Value Description 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. Writing a 1 to this bit clears `AMISC` and also the `ARIS` bit in the **FCRIS** register (see page 270). 0 When read, a 0 indicates that no improper accesses have occurred. A write of 0 has no effect on the state of this bit.

7.6 Flash Register Descriptions (System Control Offset)

The remainder of this section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the System Control base address of 0x400F.E000.

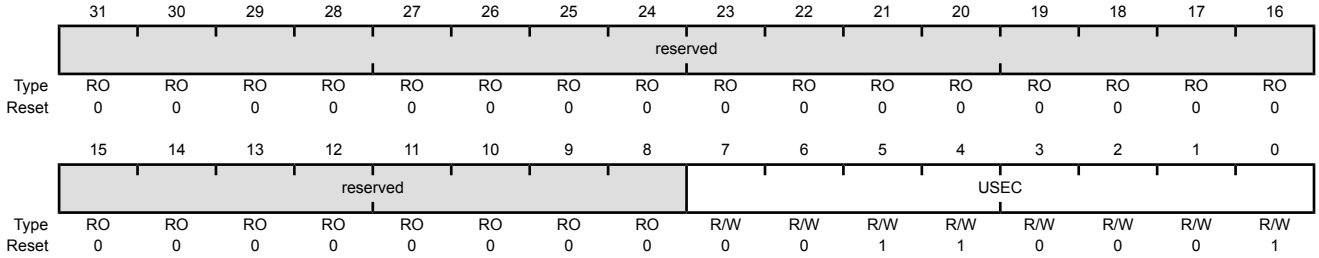
Register 7: USec Reload (USECRL), offset 0x140

Note: Offset is relative to System Control base address of 0x400F.E000

This register is provided as a means of creating a 1-µs tick divider reload value for the flash controller. The internal flash has specific minimum and maximum requirements on the length of time the high voltage write pulse can be applied. It is required that this register contain the operating frequency (in MHz -1) whenever the flash is being erased or programmed. The user is required to change this value if the clocking conditions are changed for a flash erase/program operation.

USec Reload (USECRL)

Base 0x400F.E000
Offset 0x140
Type R/W, reset 0x31



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	USEC	R/W	0x31	Microsecond Reload Value MHz -1 of the controller clock when the flash is being erased or programmed. If the maximum system frequency is being used, USEC should be set to 0x31 (50 MHz) whenever the flash is being erased or programmed.

Register 8: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

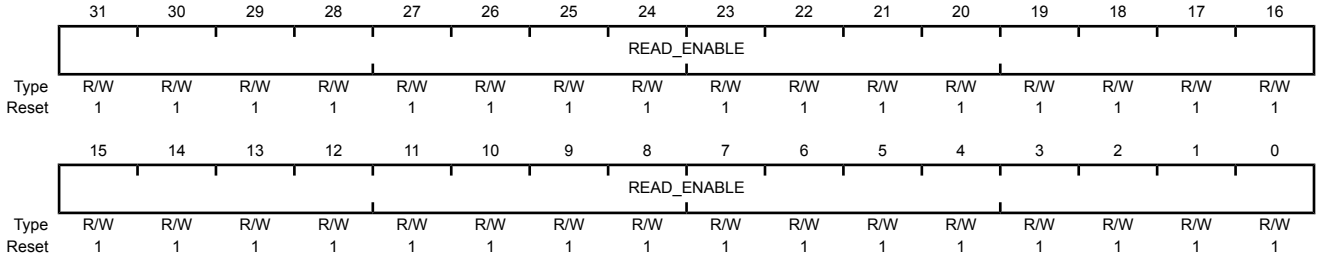
Note: This register is aliased for backwards compatability.

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. 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. Enables 2-KB Flash memory blocks to be executed or read. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.

Value	Description
0xFFFFFFFF	Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB.

Register 9: 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. 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 Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB.

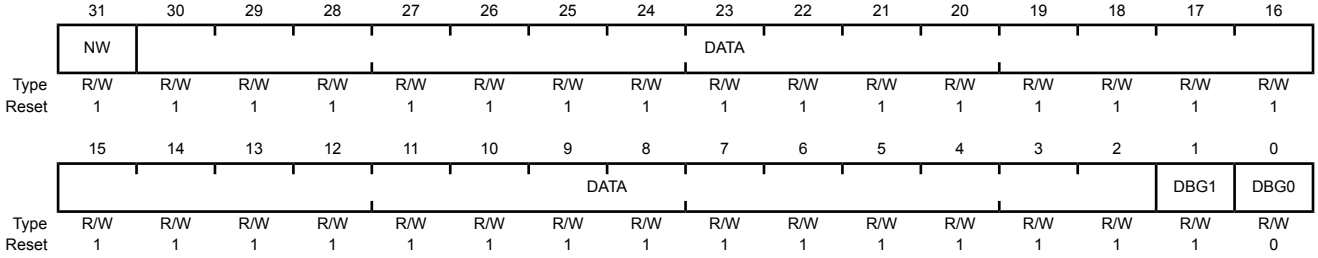
Register 10: User Debug (USER_DBG), offset 0x1D0

Note: Offset is relative to System Control base address of 0x400FE000.

This register provides a write-once mechanism to disable external debugger access to the device in addition to 27 additional bits of user-defined data. 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. Changing the DBG1 bit to 0 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. Once committed, this register cannot be restored to the factory default value.

User Debug (USER_DBG)

Base 0x400F.E000
Offset 0x1D0
Type R/W, reset 0xFFFF.FFFE



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	User Debug Not Written 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.
30:2	DATA	R/W	0x1FFFFFFF	User Data Contains the user data value. This field is initialized to all 1s and can only be committed once.
1	DBG1	R/W	1	Debug Control 1 The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.
0	DBG0	R/W	0	Debug Control 0 The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.

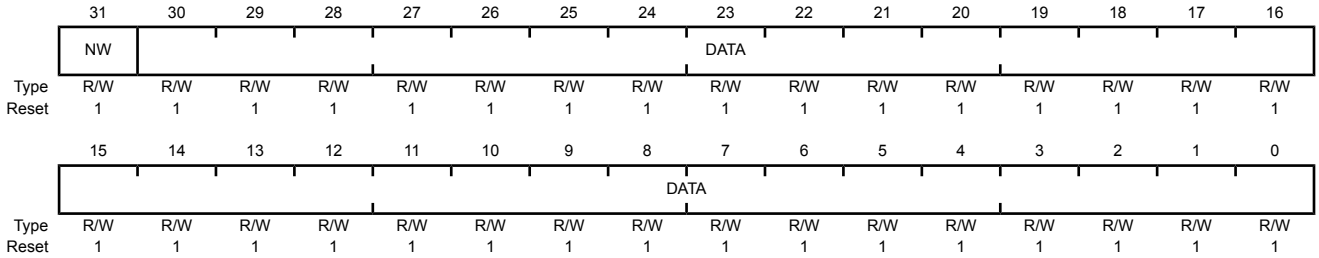
Register 11: User Register 0 (USER_REG0), offset 0x1E0

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. Once committed, this register cannot be restored to the factory default value.

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 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 Contains the user data value. This field is initialized to all 1s and can only be committed once.

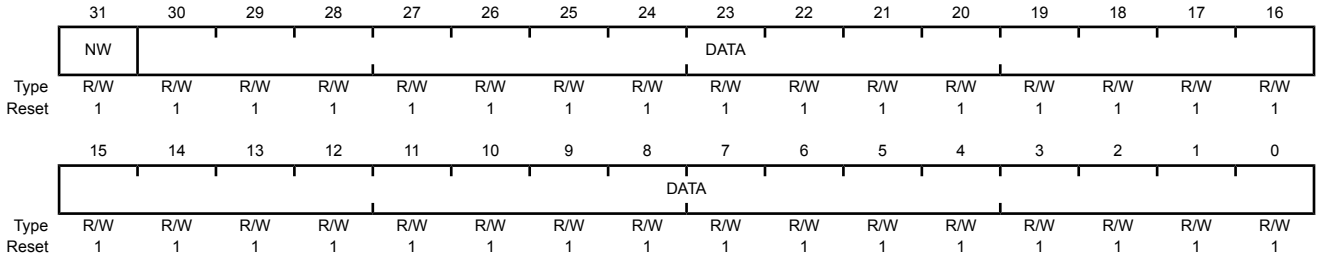
Register 12: User Register 1 (USER_REG1), offset 0x1E4

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 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. Once committed, this register cannot be restored to the factory default value.

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 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 Contains the user data value. This field is initialized to all 1s and can only be committed once.

Register 13: 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. 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

	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	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
	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	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable. Enables 2-KB Flash memory blocks to be executed or read. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.

Value	Description
0xFFFFFFFF	Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB.

Register 14: 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). 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). 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

	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	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
	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	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0xFFFFFFFF Enables 256 KB of flash.

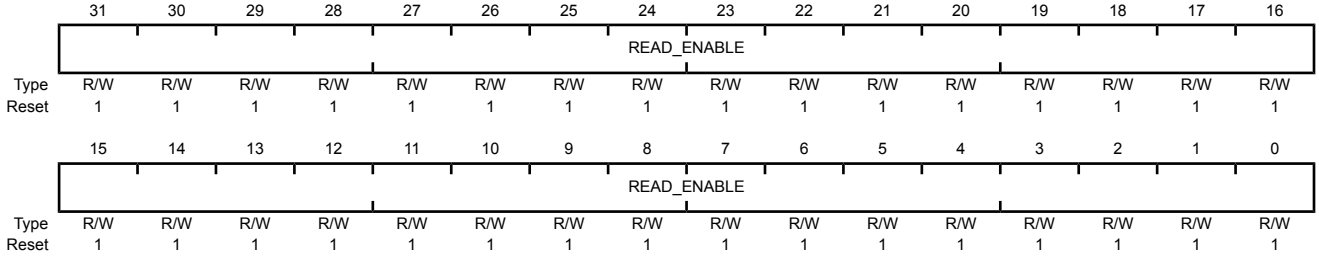
Register 15: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

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). 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. 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 Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0xFFFFFFFF Enables 256 KB of flash.

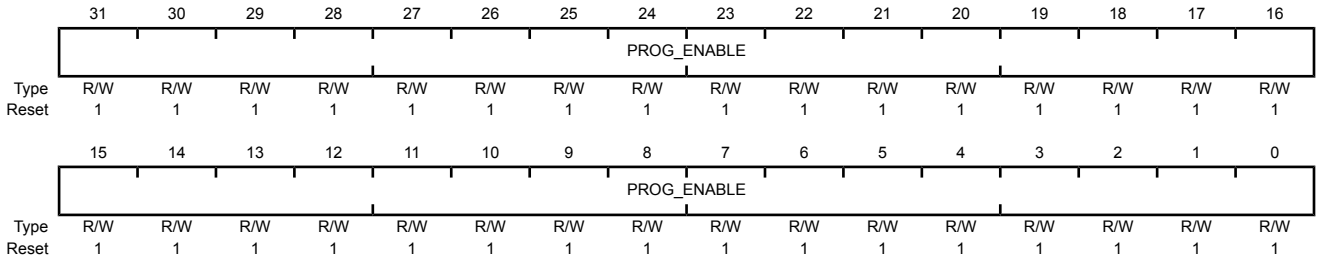
Register 16: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

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. 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



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable

Value	Description
0xFFFFFFFF	Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB.

Register 17: 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). 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. 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

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 Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0xFFFFFFFF Enables 256 KB of flash.

Register 18: 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). 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. 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 Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0xFFFFFFFF Enables 256 KB of flash.

8 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of seven physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G). The GPIO module supports 3-38 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- 3-38 GPIOs, depending on configuration
- 5-V-tolerant in input configuration
- Fast toggle capable of a change every two clock cycles
- 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 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 be configured with an 18-mA pad drive for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

8.1 Signal Description

GPIO signals have alternate hardware functions. Table 8-4 on page 289 and Table 8-5 on page 290 list the GPIO pins and the analog and digital alternate functions. The AINx 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. Other analog signals are 5-V tolerant and are connected directly to their circuitry (C0-, C0+, C1-, C1+, C2-, C2+). These signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. 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 enoding 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.

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 four JTAG/SWD 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 8-1. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^2C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

Table 8-2. GPIO Pins and Alternate Functions (100LQFP)

IO	Pin Number	Multiplexed Function	Multiplexed Function
PA0	26	U0Rx	
PA1	27	U0Tx	
PA2	28	SSI0Clk	
PA3	29	SSI0Fss	
PA4	30	SSI0Rx	
PA5	31	SSI0Tx	
PA6	34	I2C1SCL	
PA7	35	I2C1SDA	
PB0	66	CCP0	
PB1	67	CCP2	
PB2	70	I2C0SCL	
PB3	71	I2C0SDA	
PB4	92	C0-	
PB5	91	C1-	
PB6	90	C0+	C0o
PB7	89	$\overline{TRST}$	
PC0	80	TCK	SWCLK
PC1	79	TMS	SWDIO
PC2	78	TDI	
PC3	77	TDO	SWO
PC4	25	CCP5	
PC5	24	C1+	C1o
PC6	23	C2+	C2o
PC7	22	C2-	
PD0	10	CAN0Rx	
PD1	11	CAN0Tx	
PD2	12	U1Rx	
PD3	13	U1Tx	
PE0	72	CCP3	

Table 8-2. GPIO Pins and Alternate Functions (100LQFP) *(continued)*

IO	Pin Number	Multiplexed Function	Multiplexed Function
PE1	73		
PE2	74	CCP4	
PE3	75	CCP1	
PF0	47		
PF1	61		
PF2	60	LED1	
PF3	59	LED0	
PG0	19	U2Rx	
PG1	18	U2Tx	

Table 8-3. GPIO Pins and Alternate Functions (108BGA)

IO	Pin Number	Multiplexed Function	Multiplexed Function
PA0	L3	U0Rx	
PA1	M3	U0Tx	
PA2	M4	SSI0Clk	
PA3	L4	SSI0Fss	
PA4	L5	SSI0Rx	
PA5	M5	SSI0Tx	
PA6	L6	I2C1SCL	
PA7	M6	I2C1SDA	
PB0	E12	CCP0	
PB1	D12	CCP2	
PB2	C11	I2C0SCL	
PB3	C12	I2C0SDA	
PB4	A6	C0-	
PB5	B7	C1-	
PB6	A7	C0+	C0o
PB7	A8	$\overline{\text{TRST}}$	
PC0	A9	TCK	SWCLK
PC1	B9	TMS	SWDIO
PC2	B8	TDI	
PC3	A10	TDO	SWO
PC4	L1	CCP5	
PC5	M1	C1+	C1o
PC6	M2	C2+	C2o
PC7	L2	C2-	
PD0	G1	CAN0Rx	
PD1	G2	CAN0Tx	
PD2	H2	U1Rx	
PD3	H1	U1Tx	
PE0	A11	CCP3	

Table 8-3. GPIO Pins and Alternate Functions (108BGA) *(continued)*

IO	Pin Number	Multiplexed Function	Multiplexed Function
PE1	B12		
PE2	B11	CCP4	
PE3	A12	CCP1	
PF0	M9		
PF1	H12		
PF2	J11	LED1	
PF3	J12	LED0	
PG0	K1	U2Rx	
PG1	K2	U2Tx	

Table 8-4. GPIO Signals (100LQFP)

Pin Name	Pin Number	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.
PB1	67	I/O	TTL	GPIO port B bit 1.
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.
PE0	72	I/O	TTL	GPIO port E bit 0.

Table 8-4. GPIO Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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.
PF0	47	I/O	TTL	GPIO port F bit 0.
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.
PG0	19	I/O	TTL	GPIO port G bit 0.
PG1	18	I/O	TTL	GPIO port G bit 1.

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

Table 8-5. GPIO Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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.
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.
PB1	D12	I/O	TTL	GPIO port B bit 1.
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.

Table 8-5. GPIO Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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.
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.
PG0	K1	I/O	TTL	GPIO port G bit 0.
PG1	K2	I/O	TTL	GPIO port G bit 1.

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

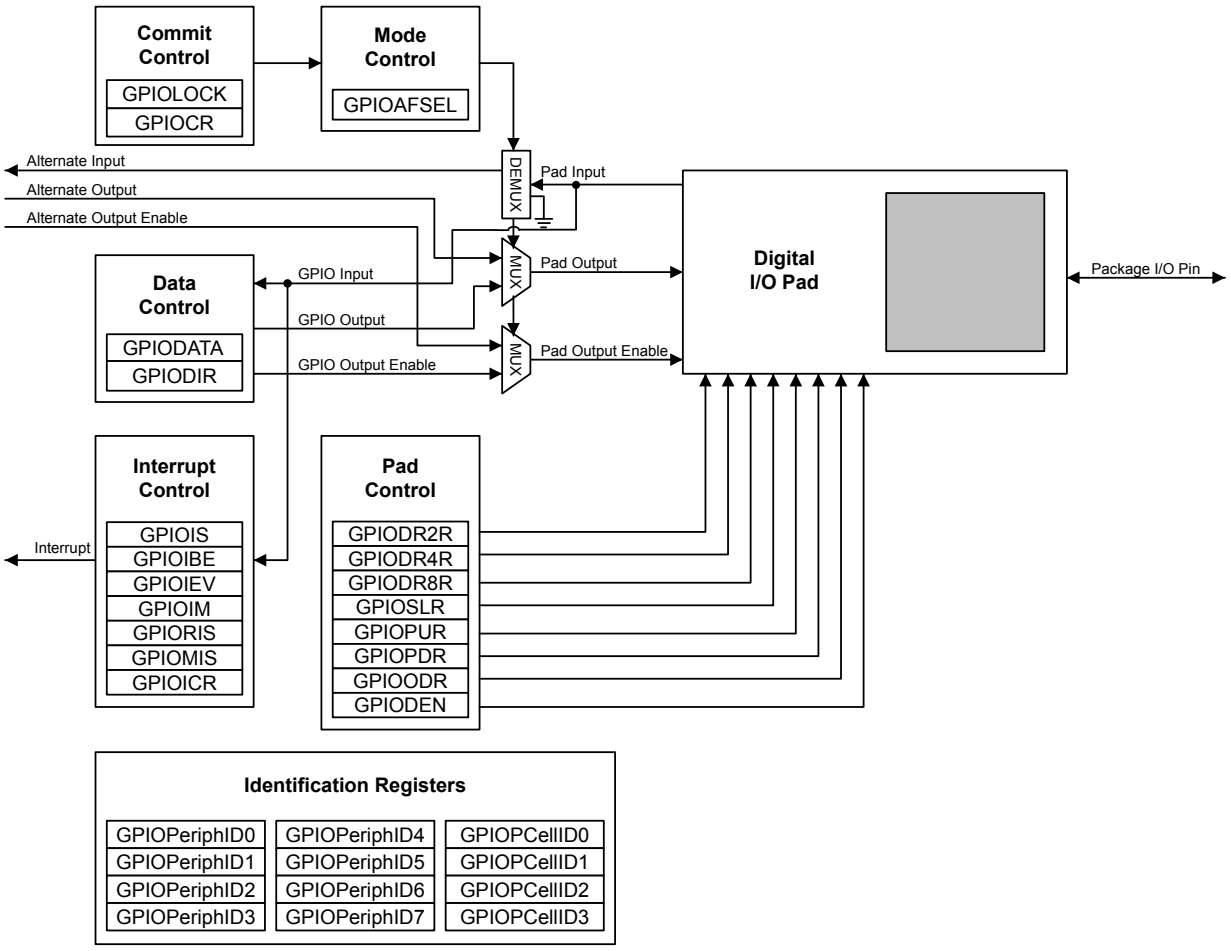
8.2 Functional Description

Important: All GPIO pins are tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0), with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). The JTAG/SWD pins default to their JTAG/SWD functionality (**GPIOAFSEL**=1, **GPIODEN=1** and **GPIOPUR**=1). A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts both groups of pins back to their default state.

While debugging systems where PB7 is being used as a GPIO, care must be taken to ensure that a low value is not applied to the pin when the part is reset. Because PB7 reverts to the $\overline{\text{TRST}}$ function after reset, a Low value on the pin causes the JTAG controller to be reset, resulting in a loss of JTAG communication.

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 8-1 on page 292). The LM3S8938 microcontroller contains seven ports and thus seven of these physical GPIO blocks.

Figure 8-1. GPIO Port Block Diagram



8.2.1 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.

8.2.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 300) is used to configure each individual pin as an input or output. When the data direction bit is set to 0, the GPIO is configured as an input and the corresponding data register bit will capture and store the value on the GPIO port. When the data direction bit is set to 1, the GPIO is configured as an output and the corresponding data register bit will be driven out on the GPIO port.

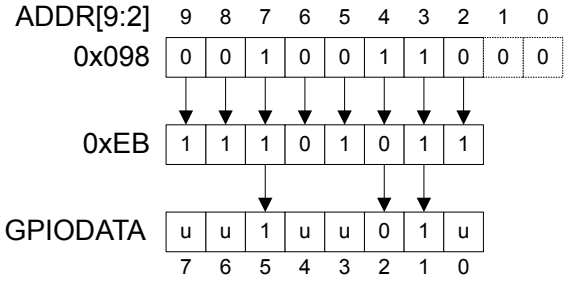
8.2.1.2 Data Register Operation

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 299) by using bits [9:2] of the address bus as a mask. This allows software drivers to modify individual GPIO pins in a single instruction, without affecting the state of the other pins. This is in contrast to the "typical" method of doing a read-modify-write operation to set or clear an individual GPIO pin. To accommodate 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 to 1, the value of the **GPIODATA** register is altered. If it is cleared to 0, it is left unchanged.

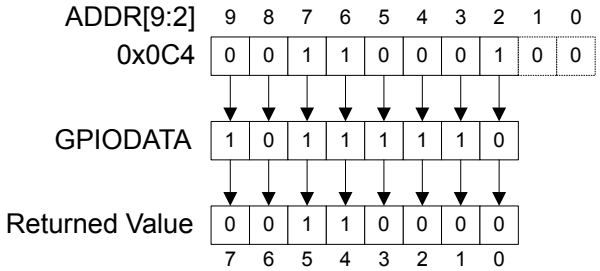
For example, writing a value of 0xEB to the address GPIODATA + 0x098 would yield as shown in Figure 8-2 on page 293, where u is data unchanged by the write.

Figure 8-2. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set to 1, the value is read. If the address bit associated with the data bit is set to 0, it is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 8-3 on page 293.

Figure 8-3. GPIODATA Read Example



8.2.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. With these registers, it is possible 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, it is assumed that the external source holds the level constant for the interrupt to be recognized by the controller.

Three registers are required to define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 301)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 302)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 303)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 304).

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 305 and page 306). As the name implies, the **GPIOMIS** register only shows interrupt

conditions that are allowed to be passed to the controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the controller.

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 to 1), not only is an interrupt for PortB generated, but 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.

If no other PortB pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** register can disable the PortB interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the PortB interrupt handler needs to 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 PortB interrupt handler must poll the ADC registers until the conversion is completed. See page 109 for more information.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 307).

When programming the following interrupt control registers, the interrupts should be masked (**GPIOIM** set to 0). Writing any value to an interrupt control register (**GPIOIS**, **GPIOIBE**, or **GPIOIEV**) can generate a spurious interrupt if the corresponding bits are enabled.

8.2.3 Mode Control

The GPIO pins can be controlled by either hardware or software. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 308), the pin state is controlled by its alternate function (that is, the peripheral). Software control corresponds to GPIO mode, where the **GPIODATA** register is used to read/write the corresponding pins.

8.2.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the five JTAG/SWD pins (PB7 and PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 308) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 318) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 319) have been set to 1.

8.2.5 Pad Control

The pad control registers allow for GPIO pad configuration by software 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 enable.

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.

8.2.6 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.

8.3 Initialization and Configuration

To use the GPIO, the peripheral clock must be enabled by setting the appropriate GPIO Port bit field (`GPIOn`) in the **RCGC2** register.

On reset, all GPIO pins (except for the five JTAG pins) are configured out of reset to be undriven (tristate): **GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0. Table 8-6 on page 295 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 8-7 on page 295 shows how a rising edge interrupt would be configured for pin 2 of a GPIO port.

Table 8-6. 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
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?

a. X=Ignored (don't care bit)

?=Can be either 0 or 1, depending on the configuration

Table 8-7. 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 1=level	X	X	X	X	X	0	X	X

Table 8-7. GPIO Interrupt Configuration Example *(continued)*

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value[a]							
		7	6	5	4	3	2	1	0
GPIOIBE	0=single edge 1=both edges	X	X	X	X	X	0	X	X
GPIOIEV	0=Low level, or negative edge 1=High level, or positive edge	X	X	X	X	X	1	X	X
GPIOIM	0=masked 1=not masked	0	0	0	0	0	1	0	0

a. X=Ignored (don't care bit)

8.4 Register Map

Table 8-8 on page 297 lists the GPIO registers. The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A: 0x4000.4000
- GPIO Port B: 0x4000.5000
- GPIO Port C: 0x4000.6000
- GPIO Port D: 0x4000.7000
- GPIO Port E: 0x4002.4000
- GPIO Port F: 0x4002.5000
- GPIO Port G: 0x4002.6000

Note that the GPIO module clock must be enabled before the registers can be programmed (see page 228). There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

Important: The 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 those unconnected bits has no effect, and reading those unconnected bits returns no meaningful data.

Note: The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins are currently 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 five JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as a GPIO, these five pins default 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 8-8. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	299
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	300
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	301
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	302
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	303
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	304
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	305
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	306
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	307
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	308
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	310
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	311
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	312
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	313
0x510	GPIOPUR	R/W	-	GPIO Pull-Up Select	314
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	315
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	316
0x51C	GPIODEN	R/W	-	GPIO Digital Enable	317
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	318
0x524	GPIOCR	-	-	GPIO Commit	319
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	321
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	322
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	323
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	324
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	325
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	326
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	327
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	328

Table 8-8. GPIO Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	329
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	330
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	331
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	332

8.5 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 300).

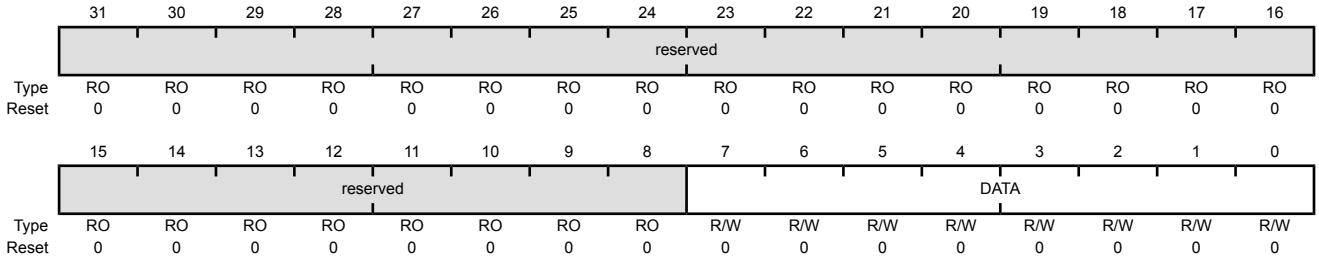
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be High. Otherwise, the bit values remain unchanged by the write.

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 1 in the address mask cause the corresponding bits in **GPIODATA** to be read, and bits that are 0 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x000
Type R/W, reset 0x0000.0000



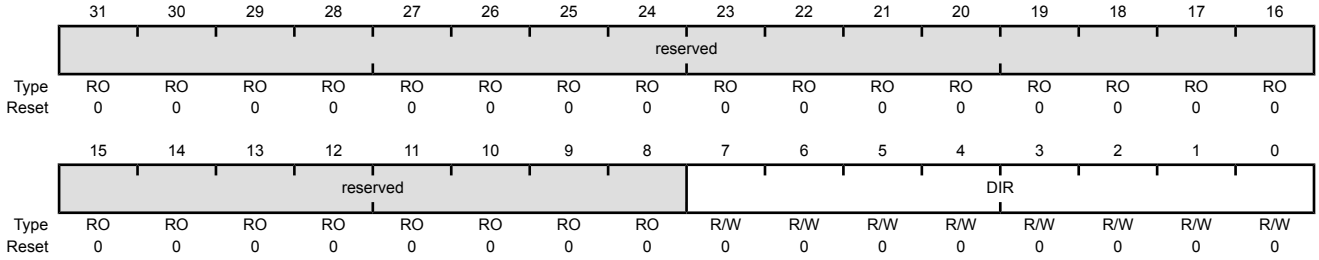
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	GPIO Data This register is virtually 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 the data written to the registers are masked by the eight address lines `ipaddr[9:2]`. Reads from this register return its current state. Writes to this register only affect bits that are not masked by `ipaddr[9:2]` and are configured as outputs. See “Data Register Operation” on page 292 for examples of reads and writes.

Register 2: GPIO Direction (GPIODIR), offset 0x400

The **GPIODIR** register is the data direction register. Bits set to 1 in the **GPIODIR** register configure the corresponding pin to be an output, while bits set to 0 configure the pins to be inputs. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIO Direction (GPIODIR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x400
Type R/W, reset 0x0000.0000



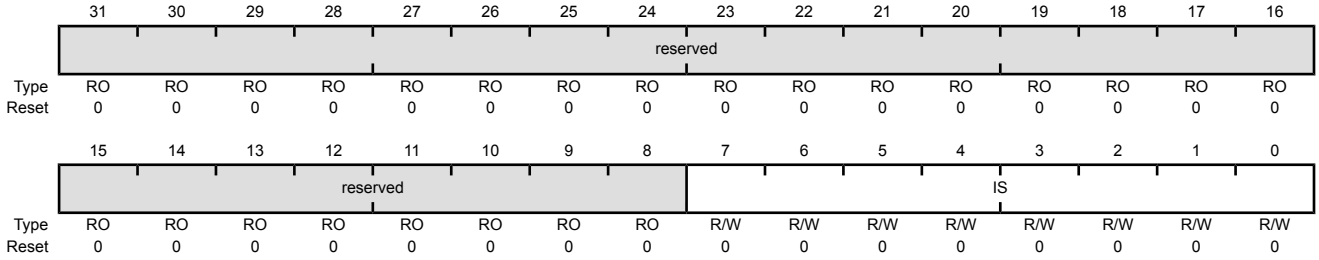
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction The `DIR` values are defined as follows: Value Description 0 Pins are inputs. 1 Pins are outputs.

Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404

The **GPIOIS** register is the interrupt sense register. Bits set to 1 in **GPIOIS** configure the corresponding pins to detect levels, while bits set to 0 configure the pins to detect edges. All bits are cleared by a reset.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x404
Type R/W, reset 0x0000.0000



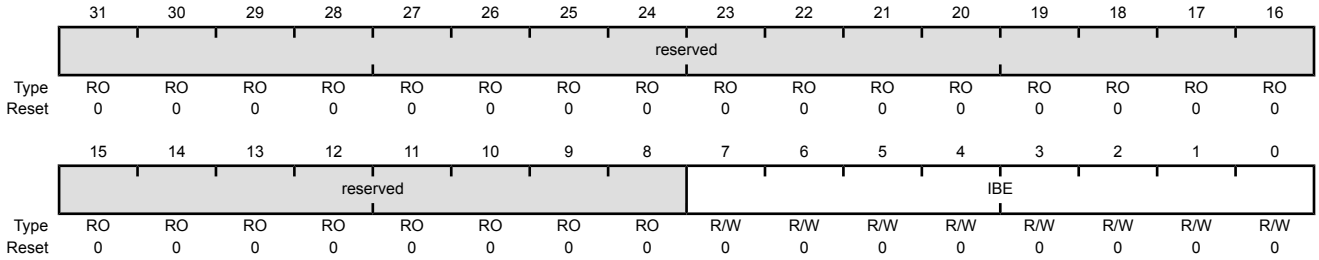
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	R/W	0x00	GPIO Interrupt Sense The `IS` values are defined as follows: Value Description 0 Edge on corresponding pin is detected (edge-sensitive). 1 Level on corresponding pin is detected (level-sensitive).

Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408

The **GPIOIBE** register is the interrupt both-edges register. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 301) is set to detect edges, bits set to High in **GPIOIBE** configure the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 303). Clearing a bit configures the pin to be controlled by **GPIOIEV**. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x408
Type R/W, reset 0x0000.0000



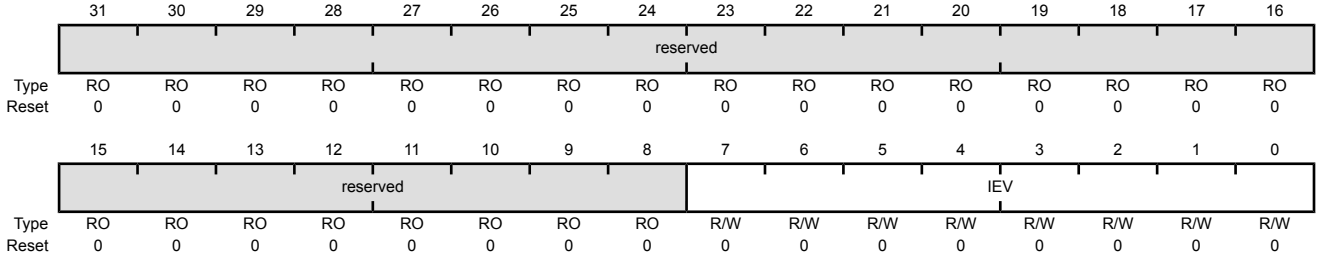
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges The IBE values are defined as follows: Value Description 0 Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 303). 1 Both edges on the corresponding pin trigger an interrupt. **Note:** Single edge is determined by the corresponding bit in **GPIOIEV**.

Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C

The **GPIOIEV** register is the interrupt event register. Bits set to High in **GPIOIEV** configure 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 301). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in **GPIOIS**. All bits are cleared by a reset.

GPIO Interrupt Event (GPIOIEV)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x40C
Type R/W, reset 0x0000.0000



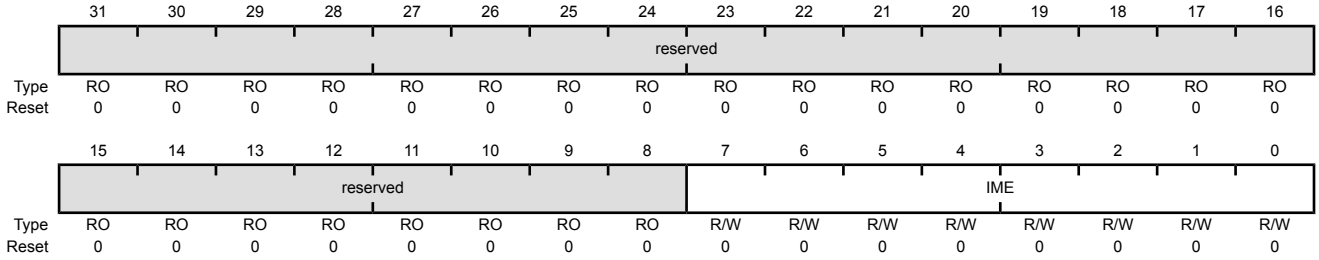
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	R/W	0x00	GPIO Interrupt Event The `IEV` values are defined as follows: Value Description 0 Falling edge or Low levels on corresponding pins trigger interrupts. 1 Rising edge or High levels on corresponding pins trigger interrupts.

Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410

The **GPIOIM** register is the interrupt mask register. Bits set to High in **GPIOIM** allow the corresponding pins to trigger their individual interrupts and the combined **GPIOINTR** line. Clearing a bit disables interrupt triggering on that pin. All bits are cleared by a reset.

GPIO Interrupt Mask (GPIOIM)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x410
Type R/W, reset 0x0000.0000



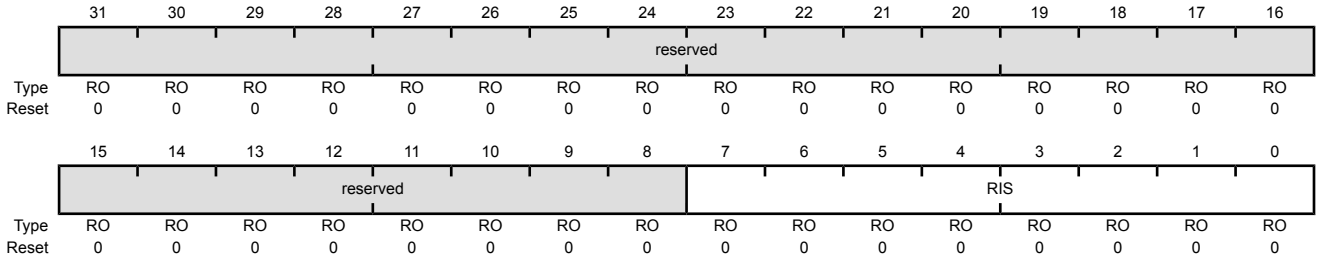
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable The `IME` values are defined as follows: Value Description 0 Corresponding pin interrupt is masked. 1 Corresponding pin interrupt is not masked.

Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414

The **GPIORIS** register is the raw interrupt status register. Bits read High in **GPIORIS** reflect the status of interrupt trigger conditions detected (raw, prior to masking), indicating that all the requirements have been met, before they are finally allowed to trigger by the **GPIO Interrupt Mask (GPIOIM)** register (see page 304). Bits read as zero indicate that corresponding input pins have not initiated an interrupt. All bits are cleared by a reset.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x414
Type RO, 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	RIS	RO	0x00	GPIO Interrupt Raw Status Reflects the status of interrupt trigger condition detection on pins (raw, prior to masking). The `RIS` values are defined as follows: Value Description 0 Corresponding pin interrupt requirements not met. 1 Corresponding pin interrupt has met requirements.

Register 8: GPIO Masked Interrupt Status (GPIOMIS), offset 0x418

The **GPIOMIS** register is the masked interrupt status register. Bits read High in **GPIOMIS** reflect the status of input lines triggering an interrupt. Bits read as Low indicate that either no interrupt has been generated, or the interrupt is masked.

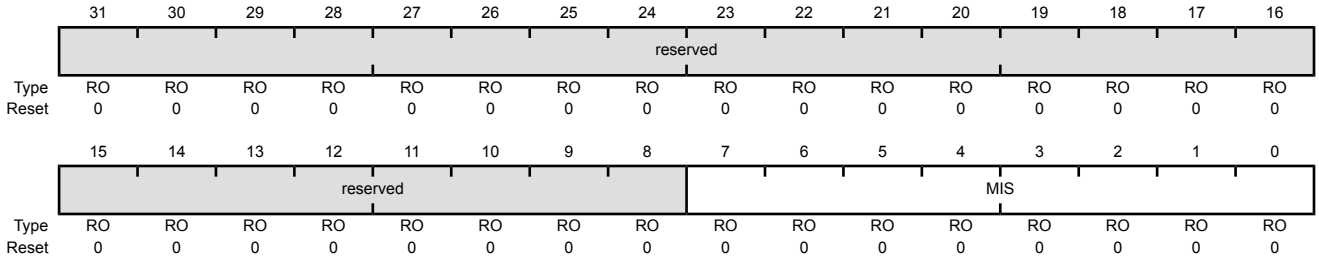
In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated.

If no other PortB pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** register can disable the PortB interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the PortB interrupt handler needs to 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 PortB interrupt handler must poll the ADC registers until the conversion is completed. See page 109 for more information.

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x418
Type RO, reset 0x0000.0000



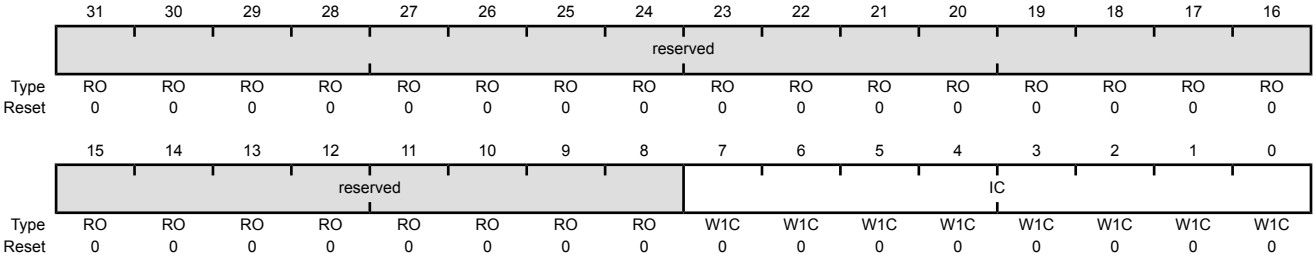
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status Masked value of interrupt due to corresponding pin. The MIS values are defined as follows: Value Description 0 Corresponding GPIO line interrupt not active. 1 Corresponding GPIO line asserting interrupt.

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 edge detection logic register. Writing a 0 has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x41C
Type W1C, 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	IC	W1C	0x00	GPIO Interrupt Clear The `IC` values are defined as follows: Value Description 0 Corresponding interrupt is unaffected. 1 Corresponding interrupt is cleared.

Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420

The **GPIOAFSEL** register is the mode control select register. Writing a 1 to any bit in this register selects the hardware control for the corresponding GPIO line. All bits are cleared by a reset, therefore no GPIO line is set to hardware control by default.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the five JTAG/SWD pins (PB7 and PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 308) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 318) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 319) have been set to 1.

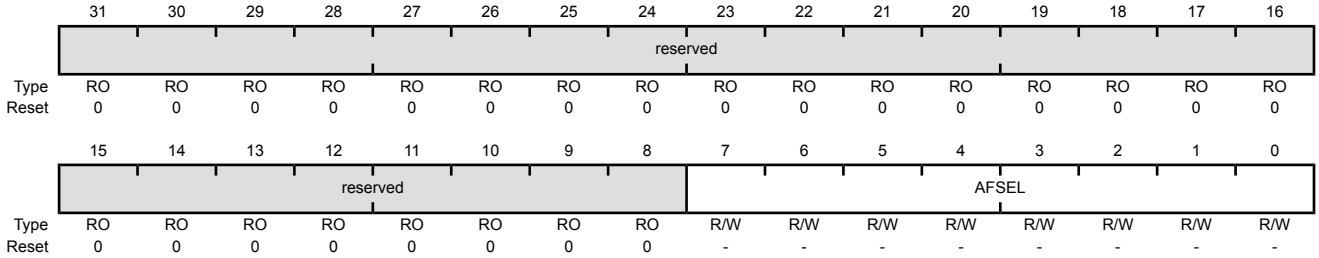
Important: All GPIO pins are tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0), with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). The JTAG/SWD pins default to their JTAG/SWD functionality (**GPIOAFSEL**=1, **GPIODEN=1** and **GPIOPUR**=1). A Power-On-Reset (POR) or asserting RST puts both groups of pins back to their default state.

While debugging systems where PB7 is being used as a GPIO, care must be taken to ensure that a low value is not applied to the pin when the part is reset. Because PB7 reverts to the TRST function after reset, a Low value on the pin causes the JTAG controller to be reset, resulting in a loss of JTAG communication.

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. This may lock the debugger out of the part. This can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x420
Type R/W, reset -



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.

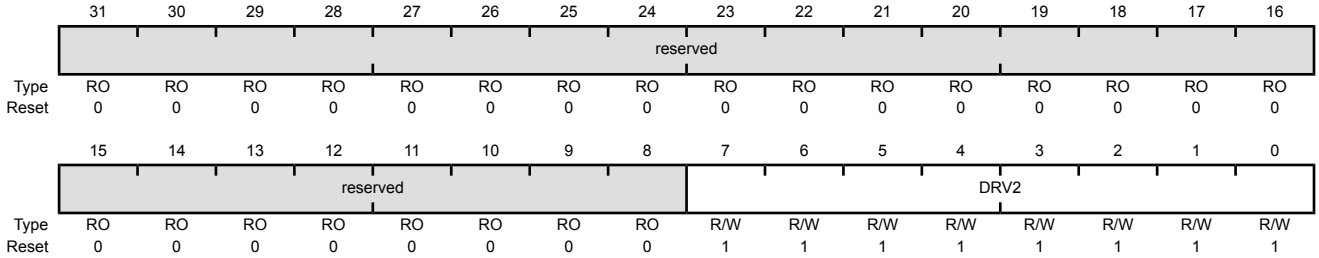
Bit/Field	Name	Type	Reset	Description
7:0	AFSEL	R/W	-	GPIO Alternate Function Select The AFSEL values are defined as follows: Value Description 0 Software control of corresponding GPIO line (GPIO mode). 1 Hardware control of corresponding GPIO line (alternate hardware function). **Note:** The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500

The **GPIODR2R** register is the 2-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing a DRV2 bit for a GPIO signal, the corresponding DRV4 bit in the **GPIODR4R** register and the DRV8 bit in the **GPIODR8R** register are automatically cleared by hardware.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x500
Type R/W, reset 0x0000.00FF



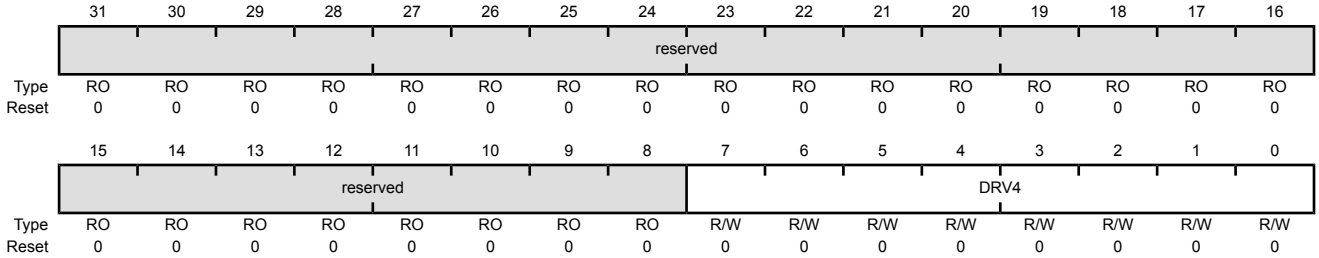
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable A write of 1 to either **GPIODR4[n]** or **GPIODR8[n]** clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write.

Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504

The **GPIODR4R** register is the 4-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing the DRV4 bit for a GPIO signal, the corresponding DRV2 bit in the **GPIODR2R** register and the DRV8 bit in the **GPIODR8R** register are automatically cleared by hardware.

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x504
Type R/W, reset 0x0000.0000



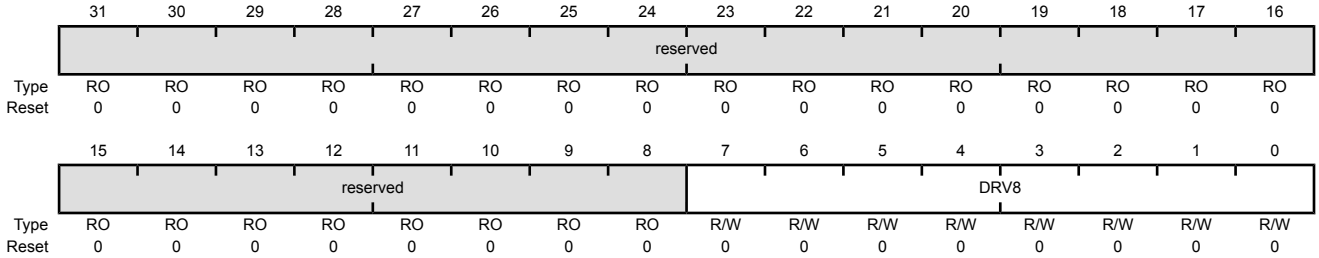
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable A write of 1 to either **GPIODR2[n]** or **GPIODR8[n]** clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write.

Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

The **GPIODR8R** register is the 8-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing the DRV8 bit for a GPIO signal, the corresponding DRV2 bit in the **GPIODR2R** register and the DRV4 bit in the **GPIODR4R** register are automatically cleared by hardware.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x508
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	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable A write of 1 to either **GPIODR2[n]** or **GPIODR4[n]** clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write.

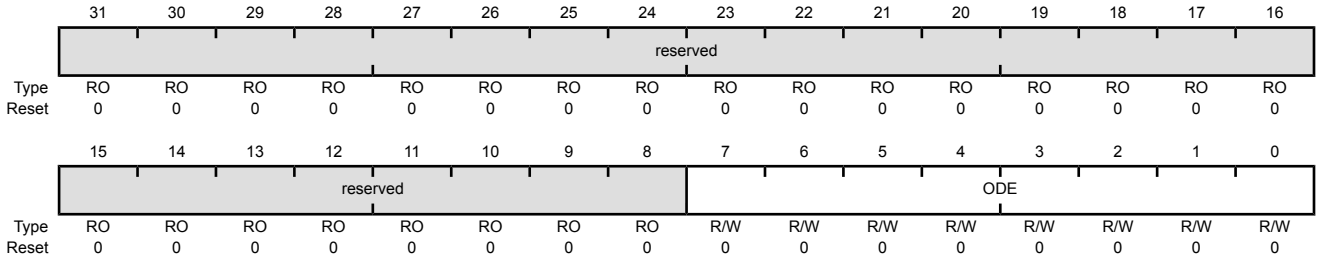
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C

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 317). Corresponding bits in the drive strength registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an open-drain 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 to 1 (see examples in “Initialization and Configuration” on page 295).

GPIO Open Drain Select (GPIOODR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x50C
Type R/W, reset 0x0000.0000



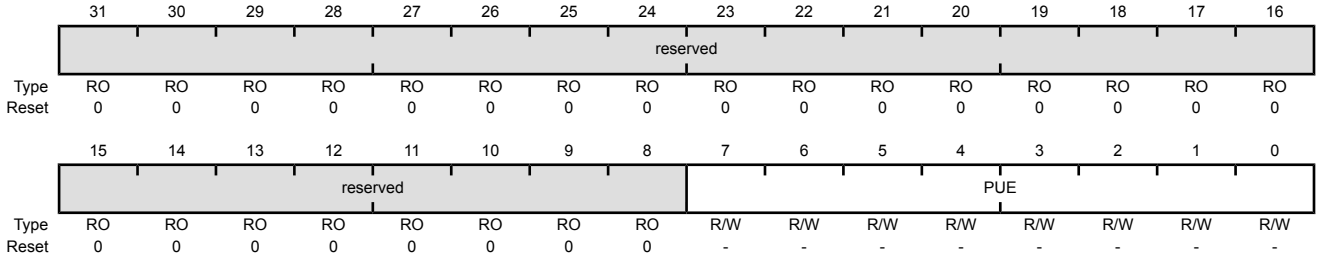
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable The ODE values are defined as follows: Value Description 0 Open drain configuration is disabled. 1 Open drain configuration is enabled.

Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510

The **GPIOPUR** register is the pull-up control register. When a bit is set to 1, it enables a weak pull-up resistor on the corresponding GPIO signal. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 315).

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x510
Type R/W, reset -



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	PUE	R/W	-	Pad Weak Pull-Up Enable

Value	Description
0	The corresponding pin's weak pull-up resistor is disabled.
1	The corresponding pin's weak pull-up resistor is enabled.

A write of 1 to **GPIOPDR[n]** clears the corresponding **GPIOPUR[n]** enables. The change is effective on the second clock cycle after the write.

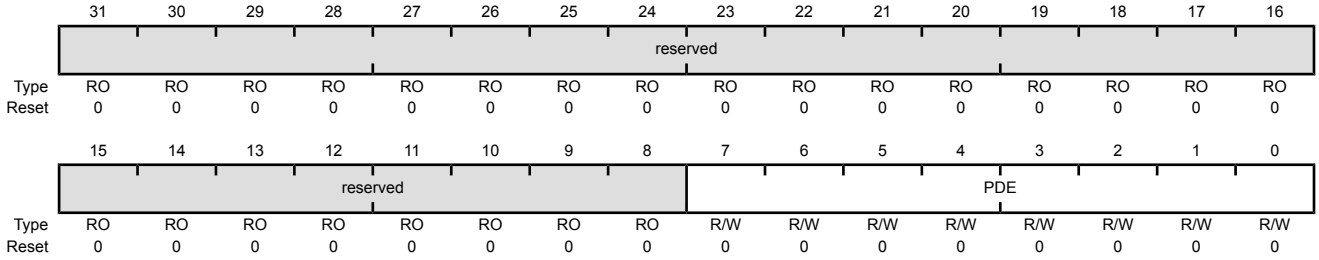
Note: The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514

The **GPIOPDR** register is the pull-down control register. When a bit is set to 1, it enables a weak pull-down resistor on the corresponding GPIO signal. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 314).

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x514
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	PDE	R/W	0x00	Pad Weak Pull-Down Enable

Value	Description
0	The corresponding pin's weak pull-down resistor is disabled.
1	The corresponding pin's weak pull-down resistor is enabled.

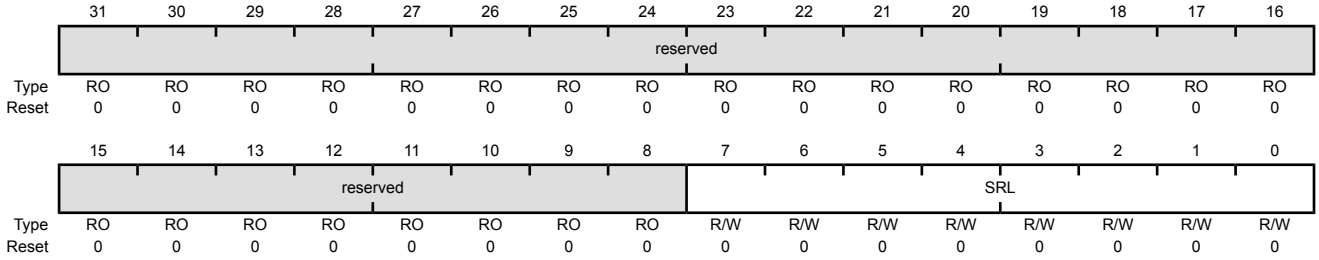
A write of 1 to **GPIOPUR[n]** clears the corresponding **GPIOPDR[n]** enables. The change is effective on the second clock cycle after the write.

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 312).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x518
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	SRL	R/W	0x00	Slew Rate Limit Enable (8-mA drive only) The SRL values are defined as follows: Value Description 0 Slew rate control disabled. 1 Slew rate control enabled.

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, with the exception of the GPIO signals used for JTAG/SWD function, all other GPIO signals 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 in a digital function (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

GPIO Digital Enable (GPIODEN)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x51C
Type R/W, 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								DEN							
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	-	-	-	-	-	-	-	-

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	DEN	R/W	-	Digital Enable The DEN values are defined as follows: Value Description 0 Digital functions disabled. 1 Digital functions enabled. **Note:** The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

Register 19: GPIO Lock (GPIOLOCK), offset 0x520

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 319). Writing 0x1ACC.E551 to the **GPIOLOCK** register will unlock 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 0x00000001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x00000000.

GPIO Lock (GPIOLOCK)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x520
Type R/W, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LOCK															
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
	LOCK															
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	1

Bit/Field	Name	Type	Reset	Description
31:0	LOCK	R/W	0x0000.0001	GPIO Lock

A write of the value 0x1ACC.E551 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. A read of this register returns the following values:

Value	Description
0x0000.0001	Locked
0x0000.0000	Unlocked

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** register are committed when a write to the **GPIOAFSEL** register is performed. If a bit in the **GPIOCR** register is a zero, the data being written to the corresponding bit in the **GPIOAFSEL** register will not be committed and will retain its previous value. If a bit in the **GPIOCR** register is a one, the data being written to the corresponding bit of the **GPIOAFSEL** register will be committed to the register and will reflect the new value.

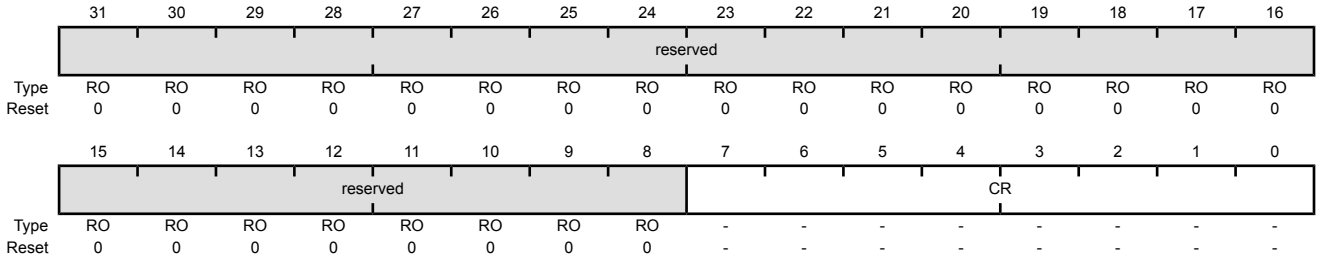
The contents of the **GPIOCR** register can only be modified if the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the **GPIOLOCK** register is locked.

Important: This register is designed to prevent accidental programming of the registers that control connectivity to the JTAG/SWD debug hardware. By initializing the bits of the **GPIOCR** register to 0 for PB7 and PC[3:0], the 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 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**register bits of these other pins.

GPIO Commit (GPIOCR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0x524
Type -, reset -



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.

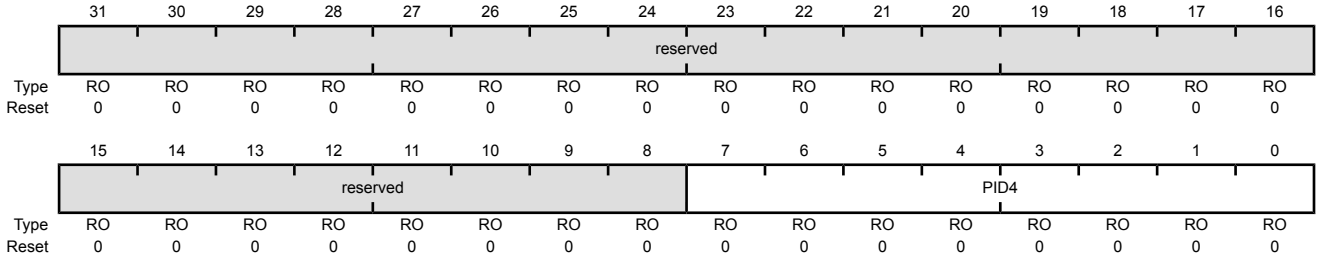
Bit/Field	Name	Type	Reset	Description
7:0	CR	-	-	GPIO Commit On a bit-wise basis, any bit set allows the corresponding **GPIOAFSEL** bit to be set to its alternate function. **Note:** The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins are currently 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 five JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as a GPIO, these five pins default 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 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFD0
Type RO, reset 0x0000.0000



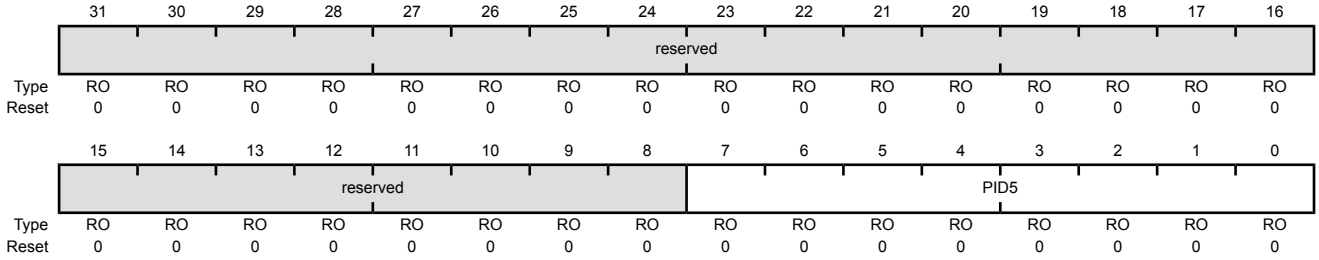
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register[7:0]

Register 22: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFD4
Type RO, reset 0x0000.0000



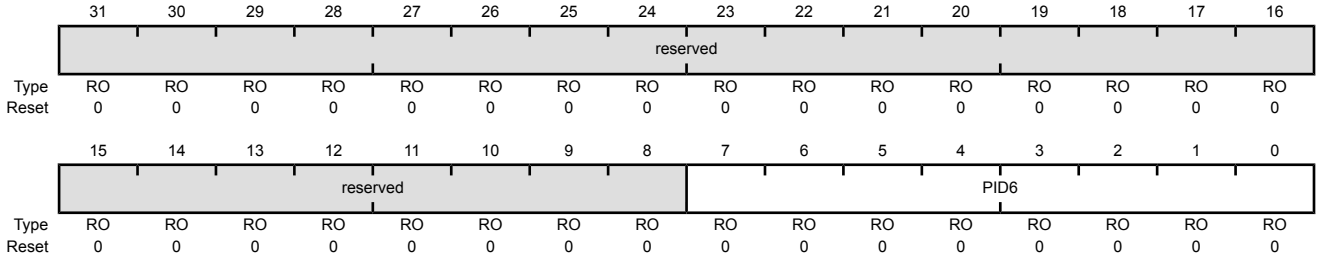
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register[15:8]

Register 23: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFD8
Type RO, reset 0x0000.0000



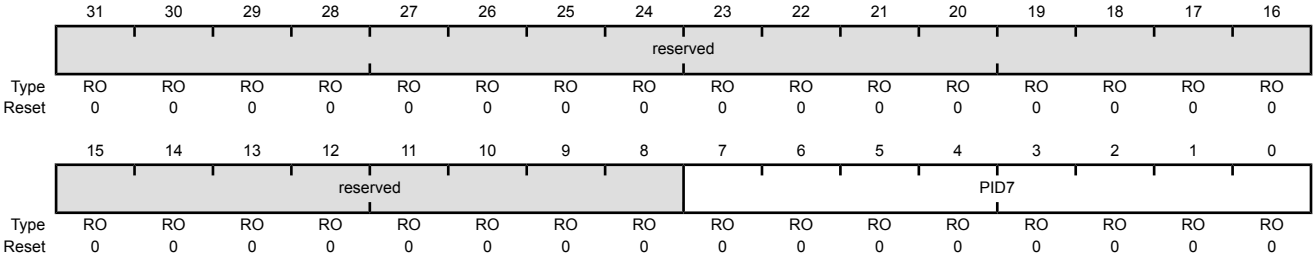
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	GPIO Peripheral ID Register[23:16]

Register 24: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFDC
Type RO, reset 0x0000.0000



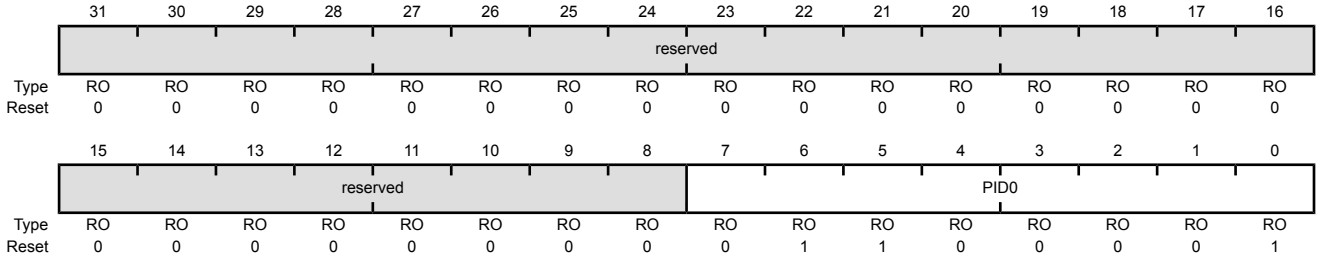
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register[31:24]

Register 25: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFE0
Type RO, reset 0x0000.0061



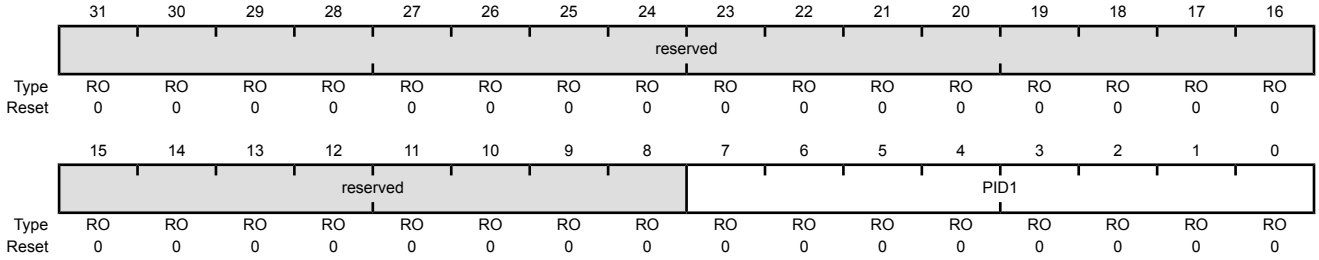
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 26: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFE4
Type RO, reset 0x0000.0000



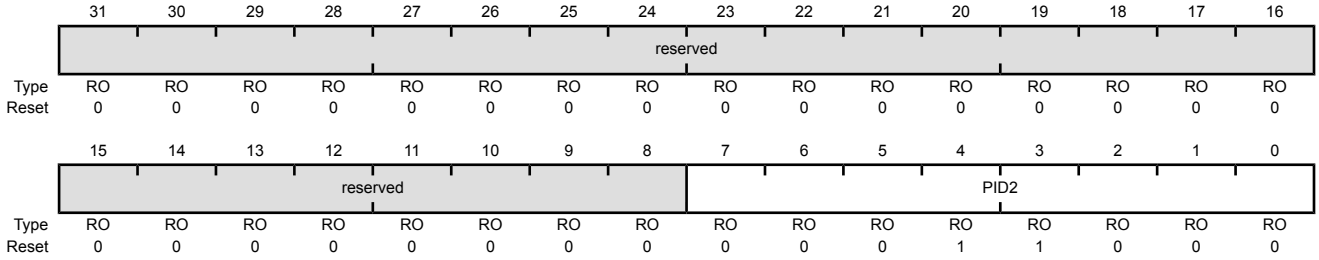
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 27: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFE8
Type RO, reset 0x0000.0018



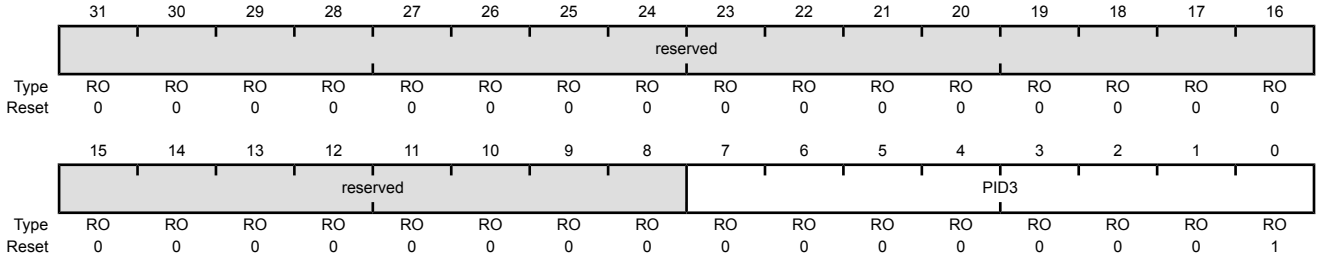
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 28: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFEC
Type RO, reset 0x0000.0001



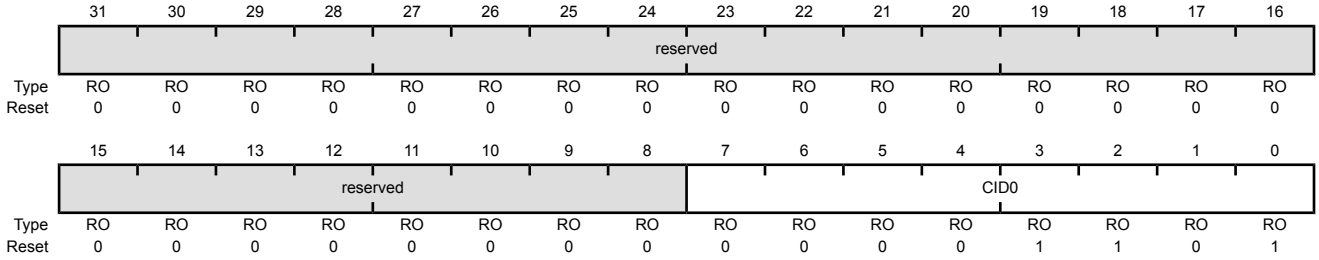
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	GPIO Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 29: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFF0
Type RO, reset 0x0000.000D



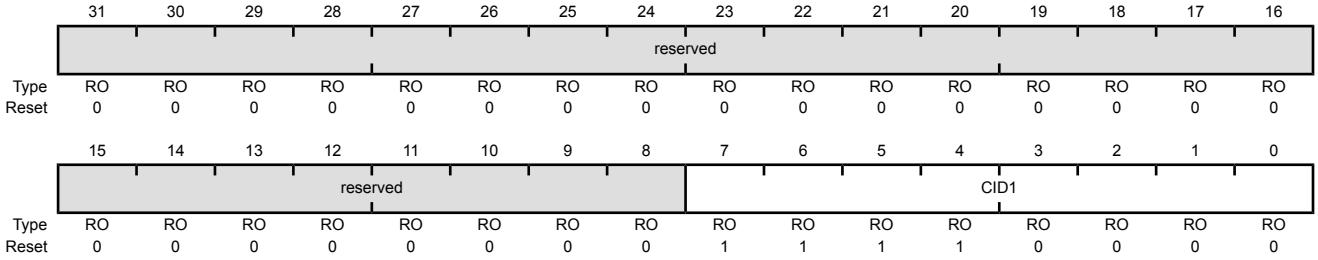
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	GPIO PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

Register 30: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFF4
Type RO, reset 0x0000.00F0



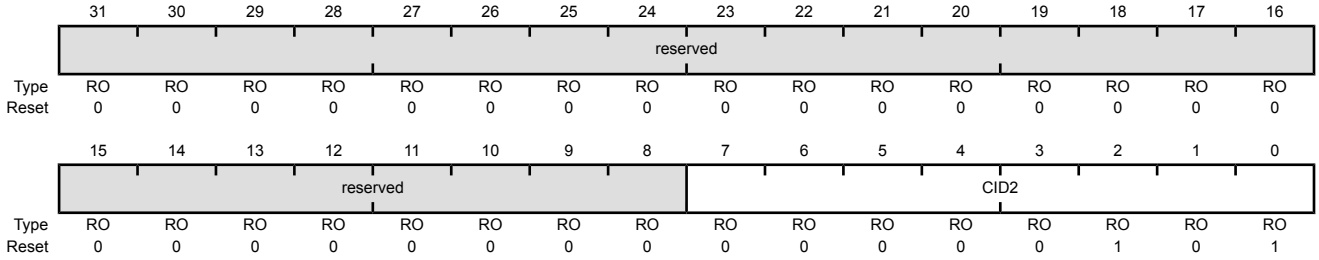
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

Register 31: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
Offset 0xFF8
Type RO, reset 0x0000.0005



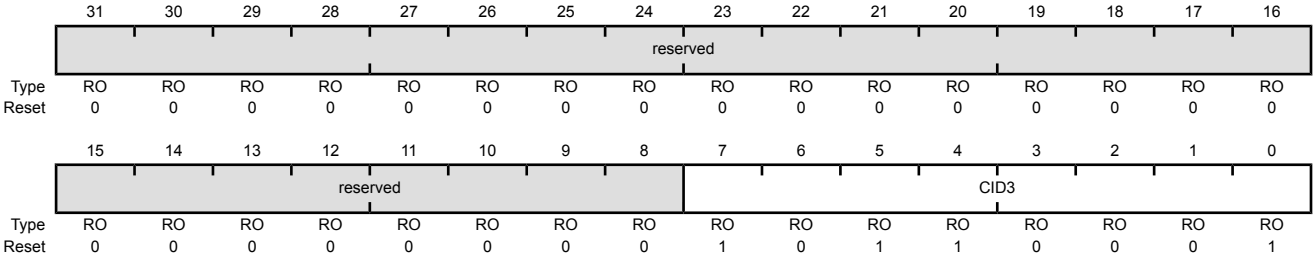
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 32: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
GPIO Port F base: 0x4002.5000
GPIO Port G base: 0x4002.6000
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	GPIO PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

9 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 (Timer0, Timer1, Timer 2, and Timer 3). Each GPTM block provides two 16-bit timers/counters (referred to as TimerA and TimerB) 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).

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 94).

The General-Purpose Timers provide the following features:

- Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers/counters. Each GPTM can be configured to operate independently:
  - As a single 32-bit timer
  - As one 32-bit Real-Time Clock (RTC) to event capture
  - For Pulse Width Modulation (PWM)
  - To trigger analog-to-digital conversions
- 32-bit Timer modes
  - Programmable one-shot timer
  - Programmable periodic timer
  - Real-Time Clock when using an external 32.768-KHz clock as the input
  - User-enabled stalling when the controller asserts CPU Halt flag during debug
  - ADC event trigger
- 16-bit Timer modes
  - General-purpose timer function with an 8-bit prescaler (for one-shot and periodic modes only)
  - Programmable one-shot timer
  - Programmable periodic timer
  - User-enabled stalling when the controller asserts CPU Halt flag during debug
  - ADC event trigger
- 16-bit Input Capture modes
  - Input edge count capture

– Input edge time capture

■ 16-bit PWM mode

– Simple PWM mode with software-programmable output inversion of the PWM signal

9.1 Block Diagram

Note: In Figure 9-1 on page 334, the specific CCP pins available depend on the Stellaris device. See Table 9-1 on page 334 for the available CCPs.

Figure 9-1. GPTM Module Block Diagram

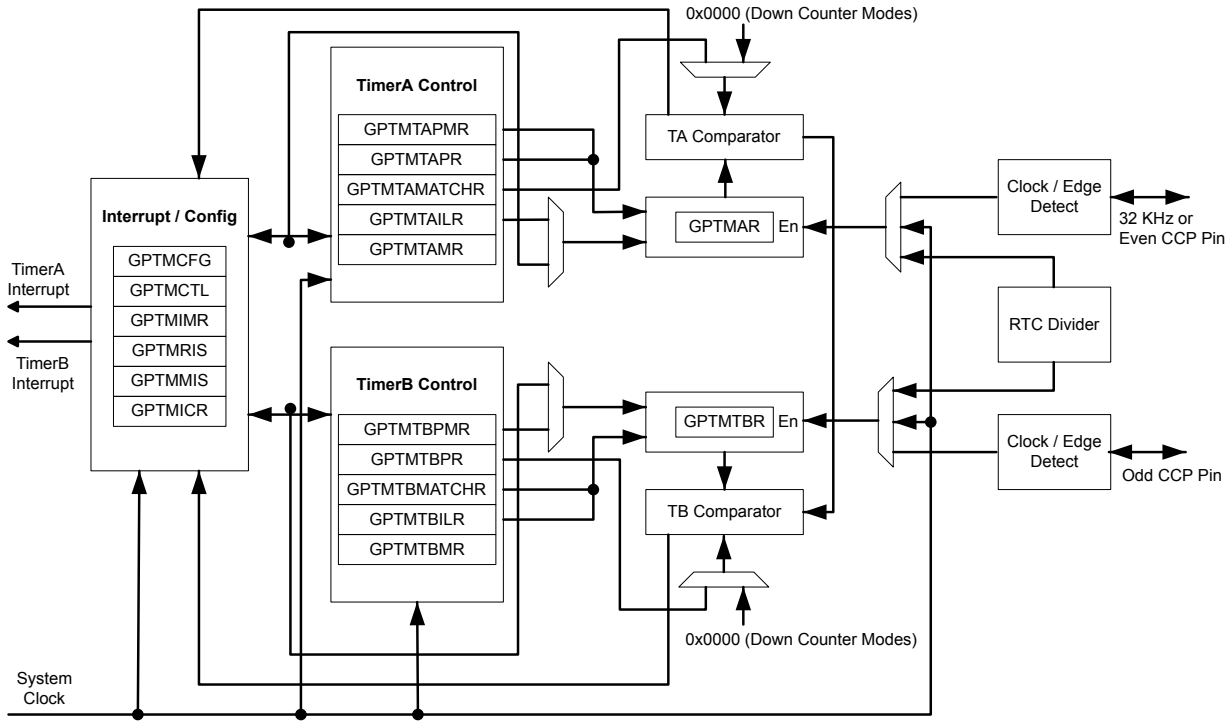


Table 9-1. Available CCP Pins

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	-	-
	TimerB	-	-

9.2 Signal Description

Table 9-2 on page 335 and Table 9-3 on page 335 list the external signals of the GP Timer module and describe 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 Assignment" lists the possible GPIO pin placements for these GP Timer signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the GP Timer function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286.

Table 9-2. General-Purpose Timers Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
CCP0	66	I/O	TTL	Capture/Compare/PWM 0.
CCP1	75	I/O	TTL	Capture/Compare/PWM 1.
CCP2	67	I/O	TTL	Capture/Compare/PWM 2.
CCP3	72	I/O	TTL	Capture/Compare/PWM 3.
CCP4	74	I/O	TTL	Capture/Compare/PWM 4.
CCP5	25	I/O	TTL	Capture/Compare/PWM 5.

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

Table 9-3. General-Purpose Timers Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
CCP0	E12	I/O	TTL	Capture/Compare/PWM 0.
CCP1	A12	I/O	TTL	Capture/Compare/PWM 1.
CCP2	D12	I/O	TTL	Capture/Compare/PWM 2.
CCP3	A11	I/O	TTL	Capture/Compare/PWM 3.
CCP4	B11	I/O	TTL	Capture/Compare/PWM 4.
CCP5	L1	I/O	TTL	Capture/Compare/PWM 5.

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

9.3 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as TimerA and TimerB), two 16-bit match registers, two prescaler match registers, and two 16-bit load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface.

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 346), the **GPTM TimerA Mode (GPTMTAMR)** register (see page 347), and the **GPTM TimerB Mode (GPTMTBMR)** register (see page 349). When in one of the 32-bit modes, the timer can only act as a 32-bit timer. However, when configured in 16-bit mode, the GPTM can have its two 16-bit timers configured in any combination of the 16-bit modes.

9.3.1 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 TimerA and TimerB are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM TimerA Interval Load (GPTMTAILR)** register (see page 360) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 361). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale**

(GPTMTAPR) register (see page 364) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 365).

9.3.2 32-Bit Timer Operating Modes

This section describes the three GPTM 32-bit timer modes (One-Shot, Periodic, and RTC) and their configuration.

The GPTM is placed into 32-bit mode by writing a 0 (One-Shot/Periodic 32-bit timer mode) or a 1 (RTC mode) to the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain GPTM registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM TimerA Interval Load (GPTMTAILR)** register [15:0], see page 360
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 361
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 368
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 369

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 read access to **GPTMTAR** returns the value:

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

9.3.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit down-counter. The selection of one-shot or periodic mode is determined by the value written to the TAMR field of the **GPTM TimerA Mode (GPTMTAMR)** register (see page 347), and there is no need to write to the **GPTM TimerB Mode (GPTMTBMR)** register.

When software writes the TAEN bit in the **GPTM Control (GPTMCTL)** register (see page 351), the timer begins counting down from its preloaded value. Once the 0x0000.0000 state is reached, the timer reloads its start value from the concatenated **GPTMTAILR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the TAEN bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the 0x000.0000 state. The GPTM sets the TATORIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 356), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 358). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register (see page 354), the GPTM also sets the TATOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 357). The ADC trigger is enabled by setting the TAOTE bit in **GPTMCTL**.

If software reloads the **GPTMTAILR** register while the counter is running, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the TASTALL 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.

9.3.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time, the counter is loaded with a value of 0x0000.0001. All subsequent load values must be written to the **GPTM TimerA Match (GPTMTAMATCHR)** register (see page 362) by the controller.

The input clock on an even 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 32-bit counter.

When software writes the TAEN bit inthe **GPTMCTL** register, the counter starts counting up from its preloaded value of 0x0000.0001. When the current count value matches the preloaded value in the **GPTMTAMATCHR** register, it rolls over to a value of 0x0000.0000 and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When a match occurs, the GPTM asserts the RTCRIS bit in **GPTMRIS**. 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**.

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**.

9.3.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 346). This section describes each of the GPTM 16-bit modes of operation. TimerA and TimerB have identical modes, so a single description is given using an **n** to reference both.

9.3.3.1 16-Bit One-Shot/Periodic Timer Mode

In 16-bit one-shot and periodic timer modes, the timer is configured as a 16-bit down-counter with an optional 8-bit prescaler that effectively extends the counting range of the timer to 24 bits. The selection of one-shot or periodic mode is determined by the value written to the TnMR field of the **GPTMTnMR** register. The optional prescaler is loaded into the **GPTM Timern Prescale (GPTMTnPR)** register.

When software writes the TnEN bit in the **GPTMCTL** register, the timer begins counting down from its preloaded value. Once the 0x0000 state is reached, the timer reloads its start value from **GPTMTnILR** and **GPTMTnPR** on the next cycle. 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, it continues counting.

In addition to reloading the count value, the timer generates interrupts and triggers when it reaches the 0x0000 state. The GPTM sets the TnTORIS bit in the **GPTMRIS** register, and holds it until it is cleared by writing the **GPTMICR** register. If the time-out interrupt is enabled in **GPTMIMR**, the GPTM also sets the TnTOMIS bit in **GPTMISR** and generates a controller interrupt. The ADC trigger is enabled by setting the TnOTE bit in the **GPTMCTL** register.

If software reloads the **GPTMTAILR** register while the counter is running, 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 example shows a variety of configurations for a 16-bit free running timer while using the prescaler. All values assume a 50-MHz clock with Tc=20 ns (clock period).

Table 9-4. 16-Bit Timer With Prescaler Configurations

Prescale	#Clock (T c)[a]	Max Time	Units
00000000	1	1.3107	mS
00000001	2	2.6214	mS
00000010	3	3.9322	mS
----------	--	--	--
11111101	254	332.9229	mS
11111110	255	334.2336	mS
11111111	256	335.5443	mS

a. Tc is the clock period.

9.3.3.2 16-Bit 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.

Note: The prescaler is not available in 16-Bit Input Edge Count mode.

In Edge Count mode, the timer is configured as a down-counter 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 set to 0. The type of edge that the timer counts is determined by the TnEVENT fields of the **GPTMCTL** register. During initialization, the **GPTM Timern Match (GPTMTnMATCHR)** register is configured so that the difference between the value in the **GPTMTnILR** register and the **GPTMTnMATCHR** register equals the number of edge events that must be counted.

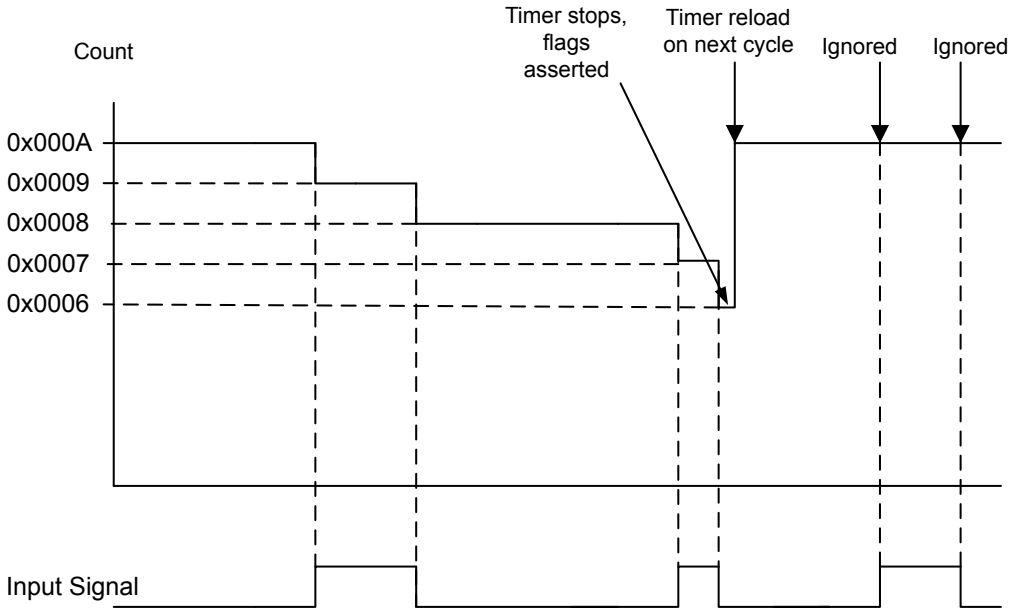
When software writes the TnEN bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the CnMRIS bit in the **GPTMRIS** register (and the CnMMIS bit, if the interrupt is not masked).

The counter is then reloaded using the value in **GPTMTnILR**, and stopped since 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 9-2 on page 339 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 since the timer automatically clears the TnEN bit after the current count matches the value in the **GPTMTnMATCHR** register.

Figure 9-2. 16-Bit Input Edge Count Mode Example



9.3.3.3 16-Bit 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.

Note: The prescaler is not available in 16-Bit Input Edge Time mode.

In Edge Time mode, the timer is configured as a free-running down-counter initialized to the value loaded in the **GPTMTnILR** register (or 0xFFFF at reset). 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.

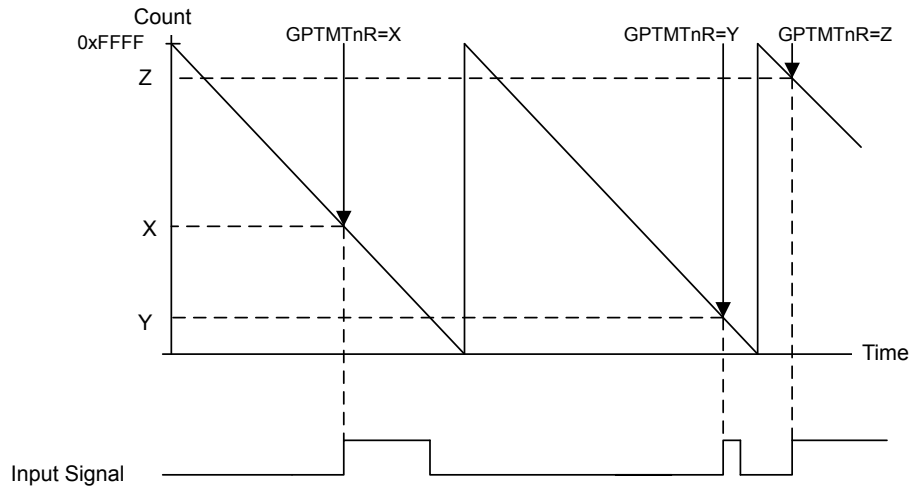
When software writes the TnEN bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current Tn counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the CnERIS bit (and the CnEMIS bit, if the interrupt is not masked).

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 0x0000 state, it is reloaded with the value from the **GPTMTnILR** register.

Figure 9-3 on page 340 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 **GPTMTnR**).

Figure 9-3. 16-Bit Input Edge Time Mode Example



9.3.3.4 16-Bit PWM Mode

Note: The prescaler is not available in 16-Bit PWM mode.

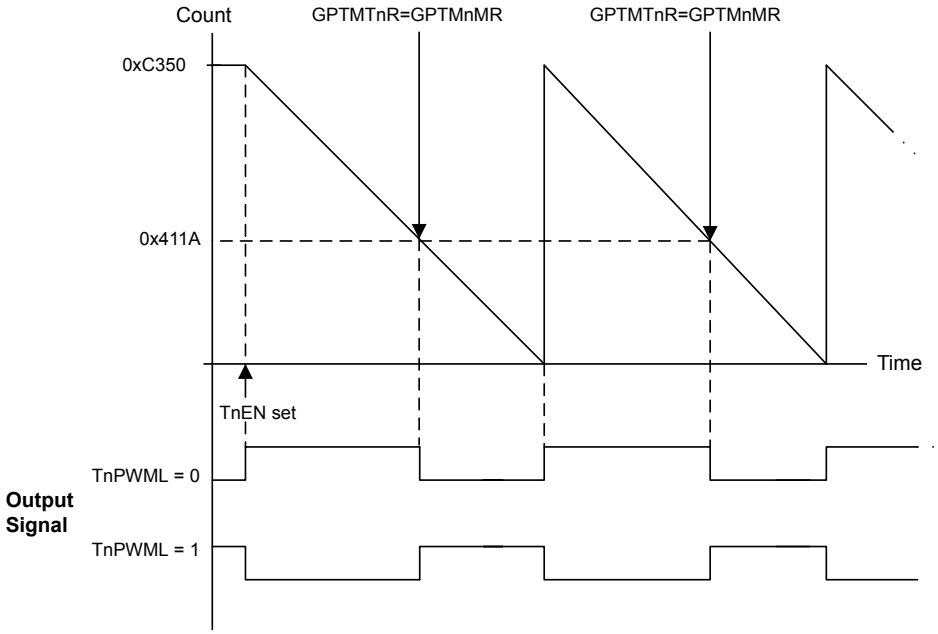
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the **GPTMTnMR** register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.

When software writes the TnEN bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0000 state. On the next counter cycle, the counter reloads its start value from **GPTMTnILR** 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 **GPTM Timern Match Register (GPTMTnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the TnPWML bit in the **GPTMCTL** register.

Figure 9-4 on page 341 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 **GPTMTnIRL**=0xC350 and the match value is **GPTMTnMATCHR**=0x411A.

Figure 9-4. 16-Bit PWM Mode Example



9.4 Initialization and Configuration

To use the general-purpose timers, the peripheral clock must be enabled by setting the TIMER0, TIMER1, TIMER2, and TIMER3 bits in the **RCGC1** register.

This section shows module initialization and configuration examples for each of the supported timer modes.

9.4.1 32-Bit One-Shot/Periodic Timer Mode

The GPTM is configured for 32-bit One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the TAEN bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0.
3. Set the TAMR field in the **GPTM TimerA Mode Register (GPTMTAMR)**:
   a. Write a value of 0x1 for One-Shot mode.
   b. Write a value of 0x2 for Periodic mode.
4. Load the start value into the **GPTM TimerA Interval Load Register (GPTMTAILR)**.
5. If interrupts are required, set the TATOIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the TAEN bit in the **GPTMCTL** register to enable the timer and start counting.

7. Poll the TATORIS 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 TATOCINT bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 7 on page 342. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

9.4.2 32-Bit 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 0x1.
3. Write the desired match value to the **GPTM TimerA Match Register (GPTMTAMATCHR)**.
4. Set/clear the RTCEN bit in the **GPTM Control Register (GPTMCTL)** as desired.
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 **GPTMTAMATCHR** 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.

9.4.3 16-Bit One-Shot/Periodic Timer Mode

A timer is configured for 16-bit One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x4.
3. Set the TnMR field in the **GPTM Timer Mode (GPTMTnMR)** register:
   a. Write a value of 0x1 for One-Shot mode.
   b. Write a value of 0x2 for Periodic mode.
4. If a prescaler is to be used, write the prescale value to the **GPTM Timern Prescale Register (GPTMTnPR)**.
5. Load the start value into the **GPTM Timer Interval Load Register (GPTMTnILR)**.
6. If interrupts are required, set the TnTOIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the TnEN bit in the **GPTM Control Register (GPTMCTL)** to enable the timer and start counting.
8. Poll the TnTORIS 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 TnTOCINT bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 8 on page 342. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

9.4.4 16-Bit 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 0x4.
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. Load the timer start value into the **GPTM Timern Interval Load (GPTMTnILR)** register.
6. Load the desired event count into the **GPTM Timern Match (GPTMTnMATCHR)** register.
7. If interrupts are required, set the CnMIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the TnEN bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
9. 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.

In Input Edge Count Mode, the timer stops after the desired 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 343 through step 9 on page 343.

9.4.5 16-Bit 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 0x4.
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 Timern 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 Timern (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.

9.4.6 16-Bit 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 0x4.
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 Timern Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timern Match (GPTMTnMATCHR)** register with the desired 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.

9.5 Register Map

Table 9-5 on page 344 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- Timer0: 0x4003.0000
- Timer1: 0x4003.1000
- Timer2: 0x4003.2000
- Timer3: 0x4003.3000

Note that the Timer module clock must be enabled before the registers can be programmed (see page 219). There must be a delay of 3 system clocks after the Timer module clock is enabled before any Timer module registers are accessed.

Table 9-5. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	346
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA Mode	347
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB Mode	349

Table 9-5. Timers Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	351
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	354
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	356
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	357
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	358
0x028	GPTMTAILR	R/W	0xFFFF.FFFF	GPTM TimerA Interval Load	360
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB Interval Load	361
0x030	GPTMTAMATCHR	R/W	0xFFFF.FFFF	GPTM TimerA Match	362
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB Match	363
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA Prescale	364
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB Prescale	365
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match	366
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match	367
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM TimerA	368
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	369

9.6 Register Descriptions

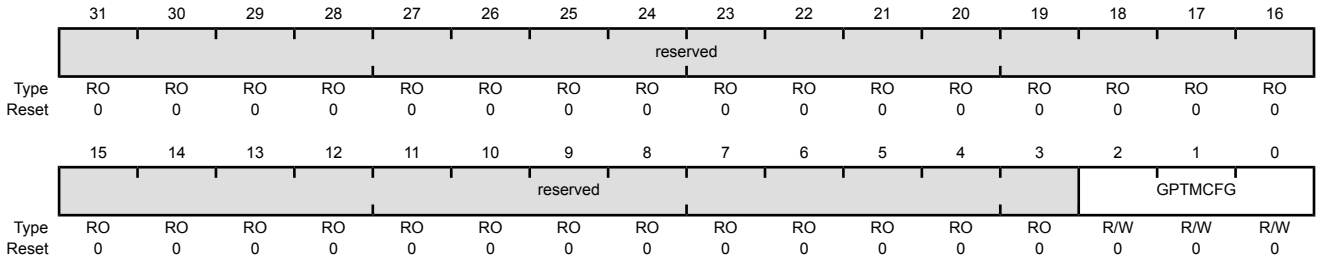
The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

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.

GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x000
Type R/W, reset 0x0000.0000



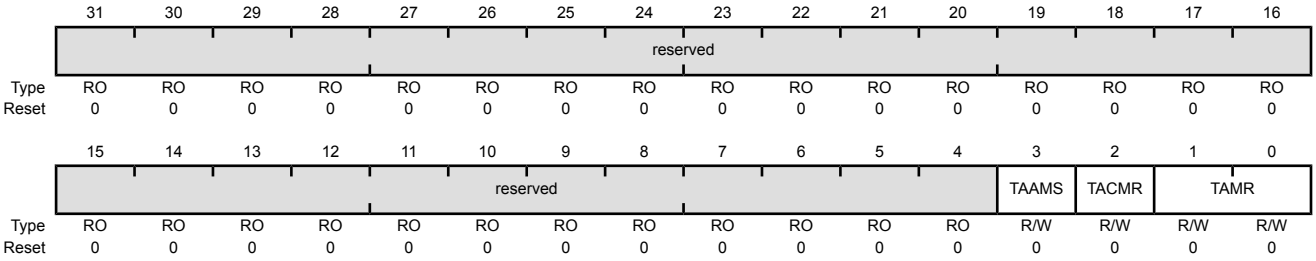
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	GPTMCFG	R/W	0x0	GPTM Configuration The GPTMCFG values are defined as follows: Value Description 0x0 32-bit timer configuration. 0x1 32-bit real-time clock (RTC) counter configuration. 0x2 Reserved 0x3 Reserved 0x4-0x7 16-bit timer configuration, function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**.

Register 2: GPTM TimerA Mode (GPTMTAMR), offset 0x004

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the TAAMS bit to 0x1, the TACMR bit to 0x0, and the TAMR field to 0x2.

GPTM TimerA Mode (GPTMTAMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x004
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	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.
3	TAAMS	R/W	0	GPTM TimerA Alternate Mode Select The TAAMS values are defined as follows: Value Description 0 Capture mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the TACMR bit and set the TAMR field to 0x2.
2	TACMR	R/W	0	GPTM TimerA Capture Mode The TACMR values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode

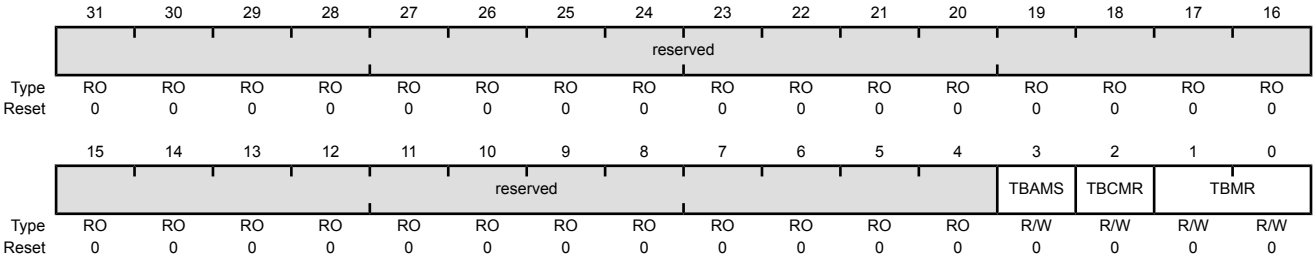
Bit/Field	Name	Type	Reset	Description
1:0	TAMR	R/W	0x0	GPTM TimerA Mode The TAMR values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register (16-or 32-bit). In 16-bit timer configuration, TAMR controls the 16-bit timer modes for TimerA. In 32-bit timer configuration, this register controls the mode and the contents of **GPTMTBMR** are ignored.

Register 3: GPTM TimerB Mode (GPTMTBMR), offset 0x008

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the TBAMS bit to 0x1, the TBCMR bit to 0x0, and the TBMR field to 0x2.

GPTM TimerB Mode (GPTMTBMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x008
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	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.
3	TBAMS	R/W	0	GPTM TimerB Alternate Mode Select The TBAMS values are defined as follows: Value Description 0 Capture mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the TBCMR bit and set the TBMR field to 0x2.
2	TBCMR	R/W	0	GPTM TimerB Capture Mode The TBCMR values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode

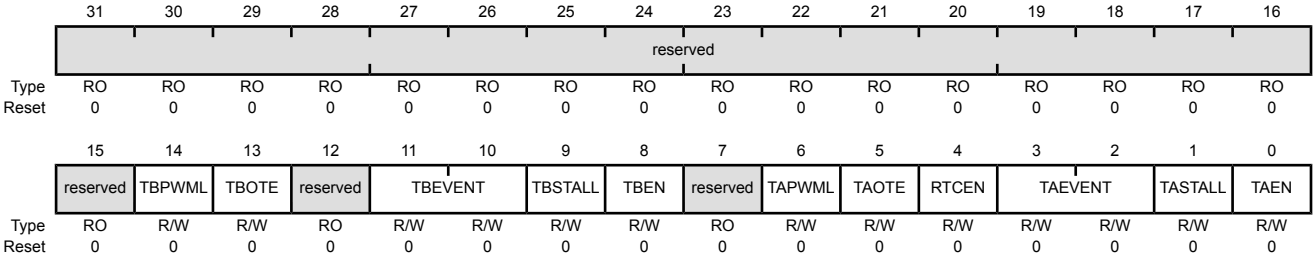
Bit/Field	Name	Type	Reset	Description
1:0	TBMR	R/W	0x0	GPTM TimerB Mode The TBMR values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register. In 16-bit timer configuration, these bits control the 16-bit timer modes for TimerB. In 32-bit timer configuration, this register's contents are ignored and **GPTMTAMR** is used.

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.

GPTM Control (GPTMCTL)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x00C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:15	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.
14	TBPWML	R/W	0	GPTM TimerB PWM Output Level The TBPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
13	TBOTE	R/W	0	GPTM TimerB Output Trigger Enable The TBOTE values are defined as follows: Value Description 0 The output TimerB ADC trigger is disabled. 1 The output TimerB ADC trigger is enabled. 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 409).
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 TimerB Event Mode The TBEVENT values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges
9	TBSTALL	R/W	0	GPTM Timer B Stall Enable The TBSTALL values are defined as follows: Value Description 0 Timer B continues counting while the processor is halted by the debugger. 1 Timer B freezes counting while the processor is halted by the debugger. If the processor is executing normally, the TBSTALL bit is ignored.
8	TBEN	R/W	0	GPTM TimerB Enable The TBEN values are defined as follows: Value Description 0 TimerB is disabled. 1 TimerB 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 TimerA PWM Output Level The TAPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
5	TAOTE	R/W	0	GPTM TimerA Output Trigger Enable The TAOTE values are defined as follows: Value Description 0 The output TimerA ADC trigger is disabled. 1 The output TimerA ADC trigger is enabled. 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 409).

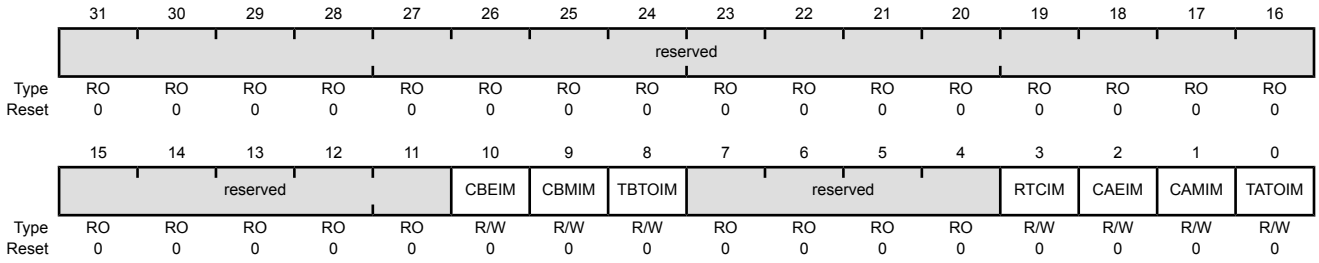
Bit/Field	Name	Type	Reset	Description
4	RTCEN	R/W	0	GPTM RTC Enable The RTCEN values are defined as follows: Value Description 0 RTC counting is disabled. 1 RTC counting is enabled.
3:2	TAEVENT	R/W	0x0	GPTM TimerA Event Mode The TAEVENT values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges
1	TASTALL	R/W	0	GPTM Timer A Stall Enable The TASTALL values are defined as follows: Value Description 0 Timer A continues counting while the processor is halted by the debugger. 1 Timer A freezes counting while the processor is halted by the debugger. If the processor is executing normally, the TASTALL bit is ignored.
0	TAEN	R/W	0	GPTM TimerA Enable The TAEN values are defined as follows: Value Description 0 TimerA is disabled. 1 TimerA 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. Writing a 1 enables the interrupt, while writing a 0 disables it.

GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBEIM	R/W	0	GPTM CaptureB Event Interrupt Mask The CBEIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
9	CBMIM	R/W	0	GPTM CaptureB Match Interrupt Mask The CBMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
8	TBTOIM	R/W	0	GPTM TimerB Time-Out Interrupt Mask The TBTOIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

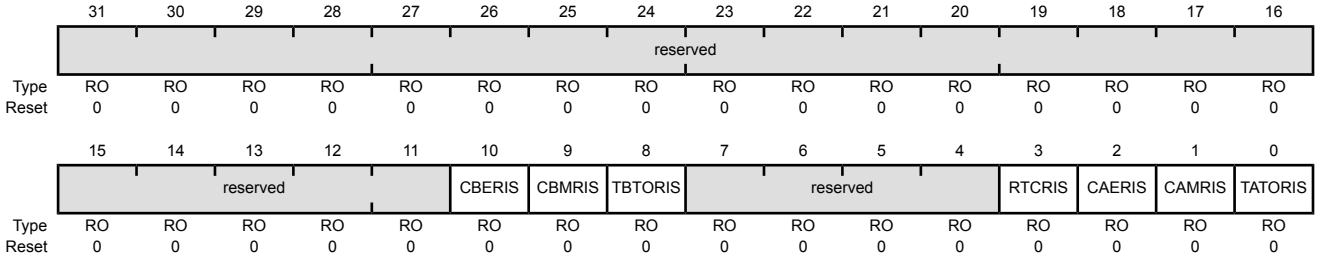
Bit/Field	Name	Type	Reset	Description
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask The RTCIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	R/W	0	GPTM CaptureA Event Interrupt Mask The CAEIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
1	CAMIM	R/W	0	GPTM CaptureA Match Interrupt Mask The CAMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
0	TATOIM	R/W	0	GPTM TimerA Time-Out Interrupt Mask The TATOIM values are defined as follows: Value Description 0 Interrupt is disabled. 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)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x01C
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBERIS	RO	0	GPTM CaptureB Event Raw Interrupt This is the CaptureB Event interrupt status prior to masking.
9	CBMRIS	RO	0	GPTM CaptureB Match Raw Interrupt This is the CaptureB Match interrupt status prior to masking.
8	TBTORIS	RO	0	GPTM TimerB Time-Out Raw Interrupt This is the TimerB time-out interrupt status prior to masking.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt This is the RTC Event interrupt status prior to masking.
2	CAERIS	RO	0	GPTM CaptureA Event Raw Interrupt This is the CaptureA Event interrupt status prior to masking.
1	CAMRIS	RO	0	GPTM CaptureA Match Raw Interrupt This is the CaptureA Match interrupt status prior to masking.
0	TATORIS	RO	0	GPTM TimerA Time-Out Raw Interrupt This the TimerA time-out interrupt status prior to masking.

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)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x020
Type RO, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved					CBEMIS	CBMMIS	TBTOMIS	reserved				RTCMIS	CAEMIS	CAMMIS	TATOMIS
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

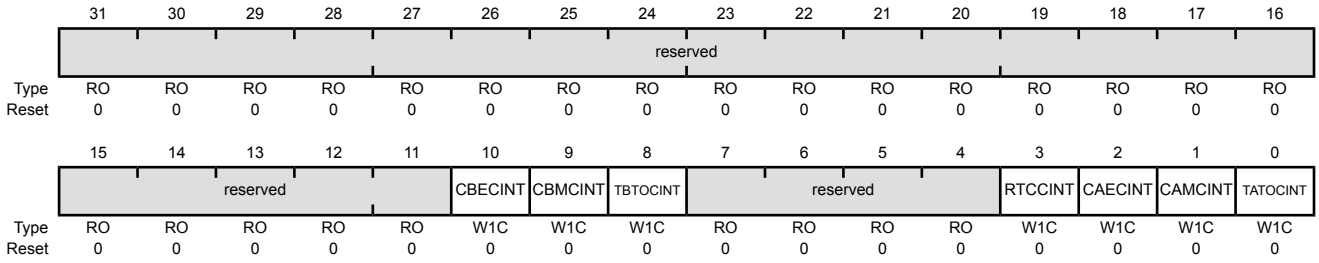
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBEMIS	RO	0	GPTM CaptureB Event Masked Interrupt This is the CaptureB event interrupt status after masking.
9	CBMMIS	RO	0	GPTM CaptureB Match Masked Interrupt This is the CaptureB match interrupt status after masking.
8	TBTOMIS	RO	0	GPTM TimerB Time-Out Masked Interrupt This is the TimerB time-out interrupt status after masking.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt This is the RTC event interrupt status after masking.
2	CAEMIS	RO	0	GPTM CaptureA Event Masked Interrupt This is the CaptureA event interrupt status after masking.
1	CAMMIS	RO	0	GPTM CaptureA Match Masked Interrupt This is the CaptureA match interrupt status after masking.
0	TATOMIS	RO	0	GPTM TimerA Time-Out Masked Interrupt This is the TimerA time-out interrupt status after masking.

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)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x024
Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBECINT	W1C	0	GPTM CaptureB Event Interrupt Clear The CBECINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
9	CBMCINT	W1C	0	GPTM CaptureB Match Interrupt Clear The CBMCINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
8	TBTOCINT	W1C	0	GPTM TimerB Time-Out Interrupt Clear The TBTOCINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

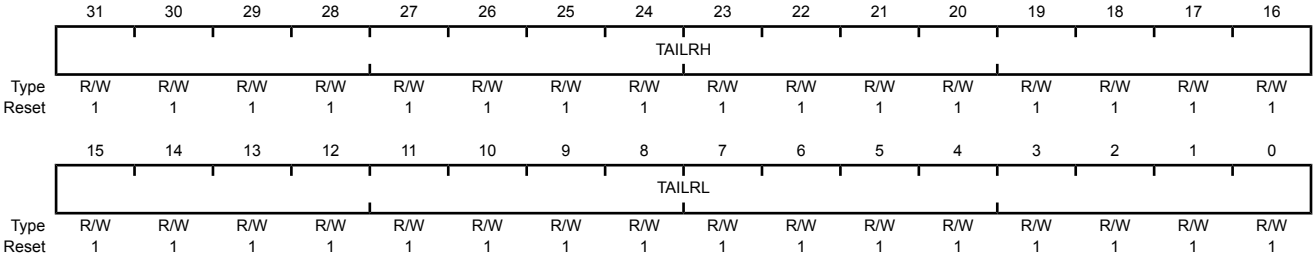
Bit/Field	Name	Type	Reset	Description
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear The RTCCINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
2	CAECINT	W1C	0	GPTM CaptureA Event Interrupt Clear The CAECINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
1	CAMCINT	W1C	0	GPTM CaptureA Match Interrupt Clear The CAMCINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
0	TATOCINT	W1C	0	GPTM TimerA Time-Out Interrupt Clear The TATOCINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.

Register 9: GPTM TimerA Interval Load (GPTMTAILR), offset 0x028

This register is used to load the starting count value into the timer. When 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 TimerB Interval Load (GPTMTBILR)** register). In 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

GPTM TimerA Interval Load (GPTMTAILR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x028
Type R/W, reset 0xFFFF.FFFF



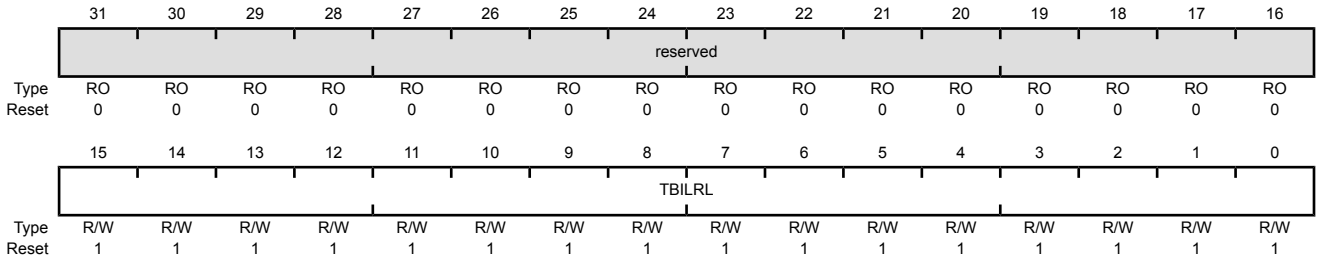
Bit/Field	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF	GPTM TimerA Interval Load Register High When configured for 32-bit mode via the **GPTMCFG** register, the **GPTM TimerB Interval Load (GPTMTBILR)** register loads this value on a write. A read returns the current value of **GPTMTBILR**. In 16-bit mode, this field reads as 0 and does not have an effect on the state of **GPTMTBILR**.
15:0	TAILRL	R/W	0xFFFF	GPTM TimerA Interval Load Register Low For both 16- and 32-bit modes, writing this field loads the counter for TimerA. A read returns the current value of **GPTMTAILR**.

Register 10: GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C

This register is used to load the starting count value into TimerB. When the GPTM is configured to a 32-bit mode, **GPTMTBILR** returns the current value of TimerB and ignores writes.

GPTM TimerB Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x02C
Type R/W, reset 0x0000.FFFF



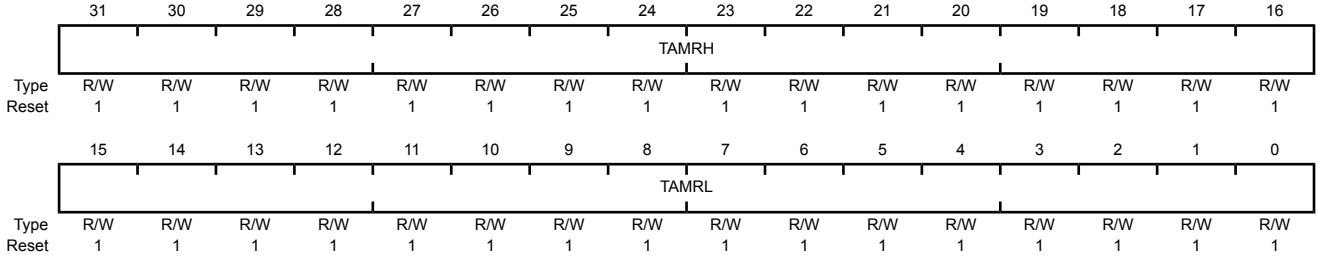
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBILRL	R/W	0xFFFF	GPTM TimerB Interval Load Register When the GPTM is not configured as a 32-bit timer, a write to this field updates **GPTMTBILR**. In 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

Register 11: GPTM TimerA Match (GPTMTAMATCHR), offset 0x030

This register is used in 32-bit Real-Time Clock mode and 16-bit PWM and Input Edge Count modes.

GPTM TimerA Match (GPTMTAMATCHR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x030
Type R/W, reset 0xFFFF.FFFF



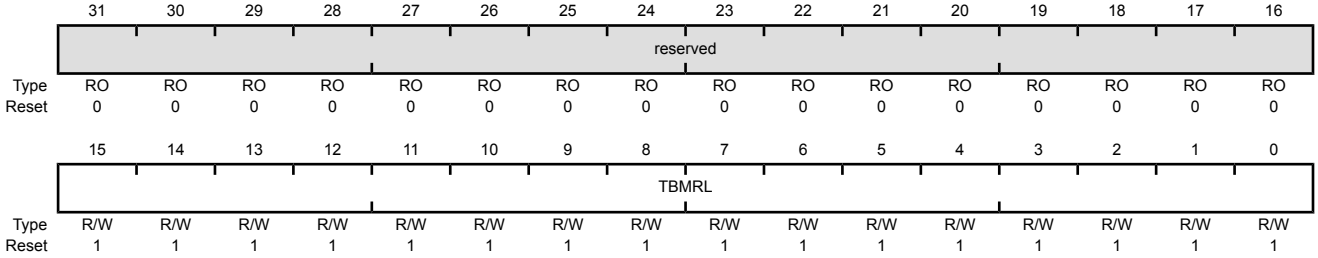
Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF	GPTM TimerA Match Register High When configured for 32-bit Real-Time Clock (RTC) mode via the **GPTMCFG** register, this value is compared to the upper half of **GPTMTAR**, to determine match events. In 16-bit mode, this field reads as 0 and does not have an effect on the state of **GPTMTBMATCHR**.
15:0	TAMRL	R/W	0xFFFF	GPTM TimerA Match Register Low When configured for 32-bit Real-Time Clock (RTC) mode via the **GPTMCFG** register, this value is compared to the lower half of **GPTMTAR**, to determine match events. When configured for PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal. When configured for Edge Count mode, this value 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.

Register 12: GPTM TimerB Match (GPTMTBMATCHR), offset 0x034

This register is used in 16-bit PWM and Input Edge Count modes.

GPTM TimerB Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x034
Type R/W, reset 0x0000.FFFF



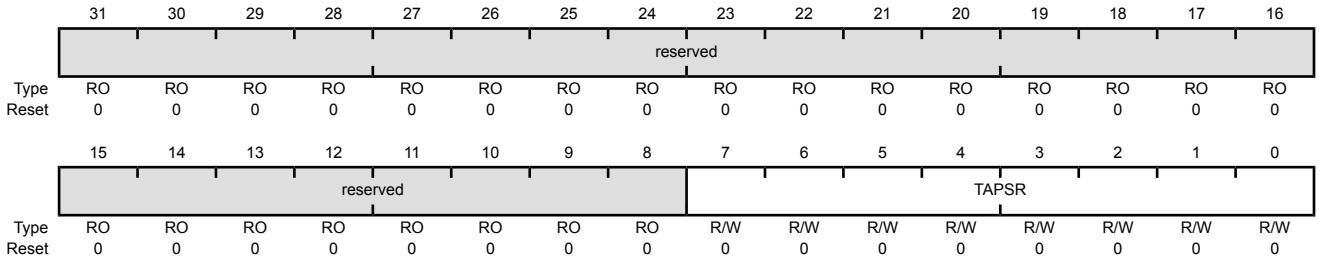
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBMRL	R/W	0xFFFF	GPTM TimerB Match Register Low When configured for PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal. When configured for Edge Count mode, this value 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.

Register 13: GPTM TimerA Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the 16-bit timers when operating in one-shot or periodic mode.

GPTM TimerA Prescale (GPTMTAPR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x038
Type R/W, reset 0x0000.0000



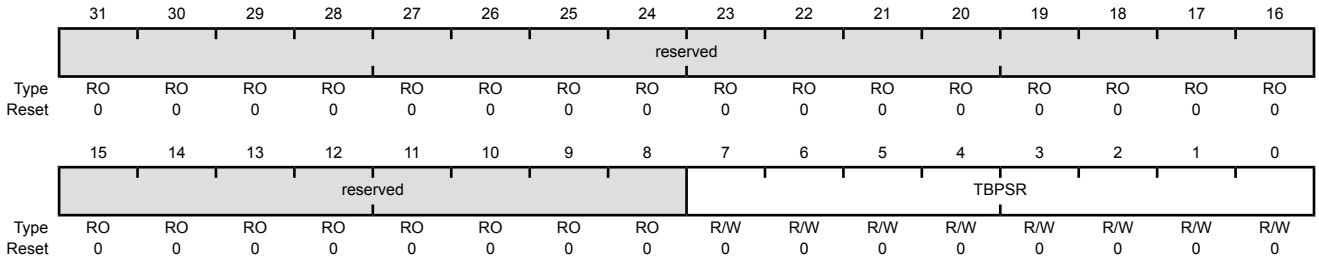
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0x00	GPTM TimerA Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 9-4 on page 338 for more details and an example.

Register 14: GPTM TimerB Prescale (GPTMTBPR), offset 0x03C

This register allows software to extend the range of the 16-bit timers when operating in one-shot or periodic mode.

GPTM TimerB Prescale (GPTMTBPR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x03C
Type R/W, reset 0x0000.0000



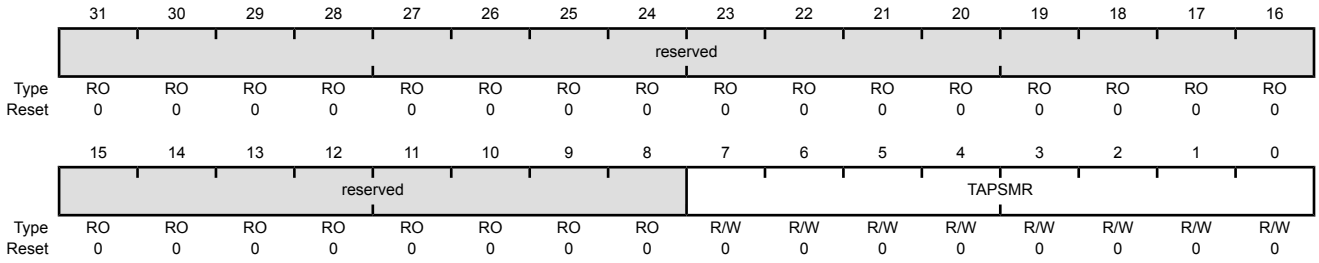
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0x00	GPTM TimerB Prescale The register loads this value on a write. A read returns the current value of this register. Refer to Table 9-4 on page 338 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)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x040
Type R/W, reset 0x0000.0000



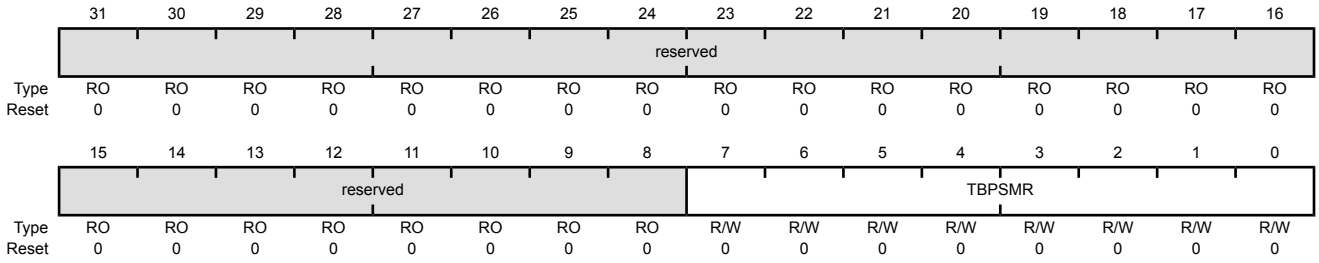
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSMR	R/W	0x00	GPTM TimerA Prescale Match This 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)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x044
Type R/W, reset 0x0000.0000



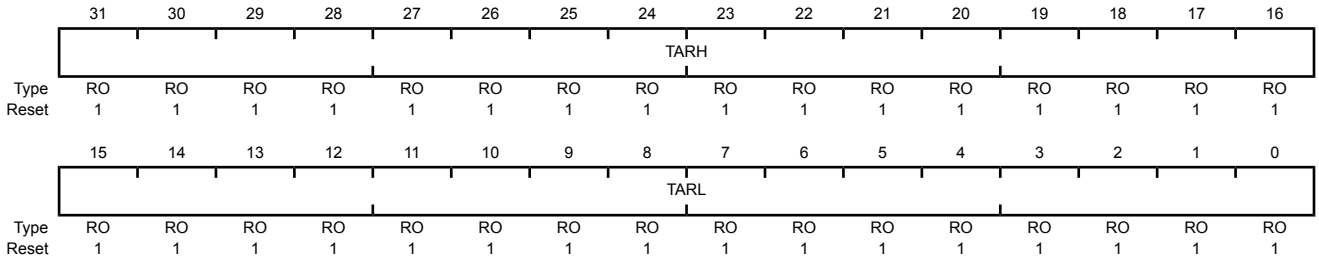
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSMR	R/W	0x00	GPTM TimerB Prescale Match This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler.

Register 17: GPTM TimerA (GPTMTAR), offset 0x048

This register shows the current value of the TimerA counter in all cases except for Input Edge Count mode. When in this mode, this register contains the number of edges that have occurred.

GPTM TimerA (GPTMTAR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x048
Type RO, reset 0xFFFF.FFFF



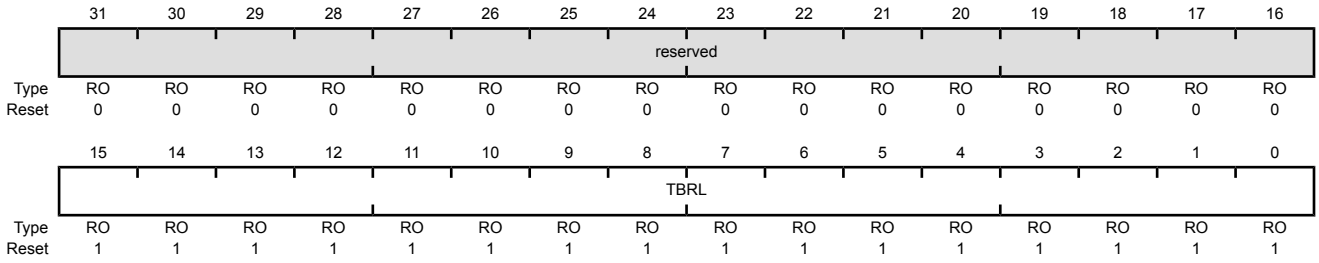
Bit/Field	Name	Type	Reset	Description
31:16	TARH	RO	0xFFFF	GPTM TimerA Register High If the **GPTMCFG** is in a 32-bit mode, TimerB value is read. If the **GPTMCFG** is in a 16-bit mode, this is read as zero.
15:0	TARL	RO	0xFFFF	GPTM TimerA Register Low A read returns the current value of the **GPTM TimerA Count Register**, except in Input Edge-Count mode, when it returns the number of edges that have occurred.

Register 18: GPTM TimerB (GPTMTBR), offset 0x04C

This register shows the current value of the TimerB counter in all cases except for Input Edge Count mode. When in this mode, this register contains the number of edges that have occurred.

GPTM TimerB (GPTMTBR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x04C
Type RO, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBRL	RO	0xFFFF	GPTM TimerB A read returns the current value of the **GPTM TimerB Count Register**, except in Input Edge-Count mode, when it returns the number of edges that have occurred.

10 Watchdog Timer

A watchdog timer can generate nonmaskable interrupts (NMIs) 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.

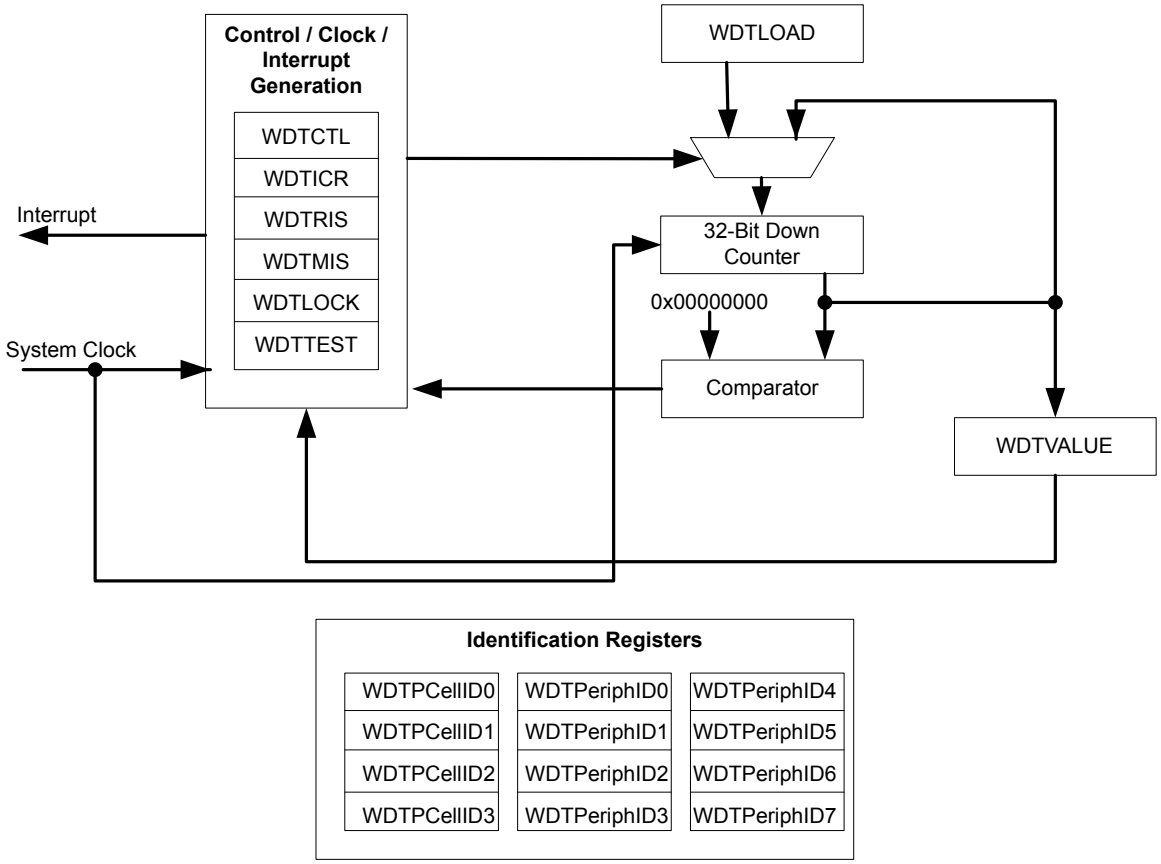
The Stellaris® Watchdog Timer module has the following features:

- 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 controller 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.

10.1 Block Diagram

Figure 10-1. WDT Module Block Diagram



10.2 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 (via the `WatchdogResetEnable` function), 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.

10.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the WDT bit in the **RCGC0** register. The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If the Watchdog is configured to trigger system resets, set the RESEN bit in the **WDTCTL** register.
3. 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.

10.4 Register Map

Table 10-1 on page 372 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address of 0x4000.0000.

Table 10-1. Watchdog Timer Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	374
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	375
0x008	WDTCTL	R/W	0x0000.0000	Watchdog Control	376
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	377
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	378
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	379
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	380
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	381
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	382
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	383
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	384
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	385
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	386
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	387
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	388

Table 10-1. Watchdog Timer Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	389
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	390
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	391
0xFF8	WDTPCellID2	RO	0x0000.0005	Watchdog PrimeCell Identification 2	392
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	393

10.5 Register Descriptions

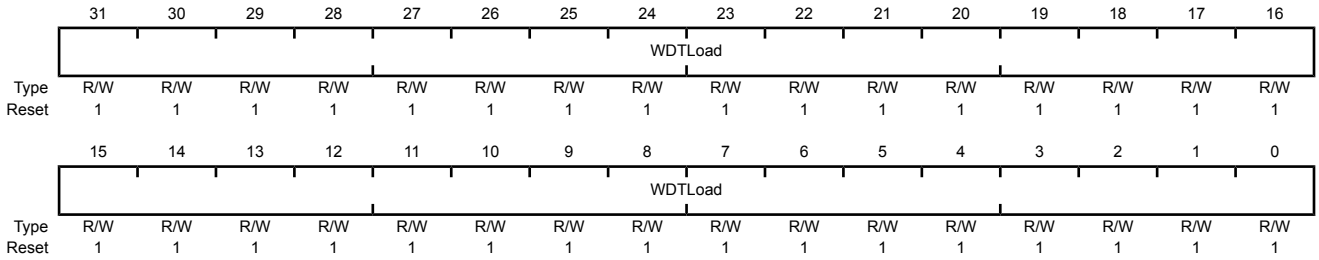
The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

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)

Base 0x4000.0000
Offset 0x000
Type R/W, reset 0xFFFF.FFFF



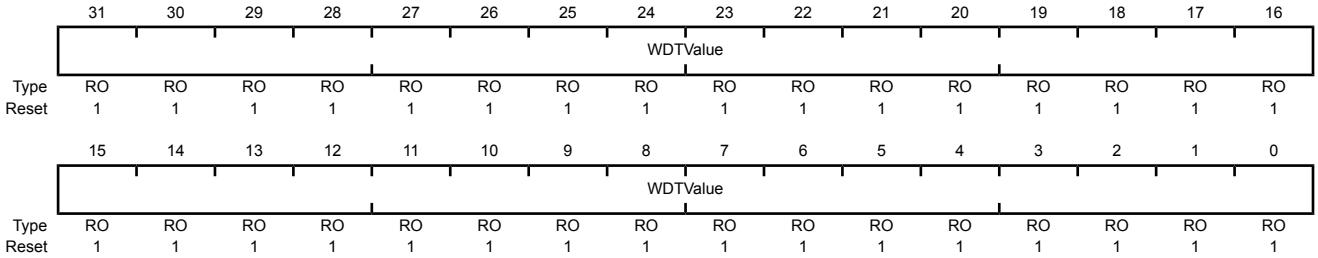
Bit/Field	Name	Type	Reset	Description
31:0	WDTLoad	R/W	0xFFFF.FFFF	Watchdog Load Value

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

Base 0x4000.0000
Offset 0x004
Type RO, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	WDTValue	RO	0xFFFF.FFFF	Watchdog Value Current value of the 32-bit down counter.

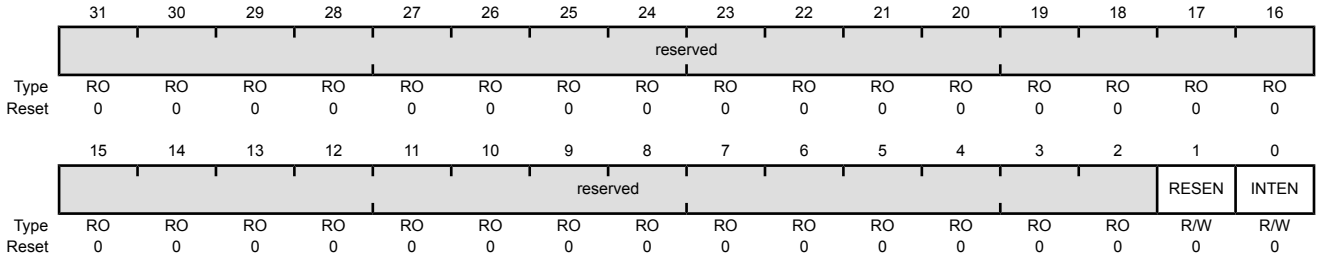
Register 3: Watchdog Control (WDTCTL), offset 0x008

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, all subsequent writes to the control register are ignored. The only mechanism that can re-enable writes is a hardware reset.

Watchdog Control (WDTCTL)

Base 0x4000.0000
Offset 0x008
Type R/W, reset 0x0000.0000



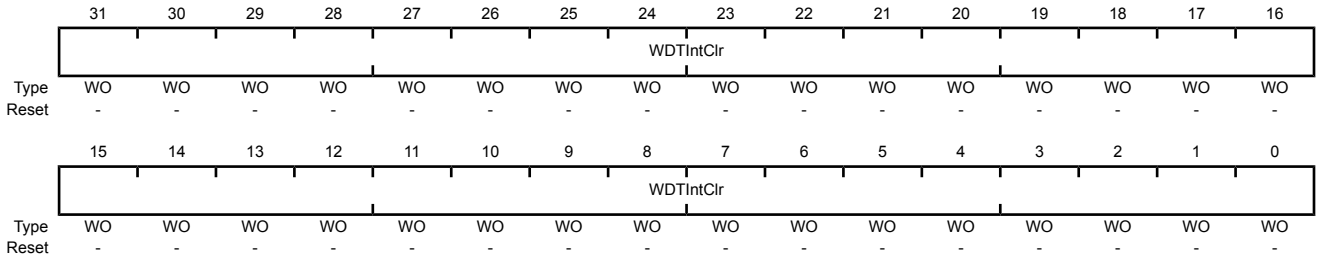
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RESEN	R/W	0	Watchdog Reset Enable The RESEN values are defined as follows: Value Description 0 Disabled. 1 Enable the Watchdog module reset output.
0	INTEN	R/W	0	Watchdog Interrupt Enable The INTEN values are defined as follows: Value Description 0 Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset). 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)

Base 0x4000.0000
Offset 0x00C
Type WO, reset -



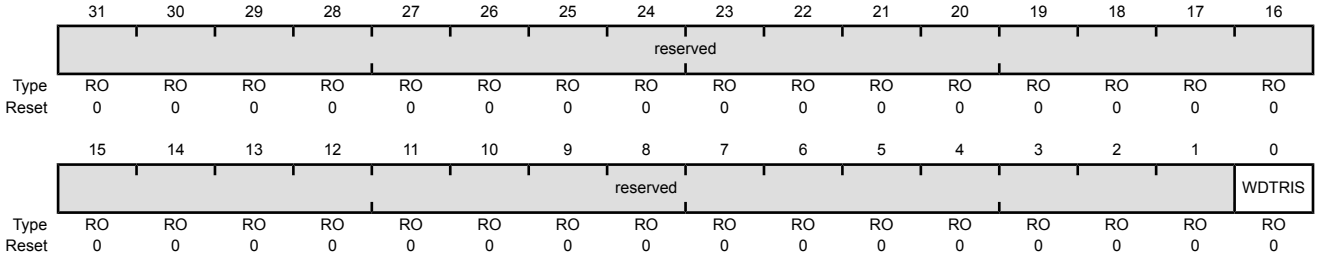
Bit/Field	Name	Type	Reset	Description
31:0	WDTIntClr	WO	-	Watchdog Interrupt Clear

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)

Base 0x4000.0000
Offset 0x010
Type RO, reset 0x0000.0000



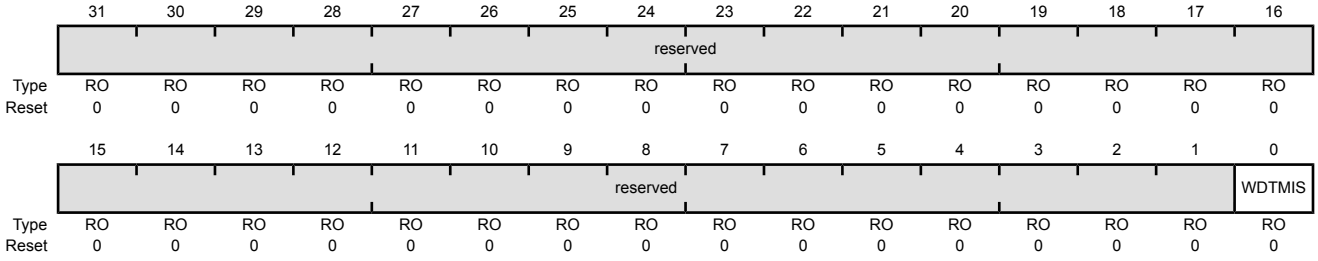
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTRIS	RO	0	Watchdog Raw Interrupt Status Gives the raw interrupt state (prior to masking) of **WDTINTR**.

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)

Base 0x4000.0000
Offset 0x014
Type RO, reset 0x0000.0000



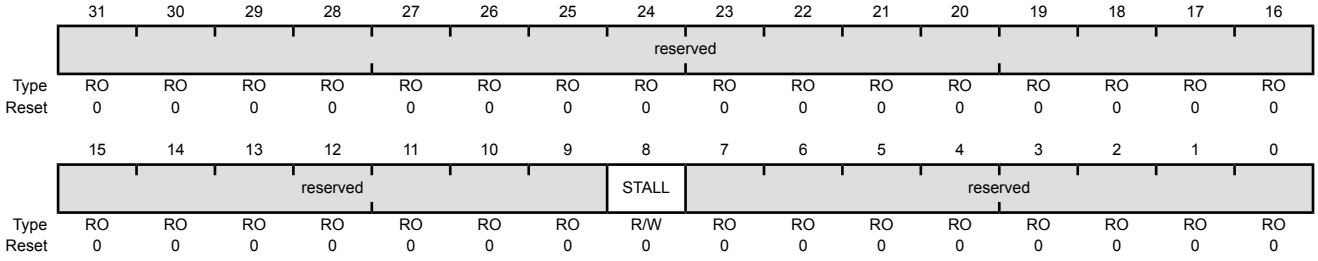
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTMIS	RO	0	Watchdog Masked Interrupt Status Gives the masked interrupt state (after masking) of the **WDTINTR** interrupt.

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)

Base 0x4000.0000
Offset 0x418
Type R/W, reset 0x0000.0000



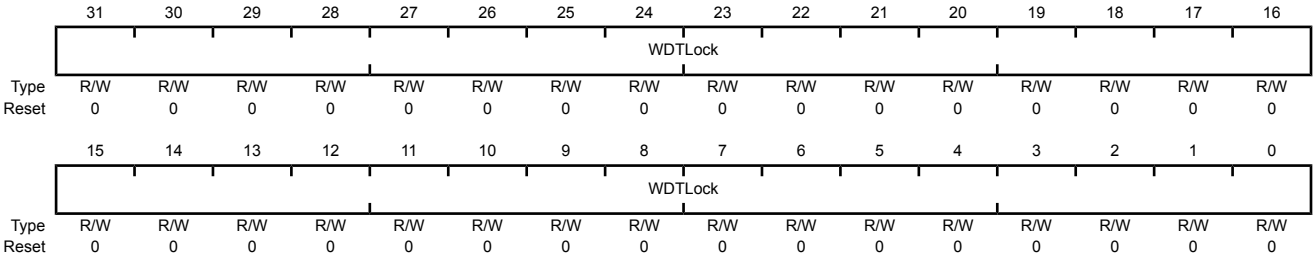
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable When set to 1, if the Stellaris microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
7:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. 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)

Base 0x4000.0000
Offset 0xC00
Type R/W, reset 0x0000.0000



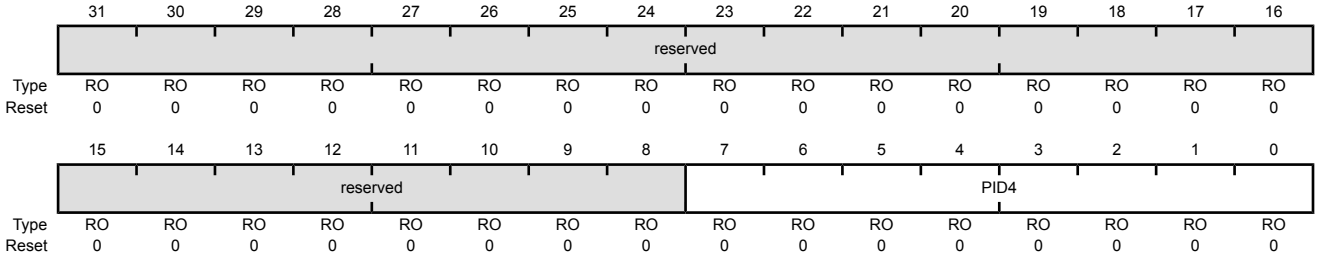
Bit/Field	Name	Type	Reset	Description
31:0	WDTLock	R/W	0x0000	Watchdog Lock A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates. A read of this register returns the following values: Value Description 0x0000.0001 Locked 0x0000.0000 Unlocked

Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 4 (WDTPeriphID4)

Base 0x4000.0000
Offset 0xFD0
Type RO, reset 0x0000.0000



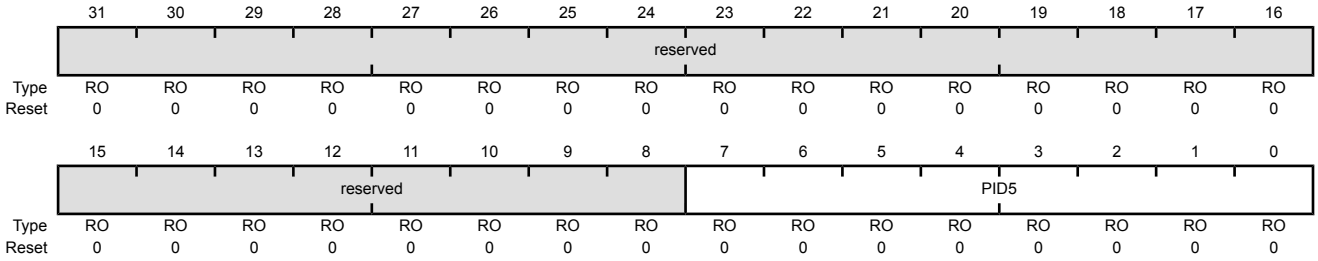
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register[7:0]

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)

Base 0x4000.0000
Offset 0xFD4
Type RO, reset 0x0000.0000



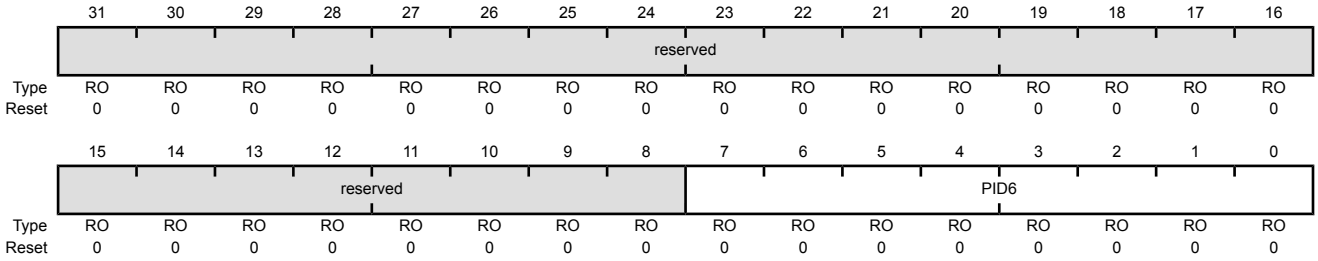
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	WDT Peripheral ID Register[15:8]

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)

Base 0x4000.0000
Offset 0xFD8
Type RO, reset 0x0000.0000



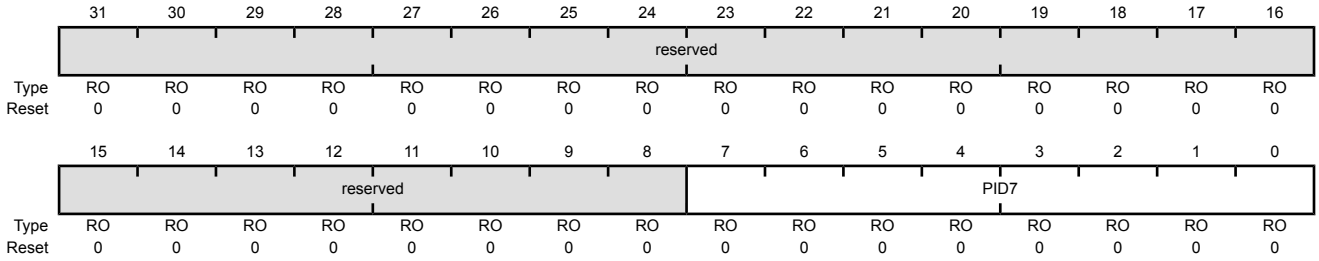
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	WDT Peripheral ID Register[23:16]

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)

Base 0x4000.0000
Offset 0xFDC
Type RO, reset 0x0000.0000



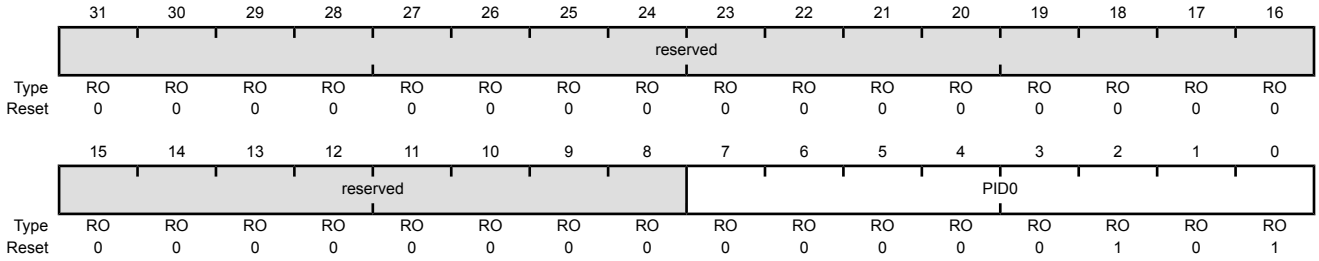
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	WDT Peripheral ID Register[31:24]

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)

Base 0x4000.0000
Offset 0xFE0
Type RO, reset 0x0000.0005



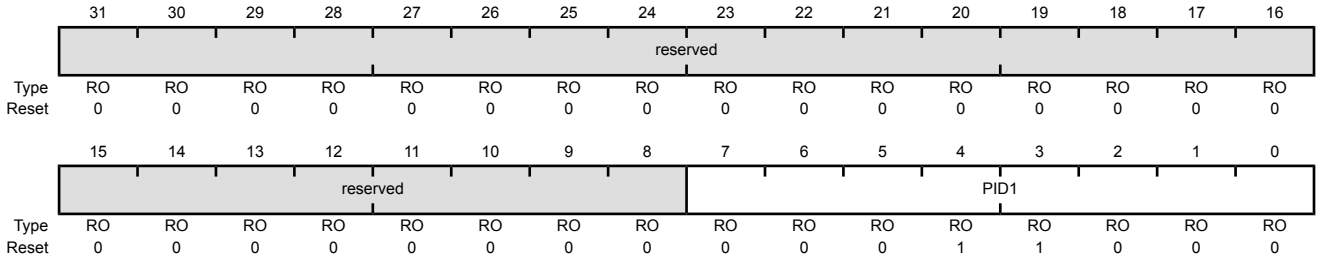
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register[7:0]

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)

Base 0x4000.0000
Offset 0xFE4
Type RO, reset 0x0000.0018



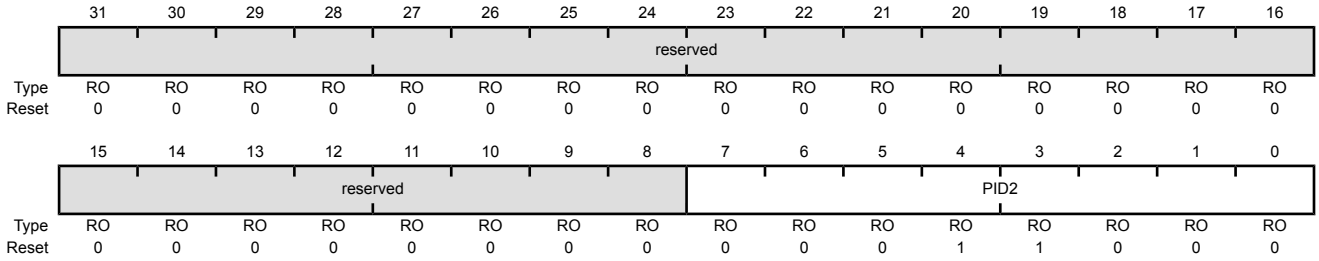
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register[15:8]

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)

Base 0x4000.0000
Offset 0xFE8
Type RO, reset 0x0000.0018



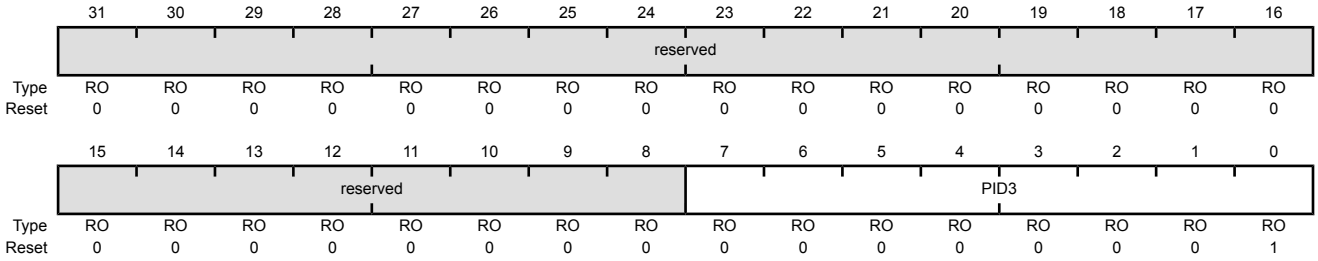
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register[23:16]

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)

Base 0x4000.0000
Offset 0xFEC
Type RO, reset 0x0000.0001



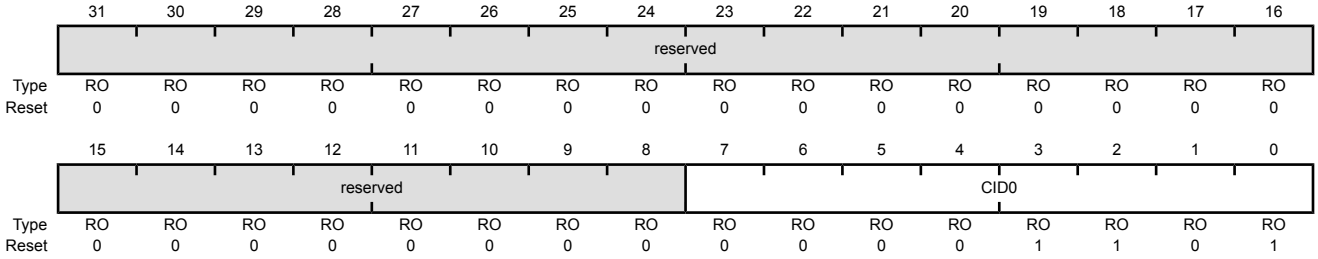
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register[31:24]

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)

Base 0x4000.0000
Offset 0xFF0
Type RO, reset 0x0000.000D



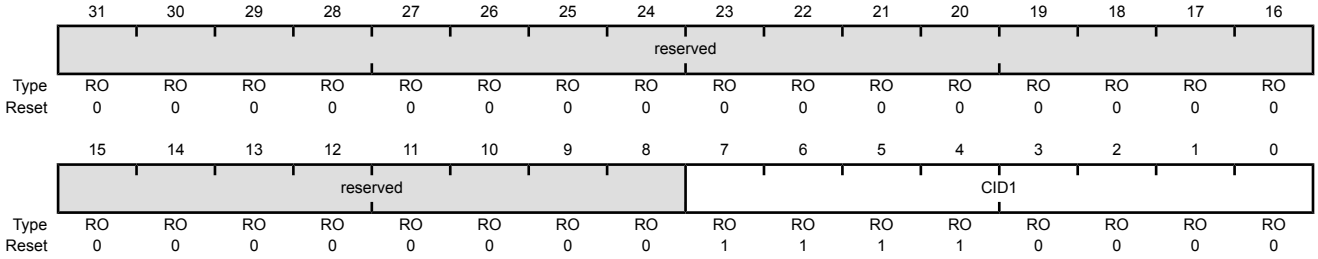
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register[7:0]

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)

Base 0x4000.0000
Offset 0xFF4
Type RO, reset 0x0000.00F0



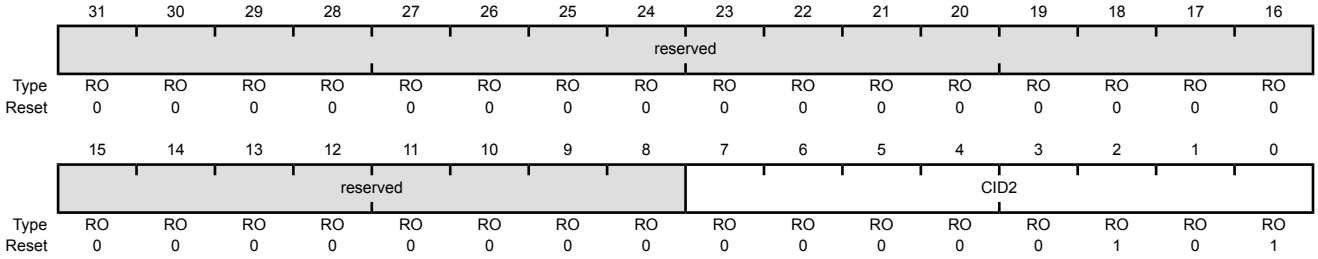
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register[15:8]

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)

Base 0x4000.0000
Offset 0xFF8
Type RO, reset 0x0000.0005



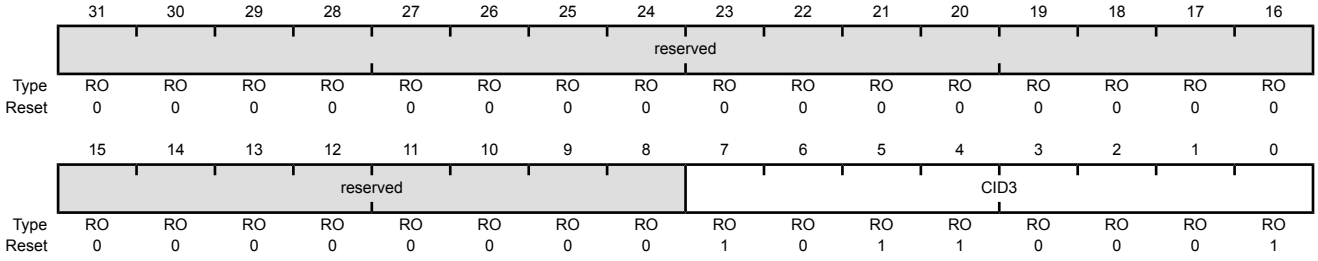
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	Watchdog PrimeCell ID Register[23:16]

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)

Base 0x4000.0000
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	Watchdog PrimeCell ID Register[31:24]

11 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.

The Stellaris® ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. The ADC module contains four programmable sequencer which allows for the sampling of multiple analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

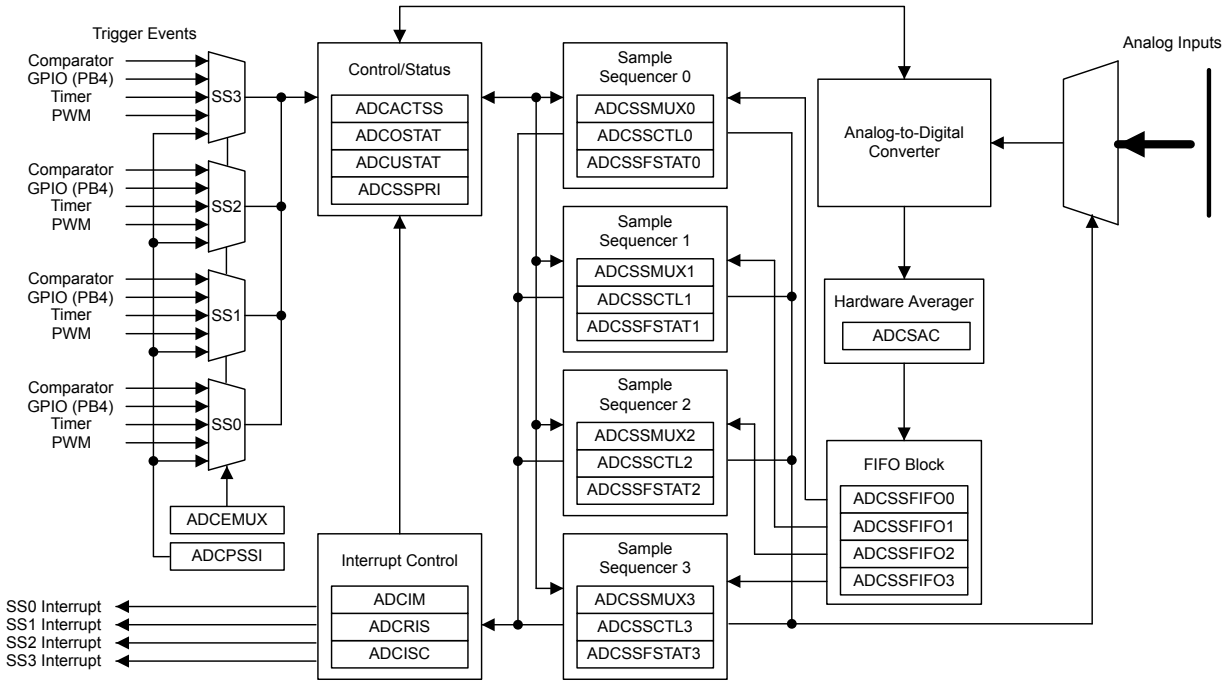
The Stellaris ADC module provides the following features:

- Eight analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Sample rate of one million samples/second
- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - GPIO
- Hardware averaging of up to 64 samples for improved accuracy
- Converter uses an internal 3-V reference
- Power and ground for the analog circuitry is separate from the digital power and ground

11.1 Block Diagram

Figure 11-1 on page 395 provides details on the internal configuration of the ADC controls and data registers.

Figure 11-1. ADC Module Block Diagram



11.2 Signal Description

Table 11-1 on page 395 and Table 11-2 on page 395 list the external signals of the ADC module and describe the function of each. The signals are analog functions for some GPIO signals. The column in the table below titled "Pin Assignment" lists the GPIO pin placement for the ADC signals. The AINx 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. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286.

Table 11-1. ADC Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC0	1	I	Analog	Analog-to-digital converter input 0.
ADC1	2	I	Analog	Analog-to-digital converter input 1.
ADC2	5	I	Analog	Analog-to-digital converter input 2.
ADC3	6	I	Analog	Analog-to-digital converter input 3.
ADC4	100	I	Analog	Analog-to-digital converter input 4.
ADC5	99	I	Analog	Analog-to-digital converter input 5.
ADC6	96	I	Analog	Analog-to-digital converter input 6.
ADC7	95	I	Analog	Analog-to-digital converter input 7.

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

Table 11-2. ADC Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC0	B1	I	Analog	Analog-to-digital converter input 0.

Table 11-2. ADC Signals (108BGA) *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC1	A1	I	Analog	Analog-to-digital converter input 1.
ADC2	B3	I	Analog	Analog-to-digital converter input 2.
ADC3	B2	I	Analog	Analog-to-digital converter input 3.
ADC4	A2	I	Analog	Analog-to-digital converter input 4.
ADC5	A3	I	Analog	Analog-to-digital converter input 5.
ADC6	B4	I	Analog	Analog-to-digital converter input 6.
ADC7	A4	I	Analog	Analog-to-digital converter input 7.

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

11.3 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 controller. 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.

11.3.1 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 11-3 on page 396 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. In this implementation, each FIFO entry is a 32-bit word, with the lower 10 bits containing the conversion result.

Table 11-3. 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 two 4-bit nibbles 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** nibbles select the input pin, while the **ADCSSCTLn** nibbles 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.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence is 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. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

11.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- Sequence prioritization
- Trigger configuration

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured automatically by hardware when the system XTAL is selected. The automatic clock divider configuration targets 16.667 MHz operation for all Stellaris devices.

11.3.2.1 Interrupts

The register configurations of the sample sequencers 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**.

11.3.2.2 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.

11.3.2.3 Sampling Events

Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. The external peripheral triggering sources vary by Stellaris family member, but all devices share the "Controller" and "Always" triggers. Software can initiate sampling by setting the SSx bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

Care must be taken when using the "Always" trigger. If a sequence's priority is too high, it is possible to starve other lower priority sequences.

11.3.3 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 416). There is a single averaging circuit and all input channels receive the same amount of averaging whether they are single-ended or differential.

11.3.4 Analog-to-Digital Converter

The converter itself generates a 10-bit output value for selected analog input. Special analog pads are used to minimize the distortion on the input. An internal 3 V reference is used by the converter resulting in sample values ranging from 0x000 at 0 V input to 0x3FF at 3 V input when in single-ended input mode.

11.3.5 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, its corresponding value in the **ADCSSMUXn** register must be set to one of the four differential pairs, numbered 0-3. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 11-4 on page 398). The ADC does not support other differential pairings such as analog input 0 with analog input 3. The number of differential pairs supported is dependent on the number of analog inputs (see Table 11-4 on page 398).

Table 11-4. Differential Sampling Pairs

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7

The voltage sampled in differential mode is the difference between the odd and even channels:

ΔV (differential voltage) = V_{IN_EVEN} (even channels) – V_{IN_ODD} (odd channels), therefore:

- If $\Delta V = 0$, then the conversion result = 0x1FF
- If $\Delta V > 0$, then the conversion result > 0x1FF (range is 0x1FF–0x3FF)
- If $\Delta V < 0$, then the conversion result < 0x1FF (range is 0–0x1FF)

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 11-2 on page 399 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 11-3 on page 399 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 since the input voltage is less than 0 V. Figure 11-4 on page 400 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 11-2. Differential Sampling Range, V_{IN_ODD} = 1.5 V

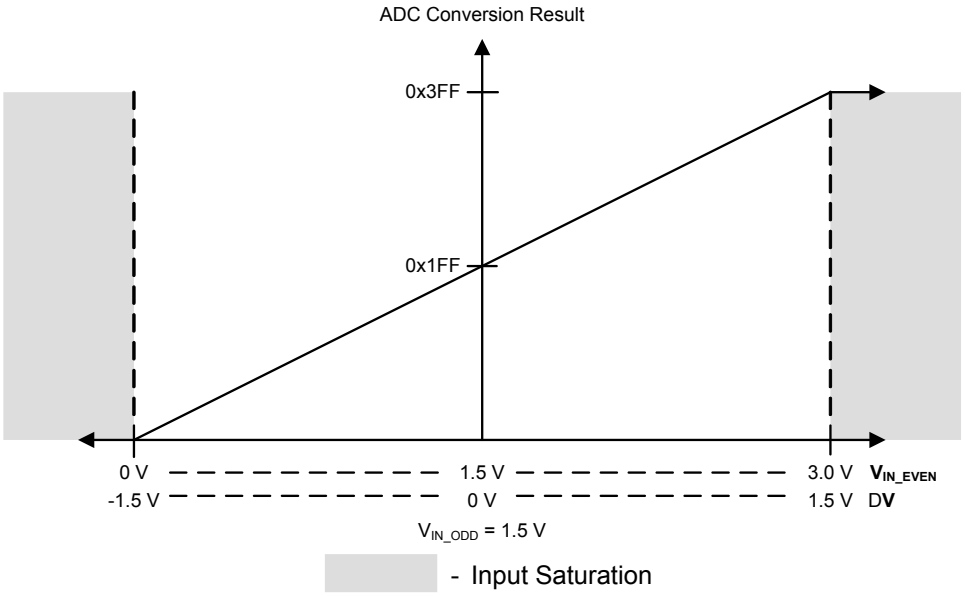


Figure 11-3. Differential Sampling Range, V_{IN_ODD} = 0.75 V

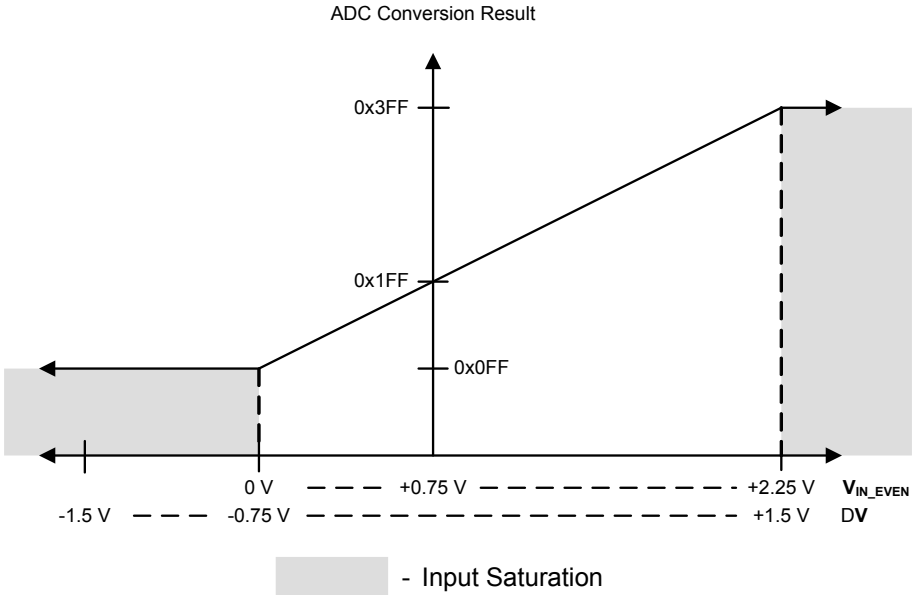
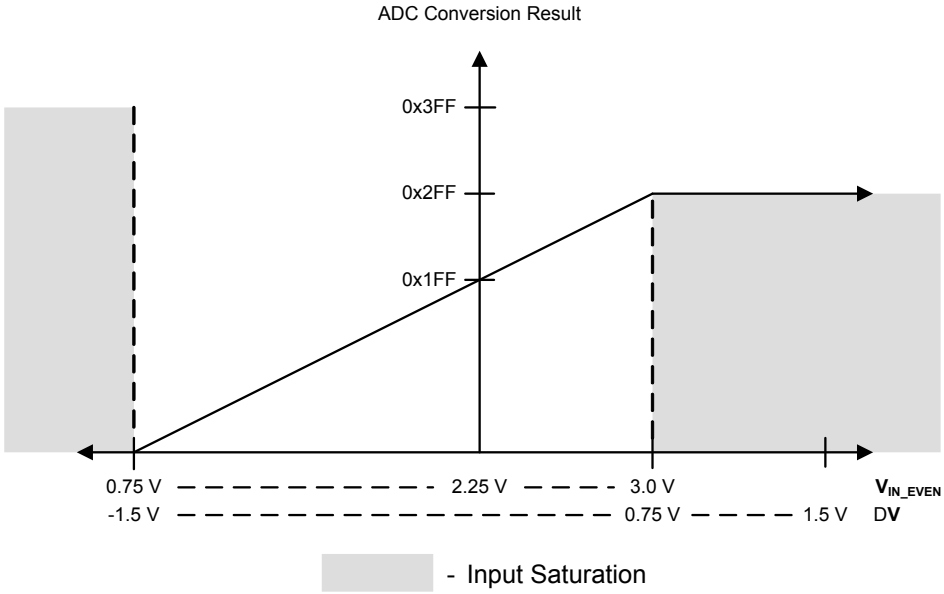


Figure 11-4. Differential Sampling Range, V_{IN_ODD} = 2.25 V



11.3.6 Test Modes

There is a user-available test mode that allows for loopback operation within the digital portion of the ADC module. This can be useful for debugging software without having to provide actual analog stimulus. This mode is available through the **ADC Test Mode Loopback (ADCTMLB)** register (see page 429).

11.3.7 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.

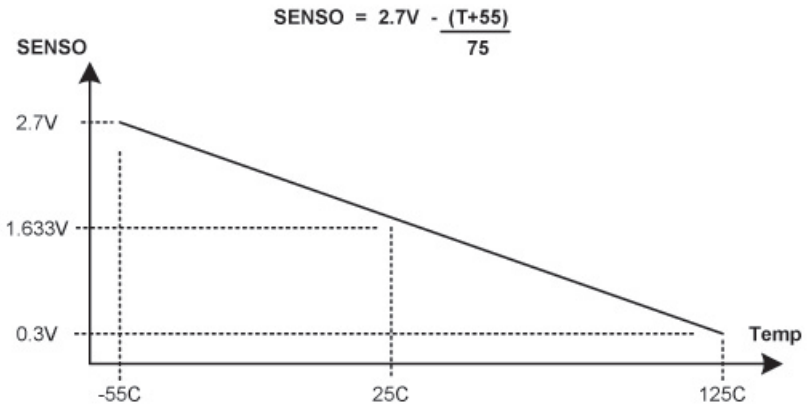
The temperature sensor does not have a separate enable, since it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal SENSO is given by the following equation:

SENSO = 2.7 - ((T + 55) / 75)

This relation is shown in Figure 11-5 on page 401.

Figure 11-5. Internal Temperature Sensor Characteristic



11.4 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and using a supported crystal frequency (see the **RCC** register). Using unsupported frequencies can cause faulty operation in the ADC module.

11.4.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps. The main steps include enabling the clock to the ADC and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC0** register (see page 213).
2. 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.

11.4.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization since each sample sequence is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by writing a 0 to 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, write a 1 to the corresponding MASK bit in the **ADCIM** register.

6. Enable the sample sequencer logic by writing a 1 to the corresponding ASENn bit in the **ADCACTSS** register.

11.5 Register Map

Table 11-5 on page 402 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to the ADC base address of 0x4003.8000.

Note that the ADC module clock must be enabled before the registers can be programmed (see page 213). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

Table 11-5. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	404
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	405
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	406
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	407
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	408
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	409
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	412
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	413
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	415
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	416
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	417
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	419
0x048	ADCSSFIFO0	RO	-	ADC Sample Sequence Result FIFO 0	422
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	423
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	424
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	425
0x068	ADCSSFIFO1	RO	-	ADC Sample Sequence Result FIFO 1	422
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	423
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	424
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	425
0x088	ADCSSFIFO2	RO	-	ADC Sample Sequence Result FIFO 2	422

Table 11-5. ADC Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	423
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	427
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	428
0x0A8	ADCSSFIFO3	RO	-	ADC Sample Sequence Result FIFO 3	422
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	423
0x100	ADCTMLB	R/W	0x0000.0000	ADC Test Mode Loopback	429

11.6 Register Descriptions

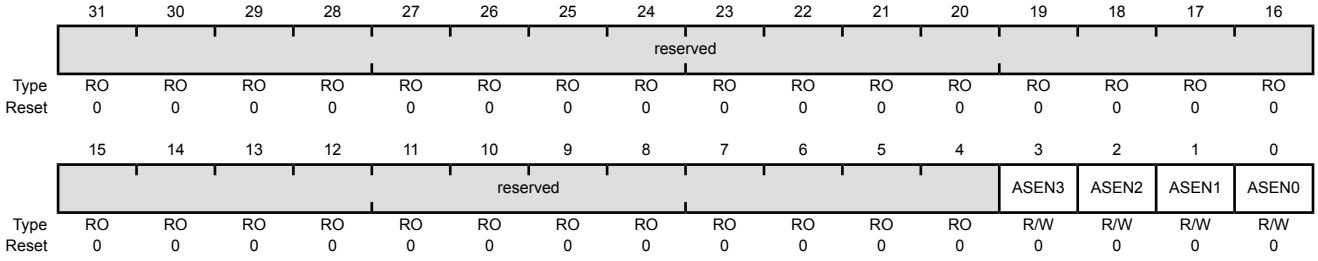
The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

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)

Base 0x4003.8000
Offset 0x000
Type R/W, reset 0x0000.0000



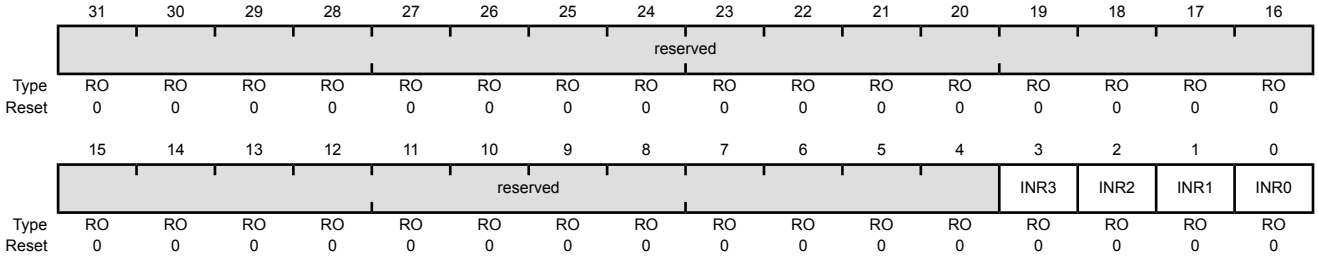
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	ADC SS3 Enable Specifies whether Sample Sequencer 3 is enabled. If set, the sample sequence logic for Sequencer 3 is active. Otherwise, the sequencer is inactive.
2	ASEN2	R/W	0	ADC SS2 Enable Specifies whether Sample Sequencer 2 is enabled. If set, the sample sequence logic for Sequencer 2 is active. Otherwise, the sequencer is inactive.
1	ASEN1	R/W	0	ADC SS1 Enable Specifies whether Sample Sequencer 1 is enabled. If set, the sample sequence logic for Sequencer 1 is active. Otherwise, the sequencer is inactive.
0	ASEN0	R/W	0	ADC SS0 Enable Specifies whether Sample Sequencer 0 is enabled. If set, the sample sequence logic for Sequencer 0 is active. Otherwise, the sequencer is inactive.

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 having to generate controller interrupts.

ADC Raw Interrupt Status (ADCRIS)

Base 0x4003.8000
Offset 0x004
Type RO, reset 0x0000.0000



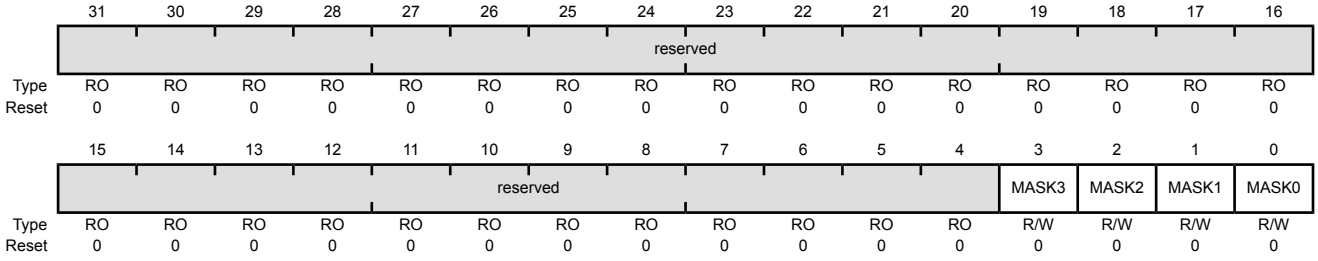
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL3** `IE` bit has completed conversion. This bit is cleared by setting the `IN3` bit in the **ADCISC** register.
2	INR2	RO	0	SS2 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL2** `IE` bit has completed conversion. This bit is cleared by setting the `IN2` bit in the **ADCISC** register.
1	INR1	RO	0	SS1 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL1** `IE` bit has completed conversion. This bit is cleared by setting the `IN1` bit in the **ADCISC** register.
0	INR0	RO	0	SS0 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL0** `IE` bit has completed conversion. This bit is cleared by setting the `IN30` bit in the **ADCISC** register.

Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer raw interrupt signals are promoted to controller interrupts. Each raw interrupt signal can be masked independently.

ADC Interrupt Mask (ADCIM)

Base 0x4003.8000
Offset 0x008
Type R/W, reset 0x0000.0000



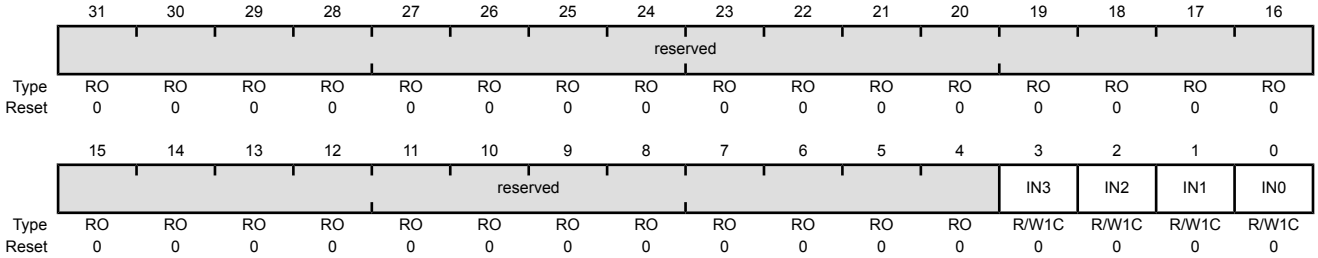
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	R/W	0	SS3 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 3 (**ADCRIS** register `INR3` bit) to be promoted to a controller interrupt. When clear, the status of Sample Sequencer 3 does not affect the SS3 interrupt status.
2	MASK2	R/W	0	SS2 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 2 (**ADCRIS** register `INR2` bit) to be promoted to a controller interrupt. When clear, the status of Sample Sequencer 2 does not affect the SS2 interrupt status.
1	MASK1	R/W	0	SS1 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 1 (**ADCRIS** register `INR1` bit) to be promoted to a controller interrupt. When clear, the status of Sample Sequencer 1 does not affect the SS1 interrupt status.
0	MASK0	R/W	0	SS0 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 0 (**ADCRIS** register `INR0` bit) to be promoted to a controller interrupt. When clear, 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 sequence interrupt conditions and shows the status of controller interrupts generated by the sample sequencers. When read, each bit field is the logical AND of the respective INR and MASK bits. Sample sequence nterrupts are cleared by setting the corresponding bit position. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence INR bits are still cleared via the **ADCISC** register, even if the IN bit is not set.

ADC Interrupt Status and Clear (ADCISC)

Base 0x4003.8000
Offset 0x00C
Type R/W1C, reset 0x0000.0000



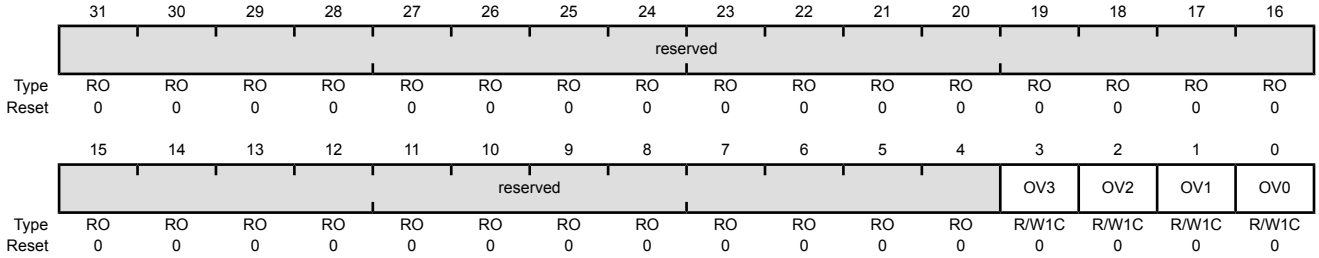
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	R/W1C	0	SS3 Interrupt Status and Clear This bit is set when 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 controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit.
2	IN2	R/W1C	0	SS2 Interrupt Status and Clear This bit is set when 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 controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit.
1	IN1	R/W1C	0	SS1 Interrupt Status and Clear This bit is set when 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 controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit.
0	IN0	R/W1C	0	SS0 Interrupt Status and Clear This bit is set when 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 controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR0 bit.

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)

Base 0x4003.8000
Offset 0x010
Type R/W1C, reset 0x0000.0000



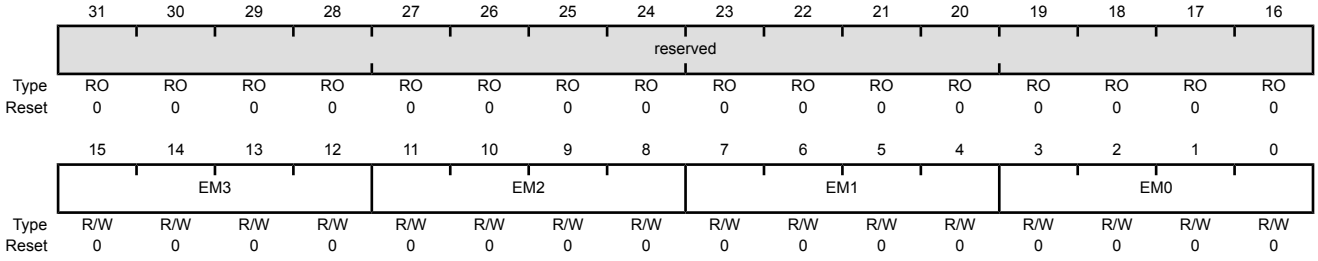
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	SS3 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 3 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.
2	OV2	R/W1C	0	SS2 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 2 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.
1	OV1	R/W1C	0	SS1 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 1 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.
0	OV0	R/W1C	0	SS0 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 0 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

Base 0x4003.8000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	EM3	R/W	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are:

Value	Event
0x0	Controller (default)
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	Analog Comparator 2
0x4	External (GPIO PB4)
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 351).
0x6	reserved
0x7	reserved
0x8	reserved
0x9-0xE	reserved
0xF	Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
11:8	EM2	R/W	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations for this field are: Value Event 0x0 Controller (default) 0x1 Analog Comparator 0 0x2 Analog Comparator 1 0x3 Analog Comparator 2 0x4 External (GPIO PB4) 0x5 Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 351). 0x6 reserved 0x7 reserved 0x8 reserved 0x9-0xE reserved 0xF Always (continuously sample)
7:4	EM1	R/W	0x0	SS1 Trigger Select This field selects the trigger source for Sample Sequencer 1. The valid configurations for this field are: Value Event 0x0 Controller (default) 0x1 Analog Comparator 0 0x2 Analog Comparator 1 0x3 Analog Comparator 2 0x4 External (GPIO PB4) 0x5 Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 351). 0x6 reserved 0x7 reserved 0x8 reserved 0x9-0xE reserved 0xF Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
3:0	EM0	R/W	0x0	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0. The valid configurations for this field are:

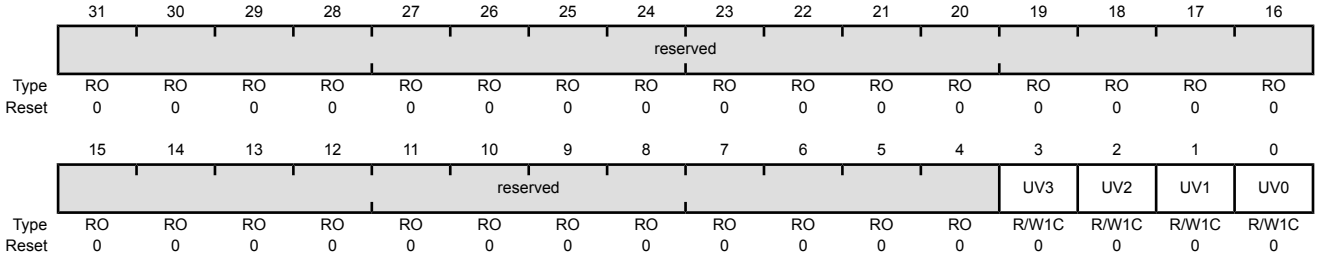
Value	Event
0x0	Controller (default)
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	Analog Comparator 2
0x4	External (GPIO PB4)
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 351).
0x6	reserved
0x7	reserved
0x8	reserved
0x9-0xE	reserved
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)

Base 0x4003.8000
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 When set, this bit specifies that the FIFO for Sample Sequencer 3 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
2	UV2	R/W1C	0	SS2 FIFO Underflow When set, this bit specifies that the FIFO for Sample Sequencer 2 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
1	UV1	R/W1C	0	SS1 FIFO Underflow When set, this bit specifies that the FIFO for Sample Sequencer 1 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
0	UV0	R/W1C	0	SS0 FIFO Underflow When set, this bit specifies that the FIFO for Sample Sequencer 0 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. 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)

Base 0x4003.8000
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 This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0 is highest and 3 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 This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0 is highest and 3 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 This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0 is highest and 3 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 This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0 is highest and 3 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 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.

ADC Processor Sample Sequence Initiate (ADCPSSI)

Base 0x4003.8000
Offset 0x028
Type WO, 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												SS3	SS2	SS1	SS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-

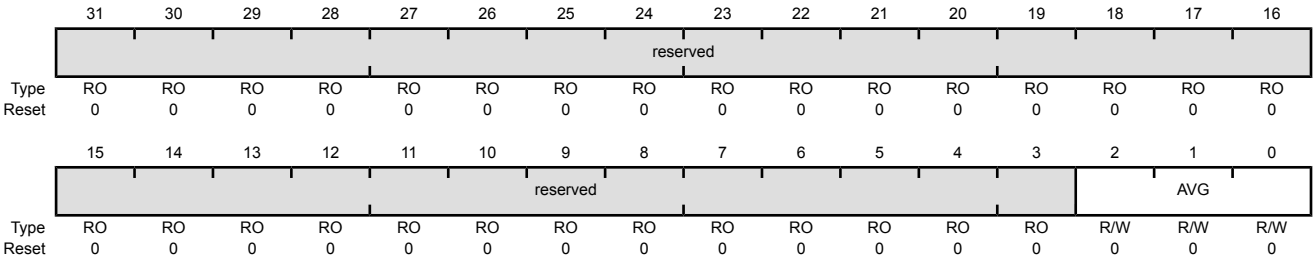
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SS3	WO	-	SS3 Initiate When set, this bit triggers sampling on Sample Sequencer 3 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
2	SS2	WO	-	SS2 Initiate When set, this bit triggers sampling on Sample Sequencer 2 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
1	SS1	WO	-	SS1 Initiate When set, this bit triggers sampling on Sample Sequencer 1 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
0	SS0	WO	-	SS0 Initiate When set, this bit triggers sampling on Sample Sequencer 0 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.

Register 10: 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)

Base 0x4003.8000
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 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 11: 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)

Base 0x4003.8000
Offset 0x040
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	MUX7			reserved	MUX6			reserved	MUX5			reserved	MUX4		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	MUX3			reserved	MUX2			reserved	MUX1			reserved	MUX0		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	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	reserved	RO	0	Software should not rely on the value of 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	MUX7	R/W	0x0	8th Sample Input Select 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 1 indicates the input is ADC1.
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	MUX6	R/W	0x0	7th Sample Input Select 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:20	MUX5	R/W	0x0	6th Sample Input Select 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
18:16	MUX4	R/W	0x0	5th Sample Input Select 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0x0	4th Sample Input Select 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0x0	3rd Sample Input Select The MUX72 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0x0	2nd Sample Input Select 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	reserved	RO	0	Software should not rely on the value of 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	MUX0	R/W	0x0	1st Sample Input Select 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 12: 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 at some point, whether it be after the first sample, last 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)

Base 0x4003.8000
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 This bit is used during the eighth sample of the sample sequence and and specifies the input source of the sample. When set, the temperature sensor is read. When clear, the input pin specified by the **ADCSSMUX** register is read.
30	IE7	R/W	0	8th Sample Interrupt Enable This bit is used during the eighth sample of the sample sequence and specifies whether the raw interrupt signal (INR0 bit) is asserted at the end of the sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to a controller-level interrupt. When this bit is set, the raw interrupt is asserted. When this bit is clear, the raw interrupt is not asserted. It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	8th Sample is End of Sequence The END7 bit indicates that this is the last sample of the sequence. It is possible to end the sequence on any sample position. Samples defined after the sample containing a set END are not requested for conversion even though the fields may be non-zero. It is required that software write the END bit somewhere within the sequence. (Sample Sequencer 3, which only has a single sample in the sequence, is hardwired to have the END0 bit set.) Setting this bit indicates that this sample is the last in the sequence.
28	D7	R/W	0	8th Sample Diff Input Select The D7 bit indicates that the analog input is to be differentially sampled. The corresponding **ADCSSMUXx** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". The temperature sensor does not have a differential option. When set, the analog inputs are differentially sampled.
27	TS6	R/W	0	7th Sample Temp Sensor Select Same definition as TS7 but used during the seventh sample.

Bit/Field	Name	Type	Reset	Description
26	IE6	R/W	0	7th Sample Interrupt Enable Same definition as IE7 but used during the seventh sample.
25	END6	R/W	0	7th Sample is End of Sequence Same definition as END7 but used during the seventh sample.
24	D6	R/W	0	7th Sample Diff Input Select Same definition as D7 but used during the seventh sample.
23	TS5	R/W	0	6th Sample Temp Sensor Select Same definition as TS7 but used during the sixth sample.
22	IE5	R/W	0	6th Sample Interrupt Enable Same definition as IE7 but used during the sixth sample.
21	END5	R/W	0	6th Sample is End of Sequence Same definition as END7 but used during the sixth sample.
20	D5	R/W	0	6th Sample Diff Input Select Same definition as D7 but used during the sixth sample.
19	TS4	R/W	0	5th Sample Temp Sensor Select Same definition as TS7 but used during the fifth sample.
18	IE4	R/W	0	5th Sample Interrupt Enable Same definition as IE7 but used during the fifth sample.
17	END4	R/W	0	5th Sample is End of Sequence Same definition as END7 but used during the fifth sample.
16	D4	R/W	0	5th Sample Diff Input Select Same definition as D7 but used during the fifth sample.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.

Bit/Field	Name	Type	Reset	Description
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 13: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

Register 14: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

Register 15: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

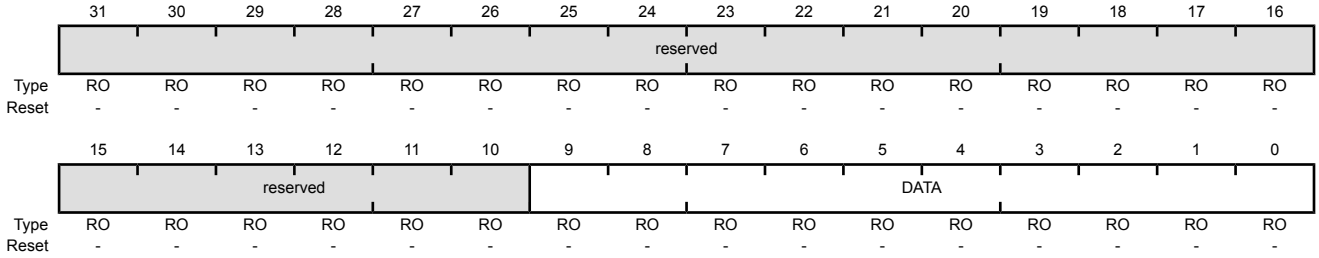
Register 16: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

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 0 (ADCSSFIFO0)

Base 0x4003.8000
Offset 0x048
Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:0	DATA	RO	-	Conversion Result Data

Register 17: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

Register 18: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

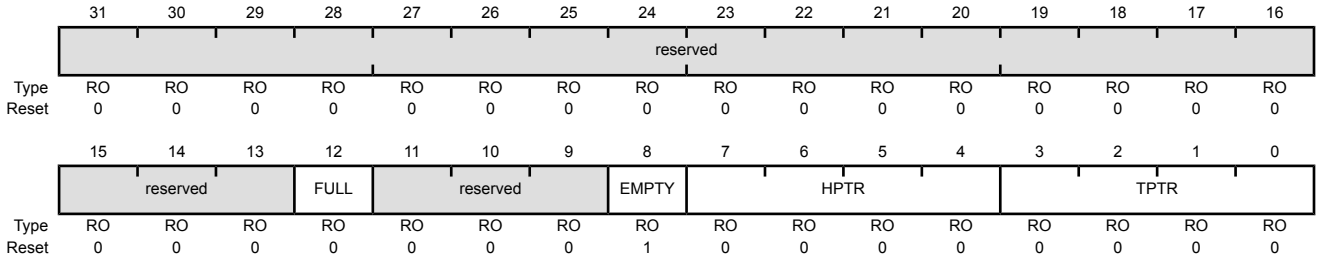
Register 19: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

Register 20: 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. The **ADCSSFSTAT0** register provides status on FIFO0, **ADCSSFSTAT1** on FIFO1, **ADCSSFSTAT2** on FIFO2, and **ADCSSFSTAT3** on FIFO3.

ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

Base 0x4003.8000
Offset 0x04C
Type RO, reset 0x0000.0100



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full When set, this bit indicates that the FIFO is currently full.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	EMPTY	RO	1	FIFO Empty When set, this bit indicates that the FIFO is currently empty.
7:4	HPTR	RO	0x0	FIFO Head Pointer This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.
3:0	TPTR	RO	0x0	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

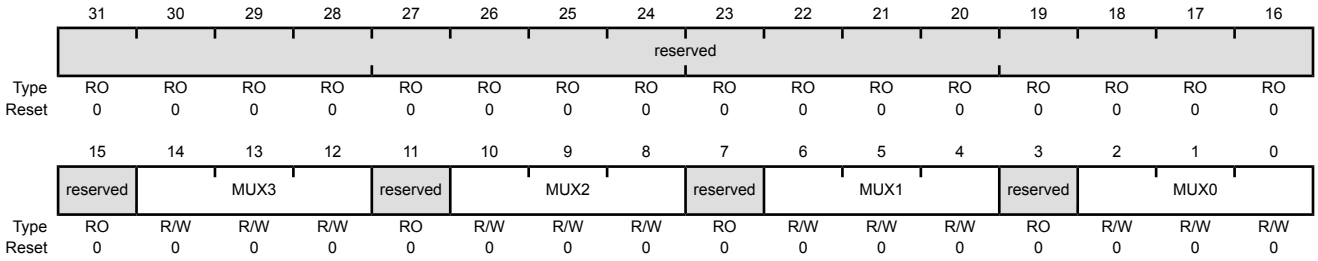
Register 21: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 22: 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 417 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)

Base 0x4003.8000
Offset 0x060
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0x0	4th Sample Input Select
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0x0	3rd Sample Input Select
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0x0	2nd Sample Input Select
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	MUX0	R/W	0x0	1st Sample Input Select

Register 23: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 24: 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 at some point, whether it be after the first sample, last 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 419 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)

Base 0x4003.8000
Offset 0x064
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
	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
31:16	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 Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.

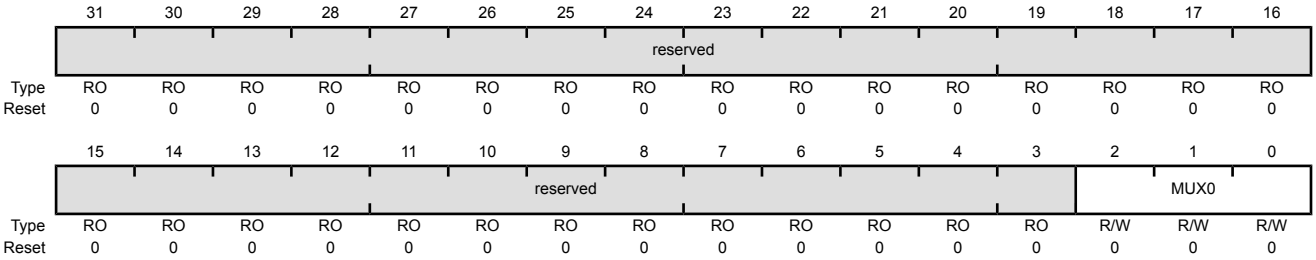
Bit/Field	Name	Type	Reset	Description
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 25: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for a 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 417 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

Base 0x4003.8000
Offset 0x0A0
Type R/W, reset 0x0000.0000



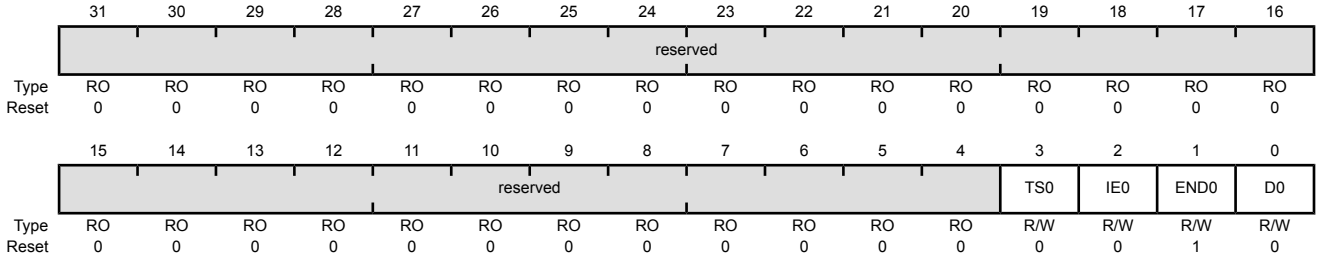
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	1st Sample Input Select

Register 26: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. The END bit is always set since there is only one sample in this sequencer. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 419 for detailed bit descriptions.

ADC Sample Sequence Control 3 (ADCSSCTL3)

Base 0x4003.8000
Offset 0x0A4
Type R/W, reset 0x0000.0002



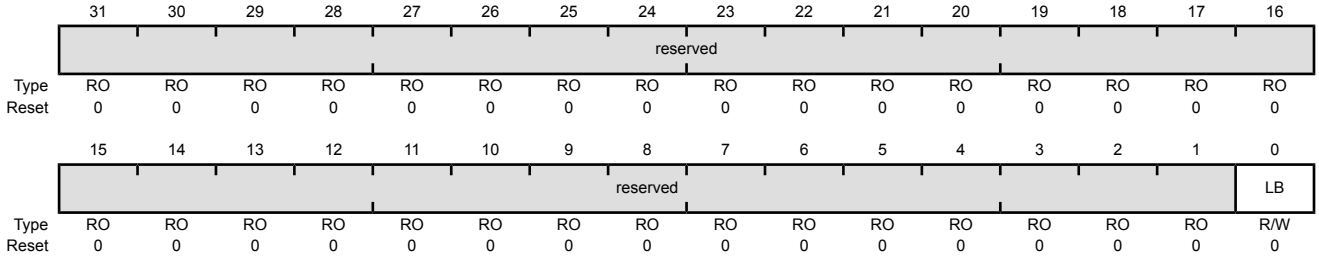
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	1	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 27: ADC Test Mode Loopback (ADCTMLB), offset 0x100

This register provides loopback operation within the digital logic of the ADC, which can be useful in debugging software without having to provide actual analog stimulus. This test mode is entered by writing a value of 0x0000.0001 to this register. When data is read from the FIFO in loopback mode, the read-only portion of this register is returned.

ADC Test Mode Loopback (ADCTMLB)

Base 0x4003.8000
Offset 0x100
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	LB	R/W	0	Loopback Mode Enable When set, forces a loopback within the digital block to provide information on input and unique numbering. The **ADCSSFIFOn** registers do not provide sample data, but instead provide the 10-bit loopback data as shown below.

Bit/Field	Name	Description
9:6	CNT	Continuous Sample Counter Continuous sample counter that is initialized to 0 and counts each sample as it processed. This helps provide a unique value for the data received.
5	CONT	Continuation Sample Indicator When set, indicates that this is a continuation sample. For example, if two sequencers were to run back-to-back, this indicates that the controller kept continuously sampling at full rate.
4	DIFF	Differential Sample Indicator When set, indicates that this is a differential sample.
3	TS	Temp Sensor Sample Indicator When set, indicates that this is a temperature sensor sample.
2:0	MUX	Analog Input Indicator Indicates which analog input is to be sampled.

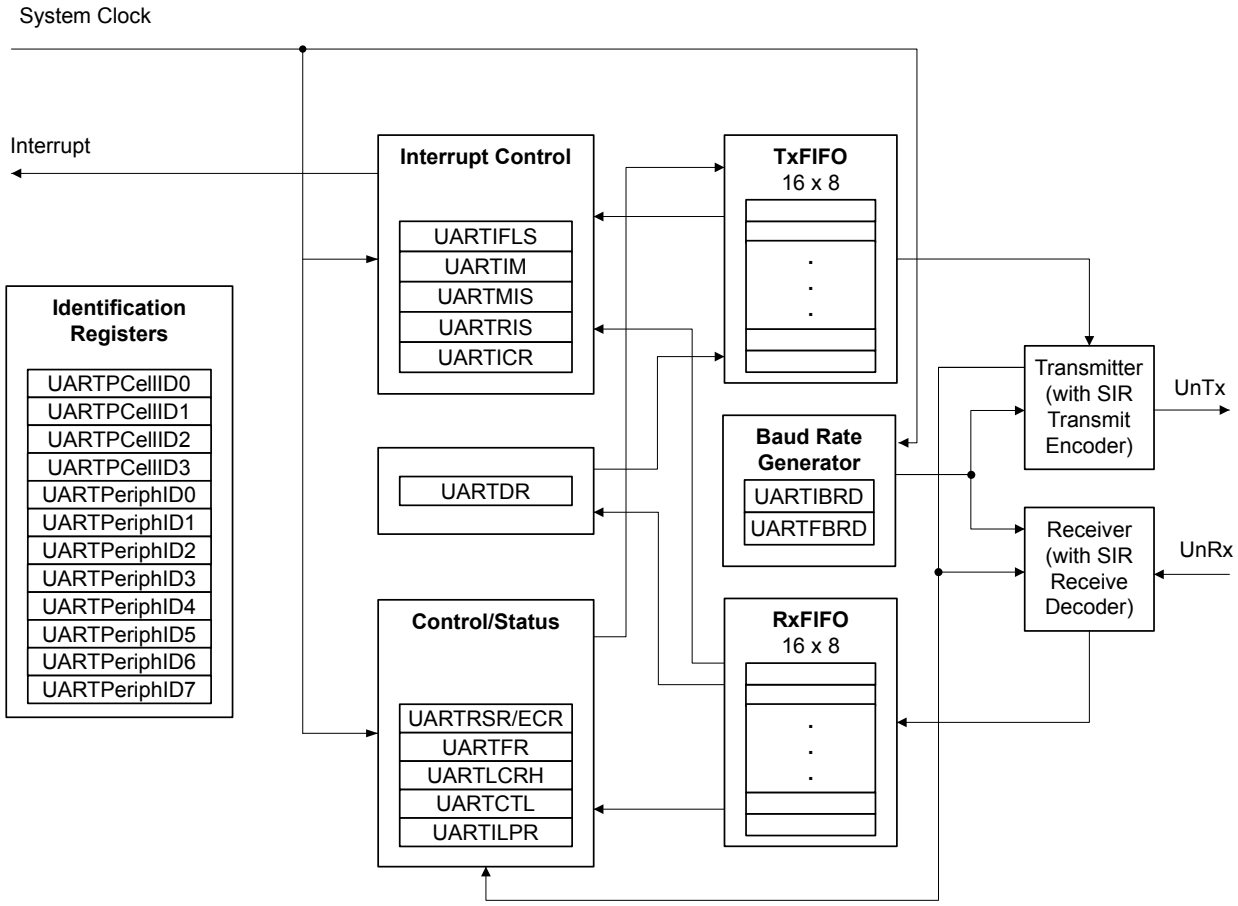
12 Universal Asynchronous Receivers/Transmitters (UARTs)

Each Stellaris® Universal Asynchronous Receiver/Transmitter (UART) has the following features:

- Three fully programmable 16C550-type UARTs with IrDA support
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable baud-rate generator allowing speeds up to 3.125 Mbps
- 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

12.1 Block Diagram

Figure 12-1. UART Module Block Diagram



12.2 Signal Description

Table 12-1 on page 431 and Table 12-2 on page 432 list the external signals of the UART module and describe 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 Assignment" lists the possible GPIO pin placements for these UART signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the UART function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286.

Table 12-1. UART Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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.
U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.

Table 12-1. UART Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	19	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	18	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 12-2. UART Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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.
U1Rx	H2	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	H1	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	K1	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	K2	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.

12.3 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 451). 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.

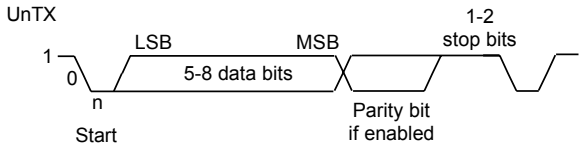
The UART peripheral also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the UARTCTL register.

12.3.1 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 12-2 on page 433 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 12-2. UART Character Frame



12.3.2 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 447) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 448). 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 / (16 * Baud Rate)
```

where UARTSysClk is the system clock connected to the UART.

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 16x the baud-rate (referred to as Baud16). This reference clock is divided by 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 449), 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

12.3.3 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 444) 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 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 (described in “Transmit/Receive Logic” on page 432).

The start bit is valid and recognized if UnRx is still low on the eighth cycle of Baud16, otherwise it is ignored. After a valid start bit is detected, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if UnRx is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

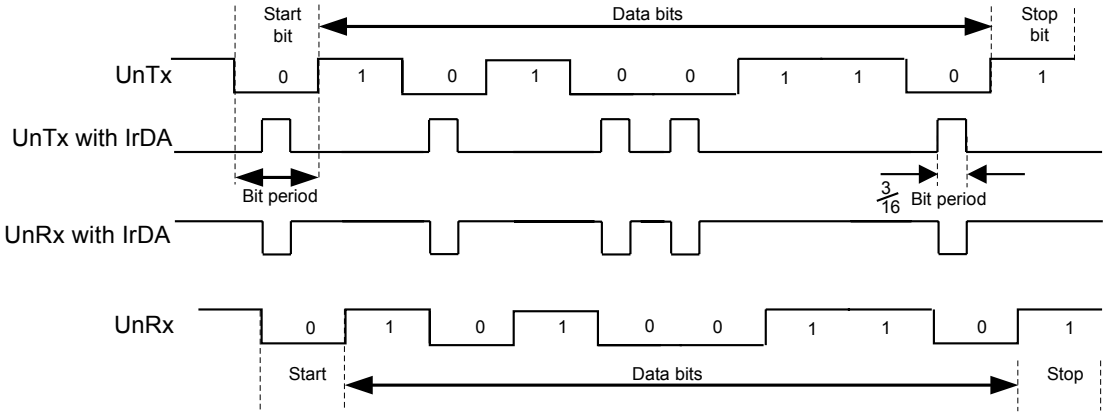
12.3.4 Serial IR (SIR)

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 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. The UART signal pins can be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as 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. This drives 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 446 for more information on IrDA low-power pulse-duration configuration.

Figure 12-3 on page 435 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 12-3. 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.

If the application does not require the use of the UnRx signal, the GPIO pin that has the UnRx signal as an alternate function must be configured as the UnRx signal and pulled High.

12.3.5 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 440). 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 449).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 444) 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 453). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include 1/8, ¼, ½, ¾, and 7/8. 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.

12.3.6 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)
- 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 458).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM**) register (see page 455) by setting the corresponding IM bit to 1. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 457).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by setting the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 459).

The receive interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level, the RXRIS bit is set. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt by writing a 1 to the RXIC bit.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the RXRIS bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt by writing a 1 to the RXIC bit.

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO progresses through the programmed trigger level, the TXRIS bit is set. The transmit interrupt is based on a transition through level, therefore the FIFO must be written past the programmed trigger level otherwise no further transmit interrupts will be generated. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt by writing a 1 to the TXIC bit.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXRIS bit is set. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt by writing a 1 to the TXIC bit.

12.3.7 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the LBE bit in the **UARTCTL** register (see page 451). In loopback mode, data transmitted on UnTx is received on the UnRx input.

12.3.8 IrDA SIR block

The IrDA SIR block contains an IrDA serial IR (SIR) protocol encoder/decoder. When enabled, the SIR block uses the UnTx and UnRx pins for the SIR protocol, which should be connected to an IR transceiver.

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.

12.4 Initialization and Configuration

To use the UARTs, the peripheral clock must be enabled by setting the UART0, UART1, or UART2 bits in the **RCGC1** register.

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), since the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 433, 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 447) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 448) 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. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

12.5 Register Map

Table 12-3 on page 438 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 219). 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 451) 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 12-3. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	440
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	442
0x018	UARTFR	RO	0x0000.0090	UART Flag	444
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	446
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	447
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	448
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	449
0x030	UARTCTL	R/W	0x0000.0300	UART Control	451
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	453
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	455
0x03C	UARTRIS	RO	0x0000.0000	UART Raw Interrupt Status	457
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	458
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	459
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	461
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	462
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	463
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	464
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART Peripheral Identification 0	465
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	466
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	467
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	468

Table 12-3. UART Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	469
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	470
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	471
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	472

12.6 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).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are 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 FIFOs are 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	0	Software should not rely on the value of 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 The OE values are defined as follows: Value Description 0 There has been no data loss due to a FIFO overrun. 1 New data was received when the FIFO was full, resulting in data loss.
10	BE	RO	0	UART Break Error This bit is set to 1 when a break condition is 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). 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 This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	RO	0	UART Framing Error This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).
7:0	DATA	R/W	0	Data Transmitted or Received When written, the data that is to be transmitted via the UART. When read, 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.

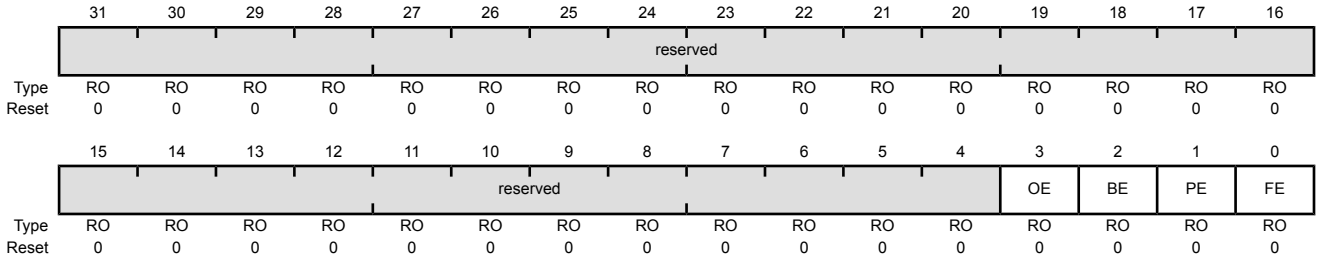
The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset.

Reads

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



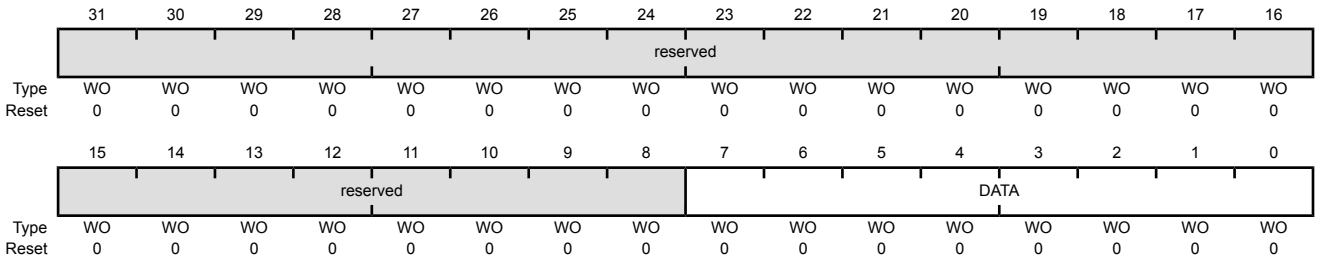
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OE	RO	0	UART Overrun Error When this bit is set to 1, data is received and the FIFO is already full. This bit is cleared to 0 by a write to **UARTECR**. The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
2	BE	RO	0	UART Break Error This bit is set to 1 when a break condition is detected, indicating that the received data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). This bit is cleared to 0 by a write to **UARTECR**. 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.

Bit/Field	Name	Type	Reset	Description
1	PE	RO	0	UART Parity Error This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. This bit is cleared to 0 by a write to **UARTECR**.
0	FE	RO	0	UART Framing Error This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1). This bit is cleared to 0 by a write to **UARTECR**. In FIFO mode, this error is associated with the character at the top of the FIFO.

Writes

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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	WO	0	Error Clear 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.

UART Flag (UARTFR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x018
Type RO, reset 0x0000.0090

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TXFE	RXFF	TXFF	RXFE	BUSY	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	1	0	0	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TXFE	RO	1	UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled (FEN is 0), this bit is set when the transmit holding register is empty. If the FIFO is enabled (FEN is 1), this bit is set when the transmit FIFO is empty.
6	RXFF	RO	0	UART Receive FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, this bit is set when the receive FIFO is full.
5	TXFF	RO	0	UART Transmit FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, this bit is set when the transmit FIFO is full.
4	RXFE	RO	1	UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, this bit is set when the receive FIFO is empty.

Bit/Field	Name	Type	Reset	Description
3	BUSY	RO	0	UART Busy When this bit is 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. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).
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 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register is an 8-bit read/write register that stores the 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 to 0 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:

$$\texttt{ILPDVSR} = \texttt{SysClk} / F_{\texttt{IrLPBaud16}}$$

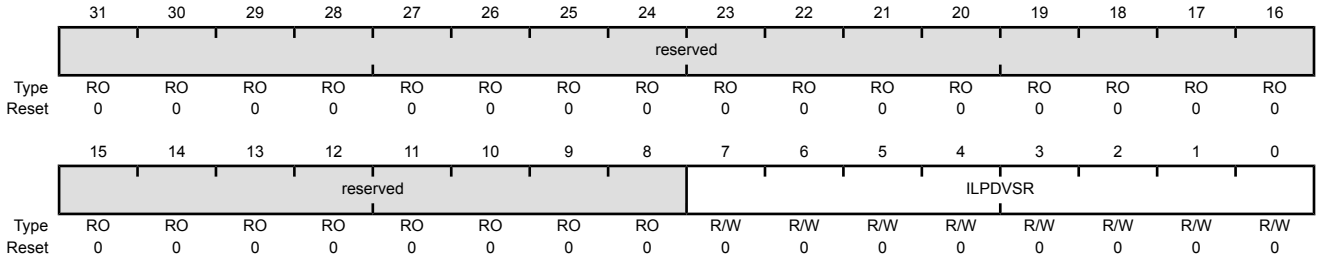
where $F_{\texttt{IrLPBaud16}}$ is nominally 1.8432 MHz.

You must choose the divisor so that 1.42 MHz < $F_{\texttt{IrLPBaud16}}$ < 2.12 MHz, which results 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 that 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



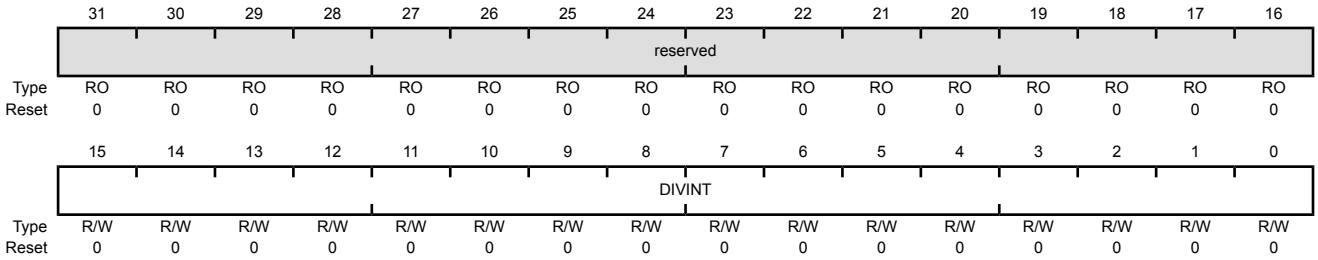
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor This is an 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 433 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



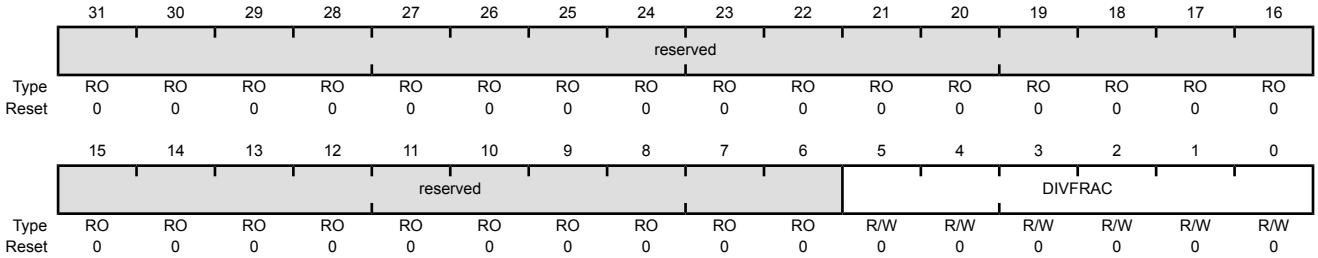
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DIVINT	R/W	0x0000	Integer Baud-Rate Divisor

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 433 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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x000	Fractional Baud-Rate Divisor

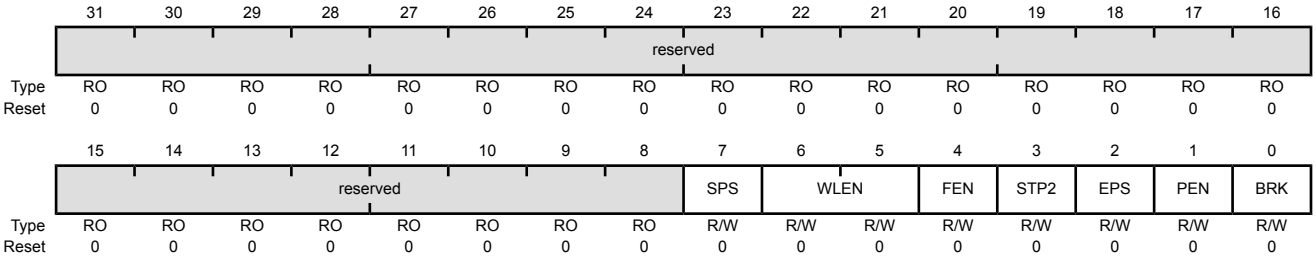
Register 7: UART Line Control (UARTLCRH), offset 0x02C

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	0	Software should not rely on the value of a 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 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. When this bit is cleared, stick parity is disabled.
6:5	WLEN	R/W	0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows: Value Description 0x3 8 bits 0x2 7 bits 0x1 6 bits 0x0 5 bits (default)
4	FEN	R/W	0	UART Enable FIFOs If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.
3	STP2	R/W	0	UART Two Stop Bits Select If this bit is set to 1, two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.

Bit/Field	Name	Type	Reset	Description
2	EPS	R/W	0	UART Even Parity Select If this bit is set to 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. When cleared to 0, then odd parity is performed, which checks for an odd number of 1s. This bit has no effect when parity is disabled by the PEN bit.
1	PEN	R/W	0	UART Parity Enable If this bit is set to 1, parity checking and generation is enabled; otherwise, parity is disabled and no parity bit is added to the data frame.
0	BRK	R/W	0	UART Send Break If this bit is set to 1, a Low level is continually output on the UnTX output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two frames (character periods). For normal use, this bit must be cleared to 0.

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 1.

To enable the UART module, the UARTEN bit must be set to 1. 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: 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 disabling 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

	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						RXE	TXE	LBE	reserved				SIRLP	SIREN	UARTEN
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	RXE	R/W	1	UART Receive Enable If this bit is set to 1, the receive section of the UART is enabled. When the UART is disabled in the middle of a receive, it completes the current character before stopping. **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 If this bit is set to 1, the transmit section of the UART is enabled. When the UART is disabled in the middle of a transmission, it completes the current character before stopping. **Note:** To enable transmission, the UARTEN bit must also be set.
7	LBE	R/W	0	UART Loop Back Enable If this bit is set to 1, the UnTX path is fed through the UnRX path.
6:3	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	SIRLP	R/W	0	UART SIR Low Power Mode This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. If this bit is set to 1, 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. Setting this bit uses less power, but might reduce transmission distances. See page 446 for more information.
1	SIREN	R/W	0	UART SIR Enable If this bit is set to 1, the IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.
0	UARTEN	R/W	0	UART Enable If this bit is set to 1, the UART is enabled. When 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

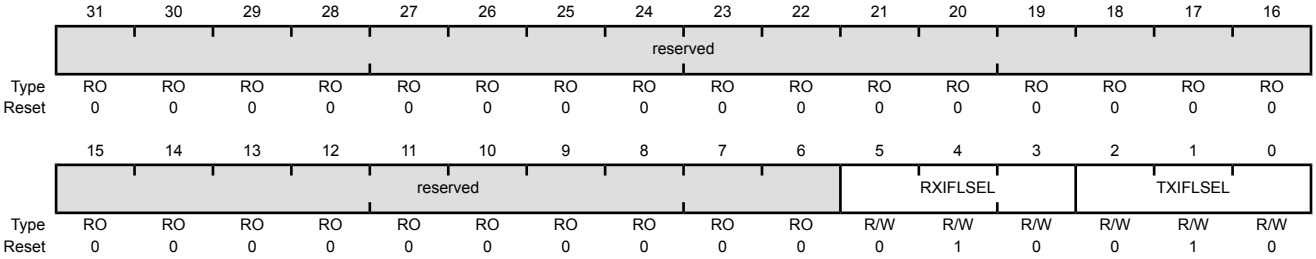
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the **UARTRIS** register are triggered.

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	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.
5:3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows: Value Description 0x0 RX FIFO ≥ ⅛ full 0x1 RX FIFO ≥ ¼ full 0x2 RX FIFO ≥ ½ full (default) 0x3 RX FIFO ≥ ¾ full 0x4 RX FIFO ≥ ⅞ full 0x5-0x7 Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select 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

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. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x038
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					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	reserved			
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask On a read, the current mask for the `OEIM` interrupt is returned. Setting this bit to 1 promotes the `OEIM` interrupt to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask On a read, the current mask for the `BEIM` interrupt is returned. Setting this bit to 1 promotes the `BEIM` interrupt to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask On a read, the current mask for the `PEIM` interrupt is returned. Setting this bit to 1 promotes the `PEIM` interrupt to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask On a read, the current mask for the `FEIM` interrupt is returned. Setting this bit to 1 promotes the `FEIM` interrupt to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask On a read, the current mask for the `RTIM` interrupt is returned. Setting this bit to 1 promotes the `RTIM` interrupt to the interrupt controller.
5	TXIM	R/W	0	UART Transmit Interrupt Mask On a read, the current mask for the `TXIM` interrupt is returned. Setting this bit to 1 promotes the `TXIM` interrupt to the interrupt controller.

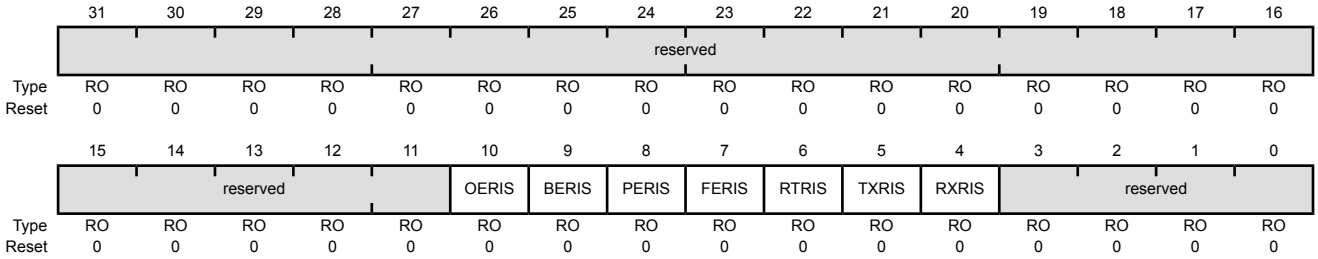
Bit/Field	Name	Type	Reset	Description
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the RXIM interrupt is returned. Setting this bit to 1 promotes the RXIM interrupt to the interrupt controller.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x03C
Type RO, reset 0x0000.0000



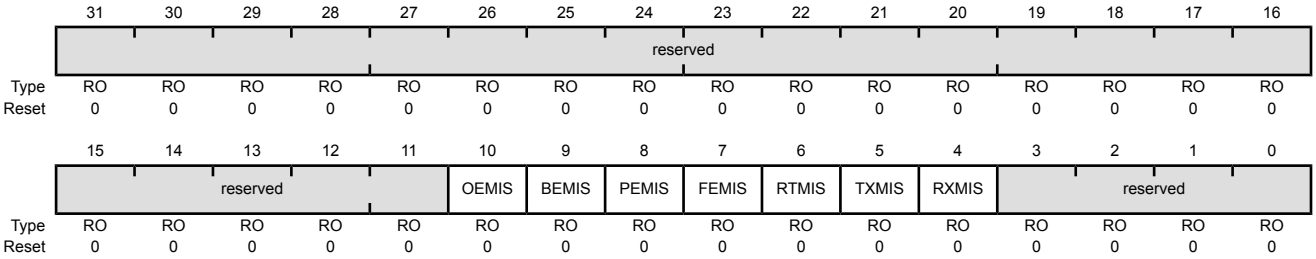
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
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 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x040
Type RO, reset 0x0000.0000



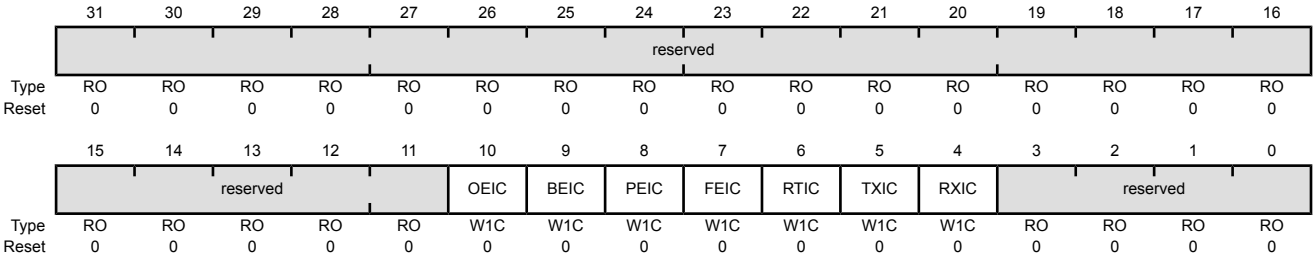
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Gives the masked interrupt state of this interrupt.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Gives the masked interrupt state of this interrupt.
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Gives the masked interrupt state of this interrupt.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 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.

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
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIC	W1C	0	Overrun Error Interrupt Clear The `OEIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
9	BEIC	W1C	0	Break Error Interrupt Clear The `BEIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
8	PEIC	W1C	0	Parity Error Interrupt Clear The `PEIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.

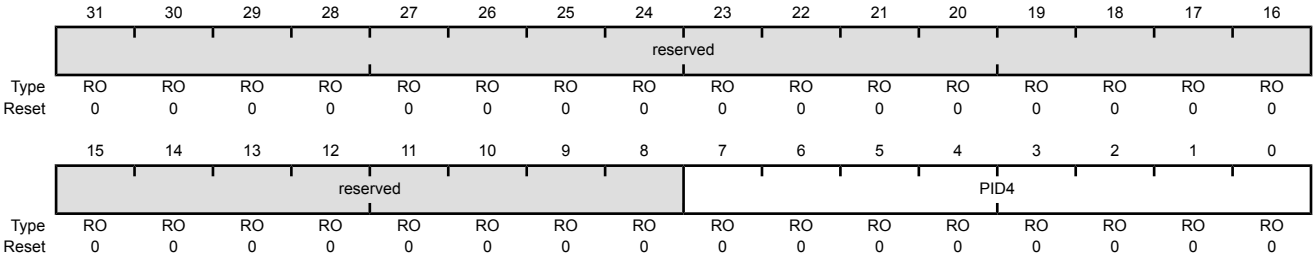
Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear The FEIC values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear The RTIC values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
5	TXIC	W1C	0	Transmit Interrupt Clear The TXIC values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
4	RXIC	W1C	0	Receive Interrupt Clear The RXIC values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
3:0	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 14: 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



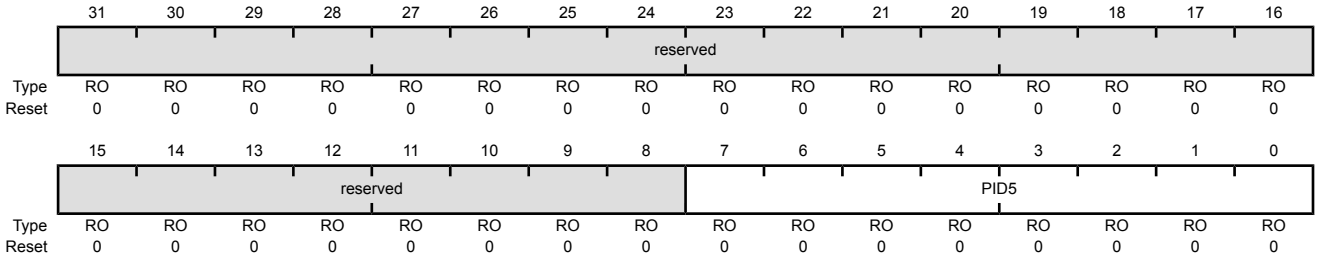
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x0000	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 15: 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



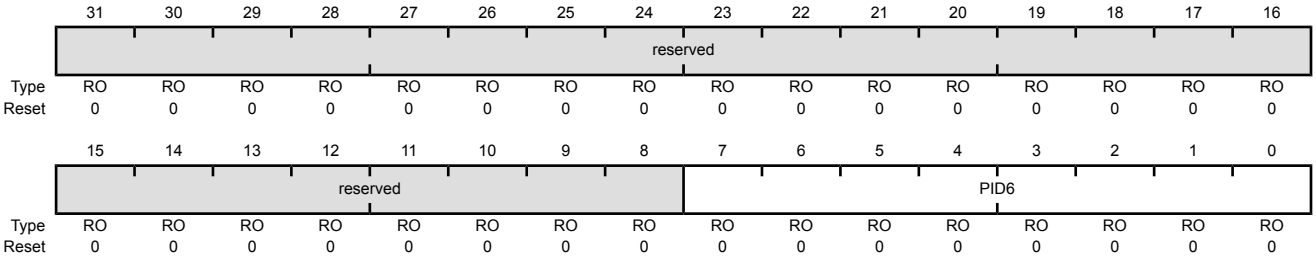
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x0000	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 16: 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



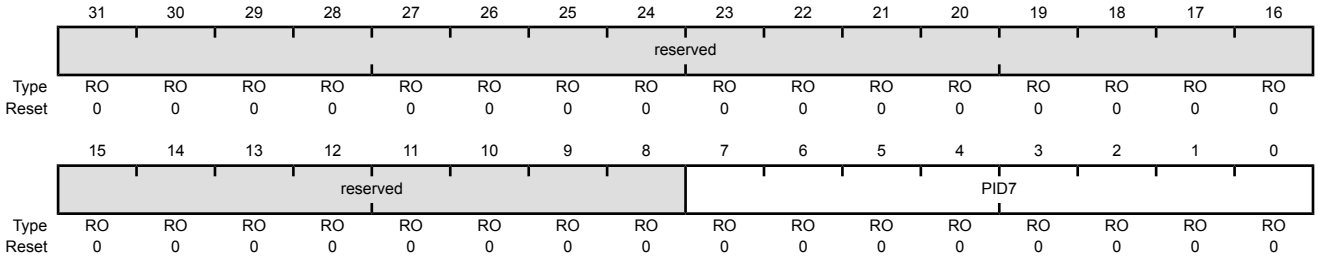
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x0000	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 17: 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



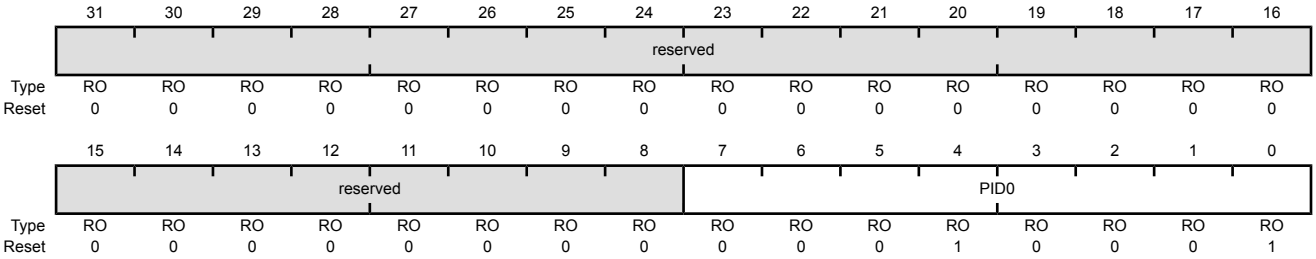
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x0000	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 18: 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 0x0000.0011



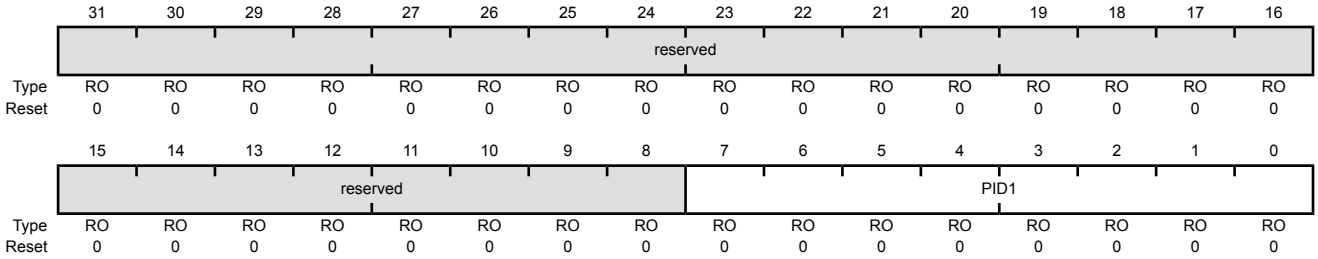
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x11	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 19: 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



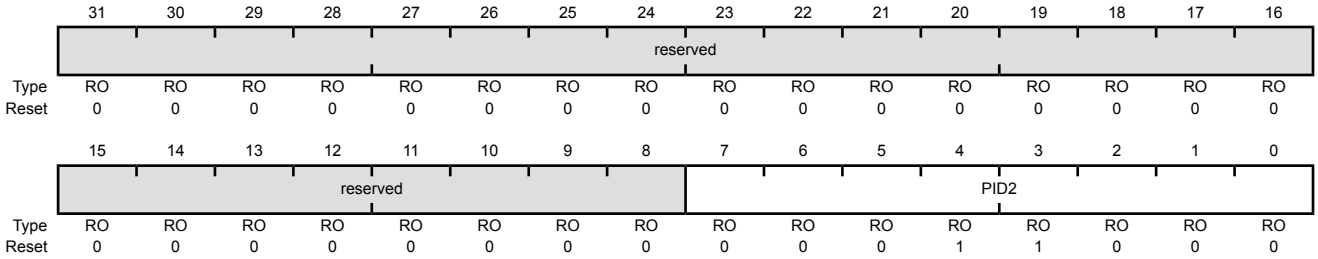
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 20: 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



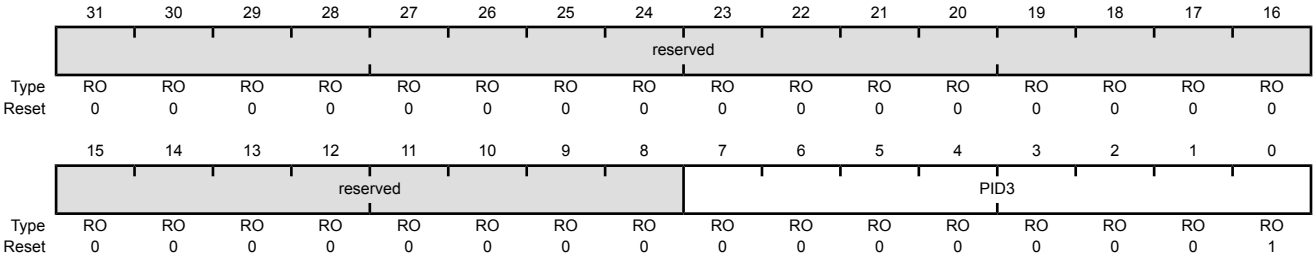
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 21: 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



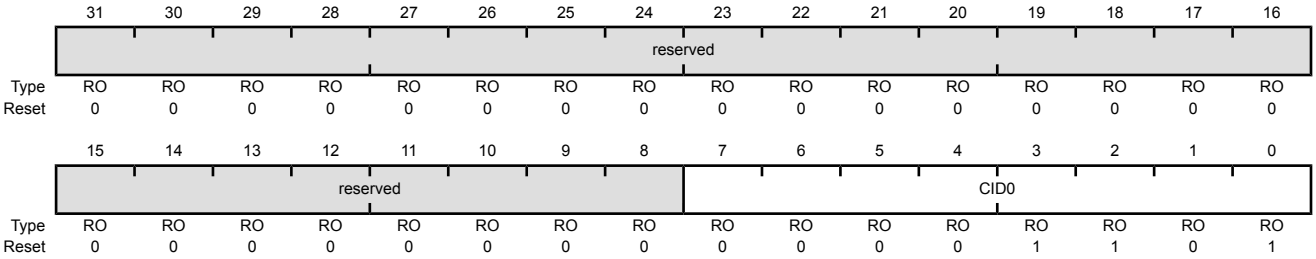
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 22: 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



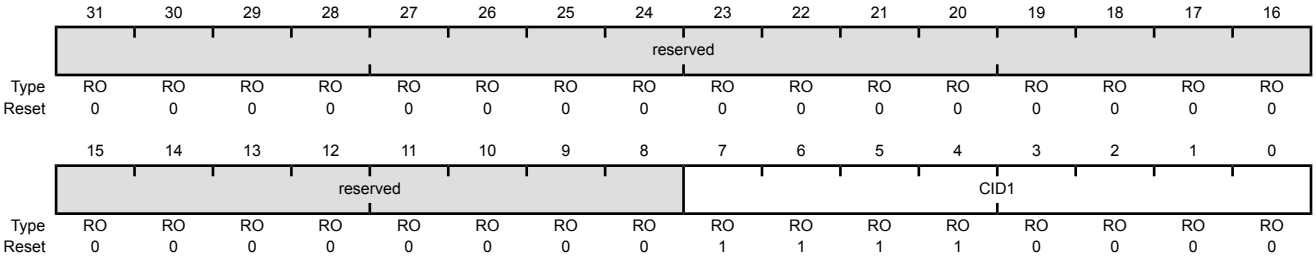
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	UART PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

Register 23: 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



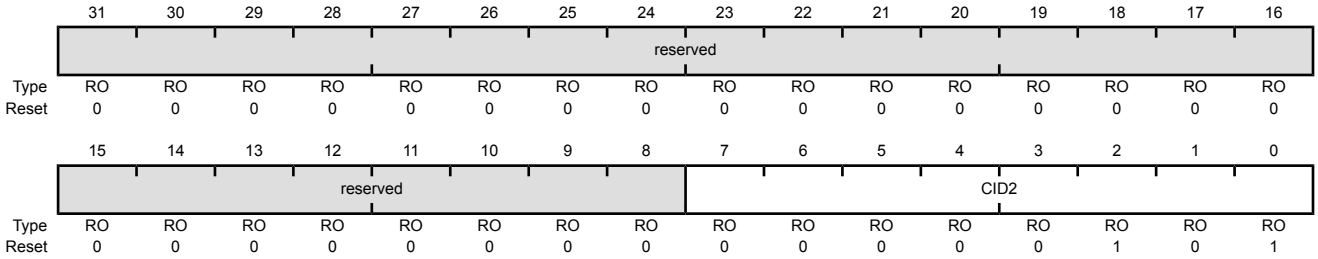
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	UART PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

Register 24: 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



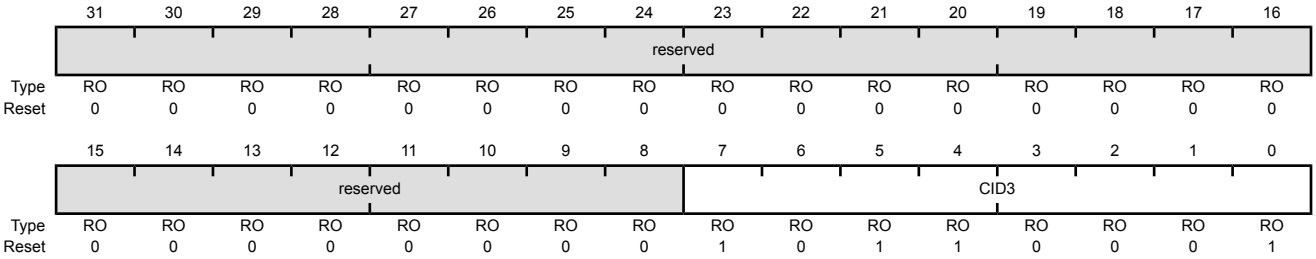
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	UART PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 25: 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	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	UART PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

13 Synchronous Serial Interface (SSI)

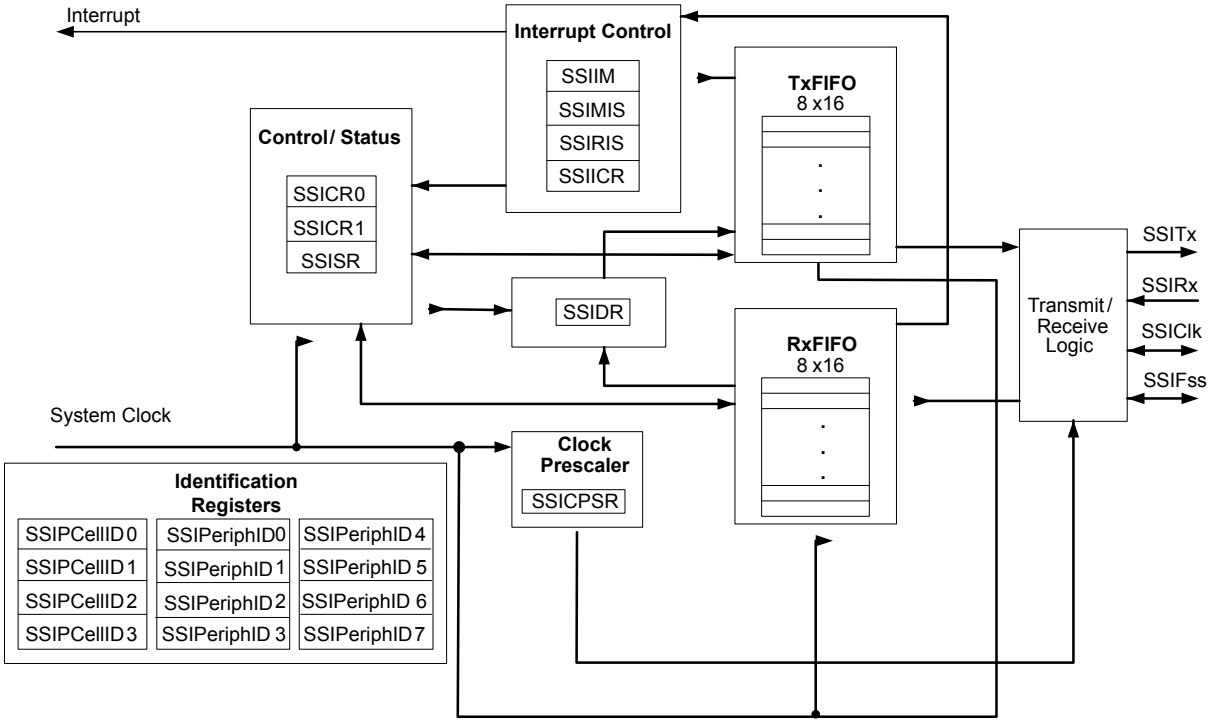
The Stellaris® Synchronous Serial Interface (SSI) is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris SSI module has the following features:

- Master or slave operation
- Programmable clock bit rate and prescale
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing

13.1 Block Diagram

Figure 13-1. SSI Module Block Diagram



13.2 Signal Description

Table 13-1 on page 474 and Table 13-2 on page 474 list the external signals of the SSI module and describe 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 Assignment" lists the possible GPIO pin placements for the SSI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the SSI function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286.

Table 13-1. SSI Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
SSI0Clk	28	I/O	TTL	SSI module 0 clock.
SSI0Fss	29	I/O	TTL	SSI module 0 frame signal.
SSI0Rx	30	I	TTL	SSI module 0 receive.
SSI0Tx	31	O	TTL	SSI module 0 transmit.

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

Table 13-2. SSI Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
SSI0Clk	M4	I/O	TTL	SSI module 0 clock.
SSI0Fss	L4	I/O	TTL	SSI module 0 frame signal.
SSI0Rx	L5	I	TTL	SSI module 0 receive.
SSI0Tx	M5	O	TTL	SSI module 0 transmit.

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

13.3 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.

13.3.1 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 (FSysClk). 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 493). 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 Control0 (SSICR0)** register (see page 486).

The frequency of the output clock SSIClk is defined by:

```
SSIClk = FSysClk / (CPSDVSR * (1 + SCR))
```

Note: For master mode, the system clock must be at least two times faster than the SSIClk. For slave mode, the system clock must be at least 12 times faster than the SSIClk.

See “Synchronous Serial Interface (SSI)” on page 700 to view SSI timing parameters.

13.3.2 FIFO Operation

13.3.2.1 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 490), 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 when the FIFO is empty.

13.3.2.2 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.

13.3.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service
- Receive FIFO time-out
- Receive FIFO overrun

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI can only generate a single interrupt request to the controller at any given time. You can mask each of the four individual maskable interrupts by setting the appropriate bits in the **SSI Interrupt Mask (SSIIM)** register (see page 494). Setting the appropriate mask bit to 1 enables the interrupt.

Provision of the individual outputs, as well as a combined interrupt output, allows 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 496 and page 497, respectively).

13.3.4 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.

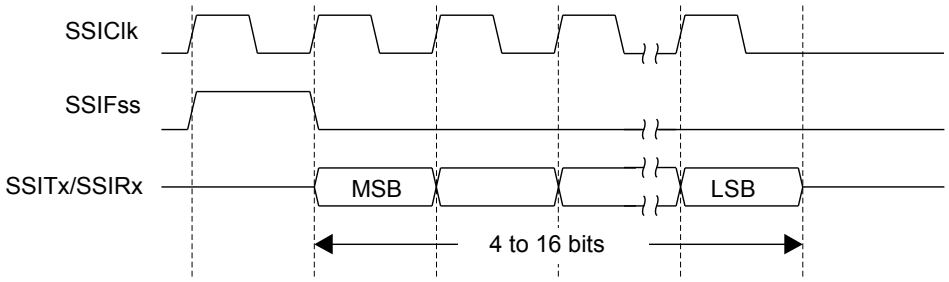
For Texas Instruments synchronous serial frame format, the SSIFss pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of SSIClk, and 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.

13.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 13-2 on page 476 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 13-2. TI Synchronous Serial Frame Format (Single Transfer)

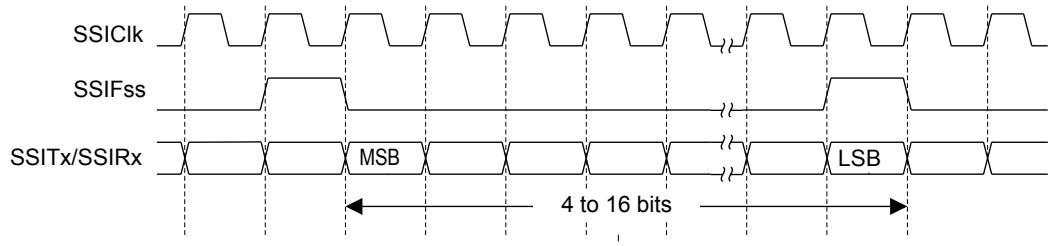


In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the 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 the falling edge of each 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 13-3 on page 477 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 13-3. TI Synchronous Serial Frame Format (Continuous Transfer)



13.3.4.2 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 within the **SSISCR0** control register.

SPO Clock Polarity Bit

When the SPO clock polarity control bit is Low, it produces a steady state Low value on the SSIClk pin. If the SPO bit is High, a steady state High value is placed on the SSIClk pin when data is not being transferred.

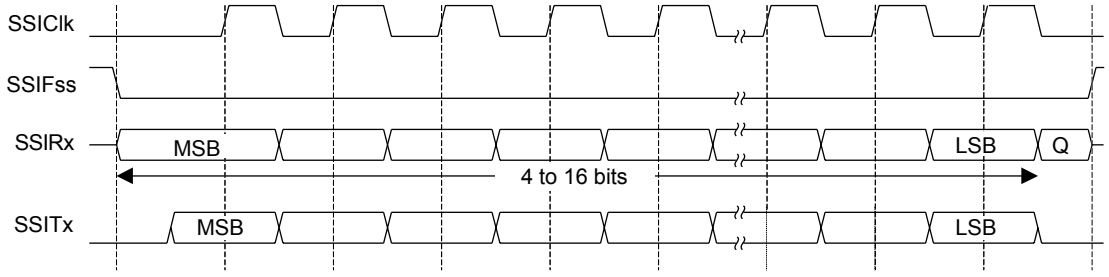
SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data and allows it to change state. It 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 Low, data is captured on the first clock edge transition. If the SPH bit is High, data is captured on the second clock edge transition.

13.3.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

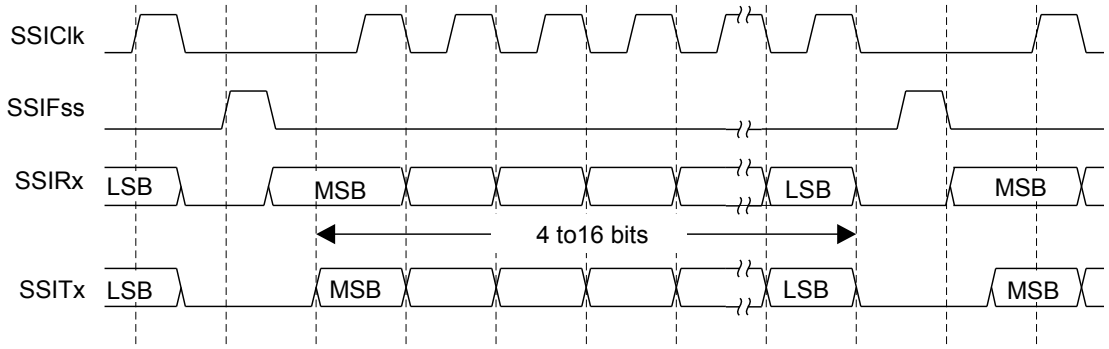
Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 13-4 on page 477 and Figure 13-5 on page 478.

Figure 13-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 13-5. 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 there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. This causes 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. Now that both the master and slave data have been set, the SSIClk master clock pin goes High after one further half SSIClk period.

The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

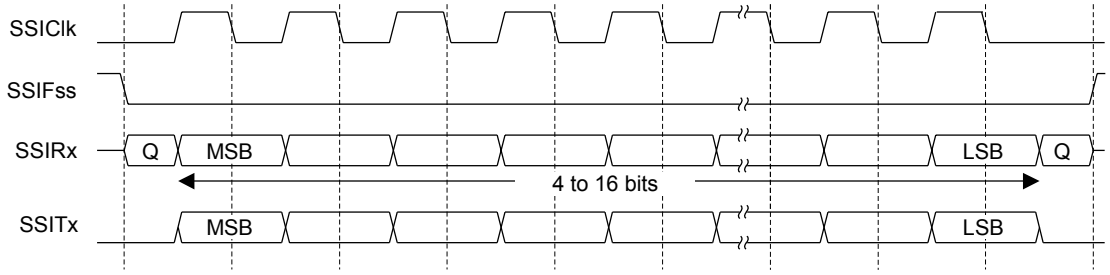
In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is 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 logic zero. 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.

13.3.4.4 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 13-6 on page 479, which covers both single and continuous transfers.

Figure 13-6. 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 there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output is enabled. After a further one half SSIClk period, both master and slave valid data is 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.

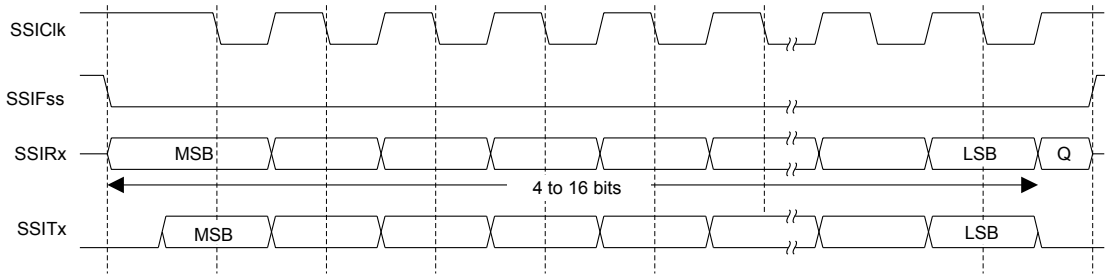
In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

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.

13.3.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

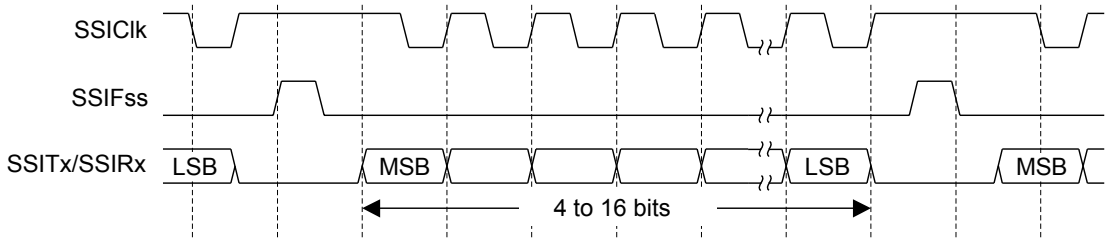
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 13-7 on page 479 and Figure 13-8 on page 480.

Figure 13-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



Note: Q is undefined.

Figure 13-8. 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 there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, which causes 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. Now that both the master and slave data have been set, the SSIClk master clock pin becomes Low after one further half SSIClk period. This means that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

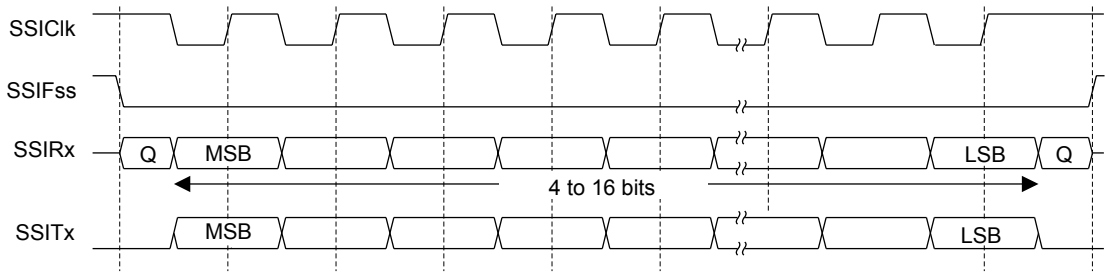
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is 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 logic zero. 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.

13.3.4.6 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 13-9 on page 481, which covers both single and continuous transfers.

Figure 13-9. 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 there is valid data within 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 a further 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.

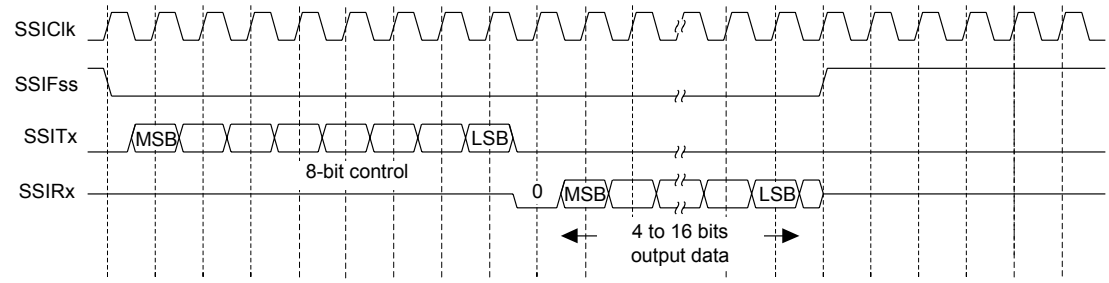
For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

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.

13.3.4.7 MICROWIRE Frame Format

Figure 13-10 on page 481 shows the MICROWIRE frame format, again for a single frame. Figure 13-11 on page 482 shows the same format when back-to-back frames are transmitted.

Figure 13-10. MICROWIRE Frame Format (Single Frame)



MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using 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

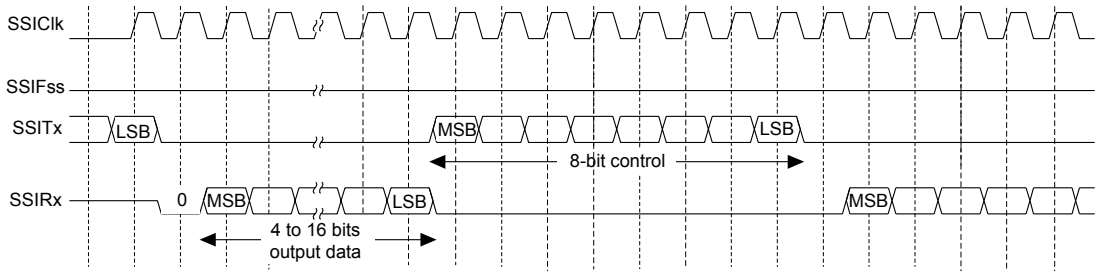
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto 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 the rising edge of each 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, which causes 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.

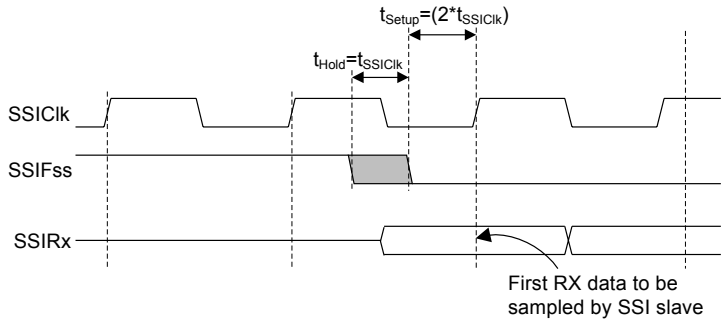
Figure 13-11. MICROWIRE Frame Format (Continuous Transfer)



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 13-12 on page 483 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 13-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements



13.4 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the SSI bit in the **RCGC1** register.

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 disabled 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. 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:

```
FSSIClk = FSysClk / (CPSDVSR * (1 + SCR))
1x10^6 = 20x10^6 / (CPSDVSR * (1 + SCR))
```

In this case, if CPSDVSR=2, SCR must be 9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the **SSICR1** register is disabled.
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 to 1.

13.5 Register Map

Table 13-3 on page 484 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

Note that the SSI module clock must be enabled before the registers can be programmed (see page 219). 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 13-3. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	486
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	488
0x008	SSIDR	R/W	0x0000.0000	SSI Data	490
0x00C	SSISR	RO	0x0000.0003	SSI Status	491
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	493
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	494
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	496
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	497

Table 13-3. SSI Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	498
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	499
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	500
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	501
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	502
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	503
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	504
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	505
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	506
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	507
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	508
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	509
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	510

13.6 Register Descriptions

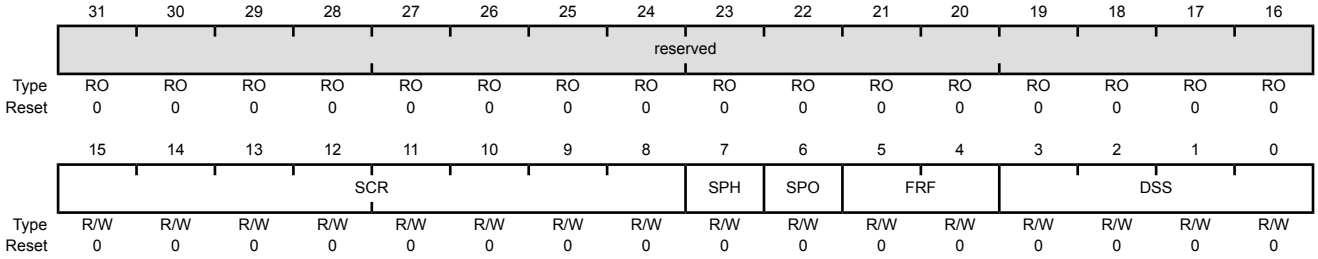
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

Register 1: SSI Control 0 (SSICR0), offset 0x000

SSICR0 is control register 0 and 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
Offset 0x000
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	R/W	0x0000	SSI Serial Clock Rate The value SCR is used to generate the transmit and receive bit rate of the SSI. The bit rate is: `BR=FSSIClk/(CPSDVSR * (1 + SCR))` 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 This bit is only applicable to the Freescale SPI Format. The SPH control bit selects the clock edge that captures data and allows it to change state. It 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 bit is 0, data is captured on the first clock edge transition. If SPH is 1, data is captured on the second clock edge transition.
6	SPO	R/W	0	SSI Serial Clock Polarity This bit is only applicable to the Freescale SPI Format. When the SPO bit is 0, it produces a steady state Low value on the SSIClk pin. If SPO is 1, a steady state High value is placed on the SSIClk pin when data is not being transferred.
5:4	FRF	R/W	0x0	SSI Frame Format Select The FRF values are defined as follows: 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	0x00	SSI Data Size Select The DSS values are defined as follows:

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

SSICR1 is control register 1 and 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
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												SOD	MS	SSE	LBM
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	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.
3	SOD	R/W	0	SSI Slave Mode Output Disable 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. The SOD values are defined as follows: Value Description 0 SSI can drive SSITx output in Slave Output mode. 1 SSI must not drive the SSITx output in Slave mode.
2	MS	R/W	0	SSI Master/Slave Select This bit selects Master or Slave mode and can be modified only when SSI is disabled (SSE=0). The MS values are defined as follows: Value Description 0 Device configured as a master. 1 Device configured as a slave.

Bit/Field	Name	Type	Reset	Description
1	SSE	R/W	0	SSI Synchronous Serial Port Enable Setting this bit enables SSI operation. The SSE values are defined as follows: Value Description 0 SSI operation disabled. 1 SSI operation enabled. **Note:** This bit must be set to 0 before any control registers are reprogrammed.
0	LBM	R/W	0	SSI Loopback Mode Setting this bit enables Loopback Test mode. The LBM values are defined as follows: Value Description 0 Normal serial port operation enabled. 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.

SSIDR is the data register and is 16-bits wide. When **SSIDR** is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSI receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

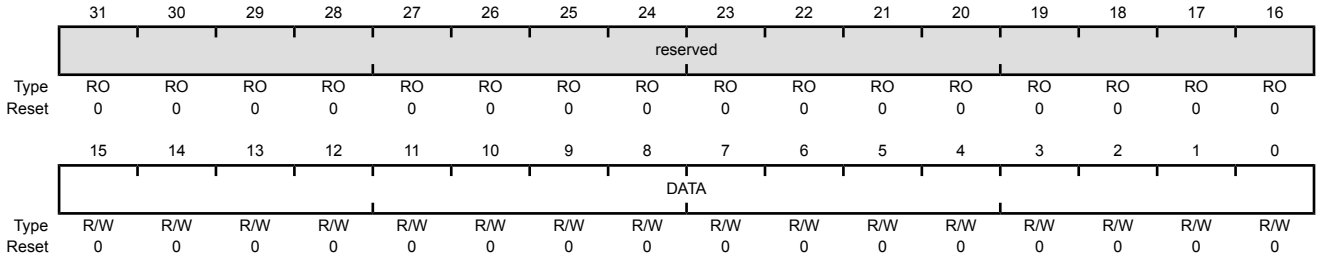
When **SSIDR** is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the SSITx pin at the programmed 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 set to zero. This allows the software to fill the transmit FIFO before enabling the SSI.

SSI Data (SSIDR)

SSI0 base: 0x4000.8000
Offset 0x008
Type R/W, reset 0x0000.0000



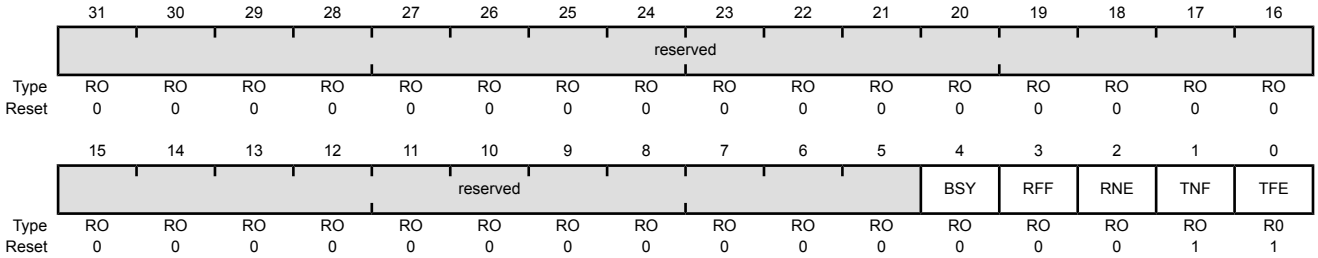
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	SSI Receive/Transmit Data A read operation reads the receive FIFO. A write operation writes the transmit FIFO. 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

SSISR is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000
Offset 0x00C
Type RO, reset 0x0000.0003



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BSY	RO	0	SSI Busy Bit The BSY values are defined as follows: Value Description 0 SSI is idle. 1 SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	RO	0	SSI Receive FIFO Full The RFF values are defined as follows: Value Description 0 Receive FIFO is not full. 1 Receive FIFO is full.
2	RNE	RO	0	SSI Receive FIFO Not Empty The RNE values are defined as follows: Value Description 0 Receive FIFO is empty. 1 Receive FIFO is not empty.
1	TNF	RO	1	SSI Transmit FIFO Not Full The TNF values are defined as follows: Value Description 0 Transmit FIFO is full. 1 Transmit FIFO is not full.

Bit/Field	Name	Type	Reset	Description
0	TFE	R0	1	SSI Transmit FIFO Empty The TFE values are defined as follows: Value Description 0 Transmit FIFO is not empty. 1 Transmit FIFO is empty.

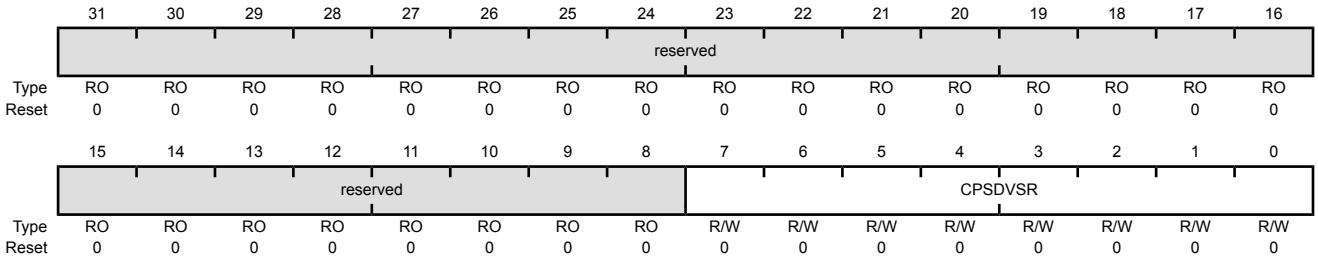
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

SSICPSR is the clock prescale register and specifies the division factor by which the system clock must be internally divided before further use.

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
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 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 to 0 on reset.

On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
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	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.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask The `TXIM` values are defined as follows: Value Description 0 TX FIFO half-empty or less condition interrupt is masked. 1 TX FIFO half-empty or less condition interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask The `RXIM` values are defined as follows: Value Description 0 RX FIFO half-full or more condition interrupt is masked. 1 RX FIFO half-full or more condition interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask The `RTIM` values are defined as follows: Value Description 0 RX FIFO time-out interrupt is masked. 1 RX FIFO time-out interrupt is not masked.

Bit/Field	Name	Type	Reset	Description
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask The RORIM values are defined as follows: Value Description 0 RX FIFO overrun interrupt is masked. 1 RX 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
Offset 0x018
Type RO, reset 0x0000.0008

	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												TXRIS	RXRIS	RTRIS	RORRIS
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	0	0	0

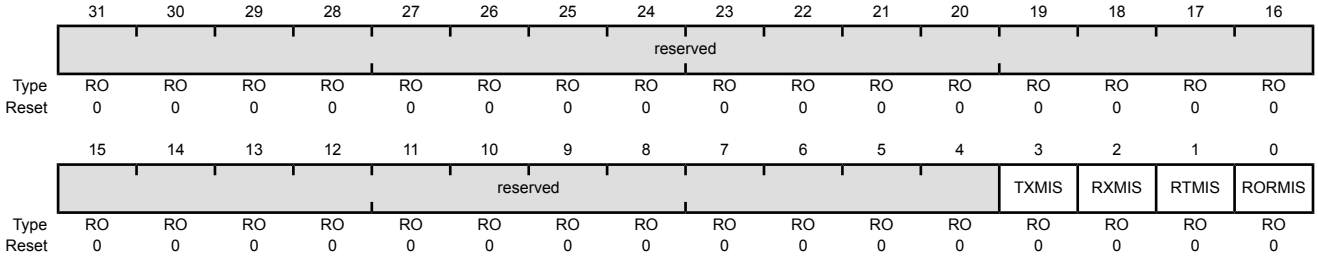
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Indicates that the transmit FIFO is half empty or less, when set.
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status Indicates that the receive FIFO has overflowed, when set.

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
Offset 0x01C
Type RO, reset 0x0000.0000



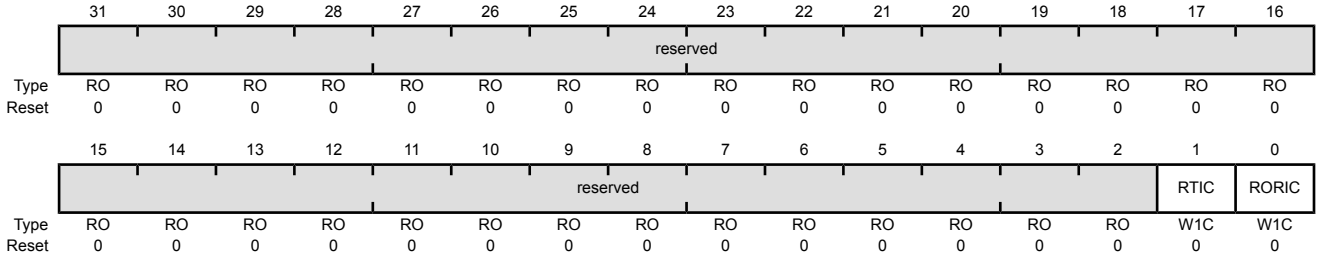
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Indicates that the transmit FIFO is half empty or less, when set.
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Indicates that the receive FIFO has overflowed, when set.

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
Offset 0x020
Type W1C, reset 0x0000.0000



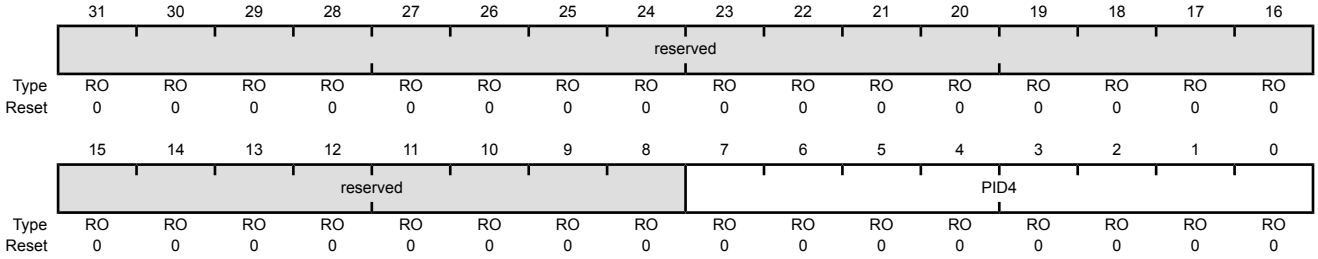
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear The `RTIC` values are defined as follows: Value Description 0 No effect on interrupt. 1 Clears interrupt.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear The `RORIC` values are defined as follows: Value Description 0 No effect on interrupt. 1 Clears interrupt.

Register 10: 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
Offset 0xFD0
Type RO, reset 0x0000.0000



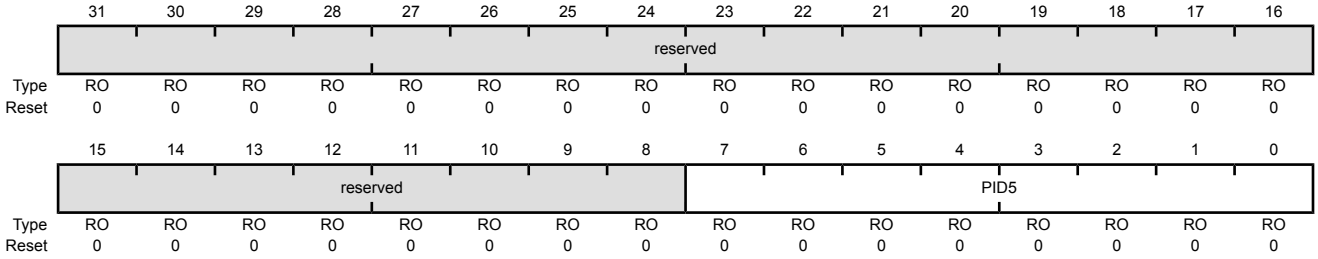
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 11: 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
Offset 0xFD4
Type RO, reset 0x0000.0000



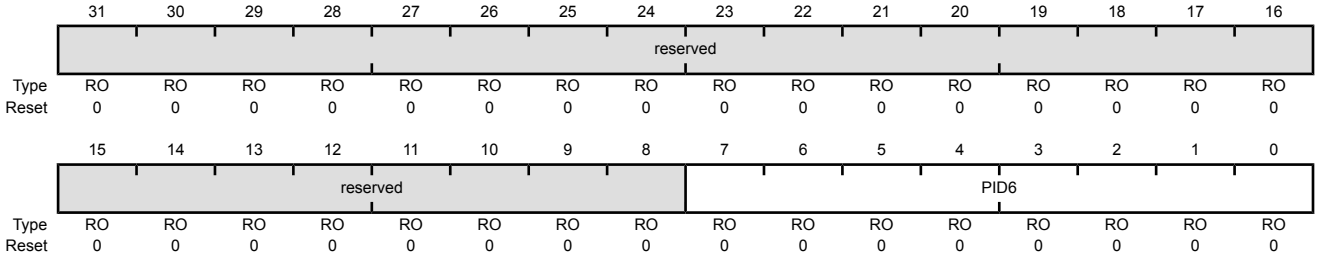
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 12: 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
Offset 0xFD8
Type RO, reset 0x0000.0000



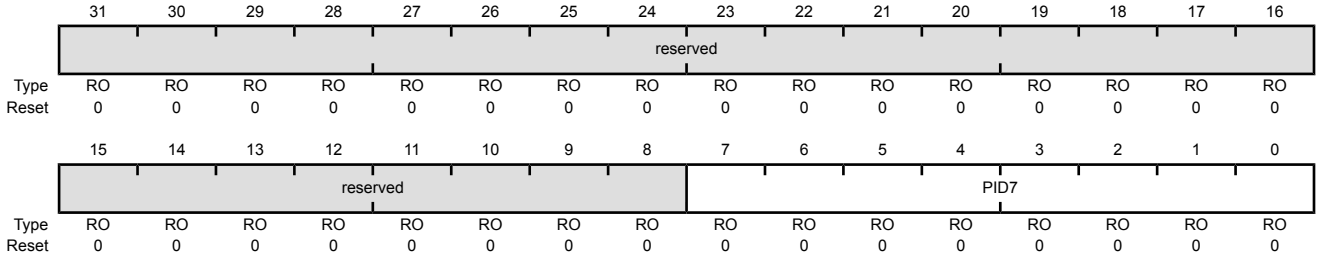
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	SSI Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 13: 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
Offset 0xFDC
Type RO, reset 0x0000.0000



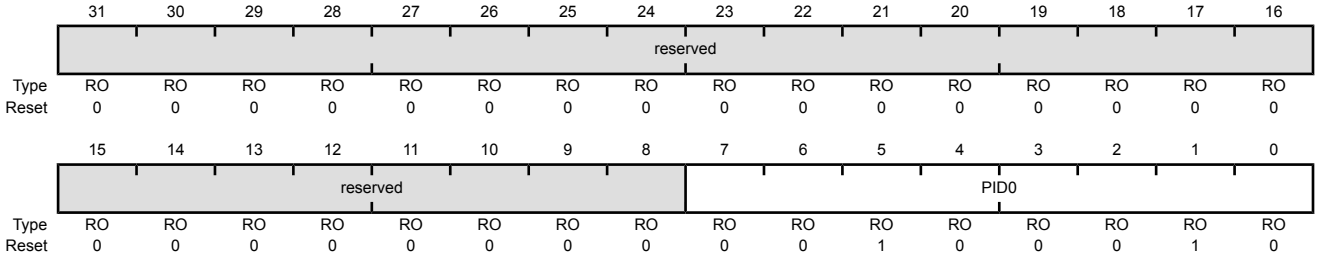
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	SSI Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 14: 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
Offset 0xFE0
Type RO, reset 0x0000.0022



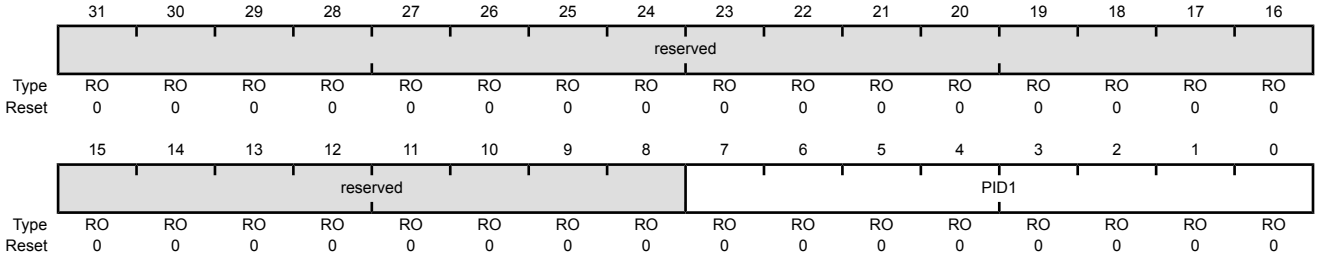
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 15: 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
Offset 0xFE4
Type RO, reset 0x0000.0000



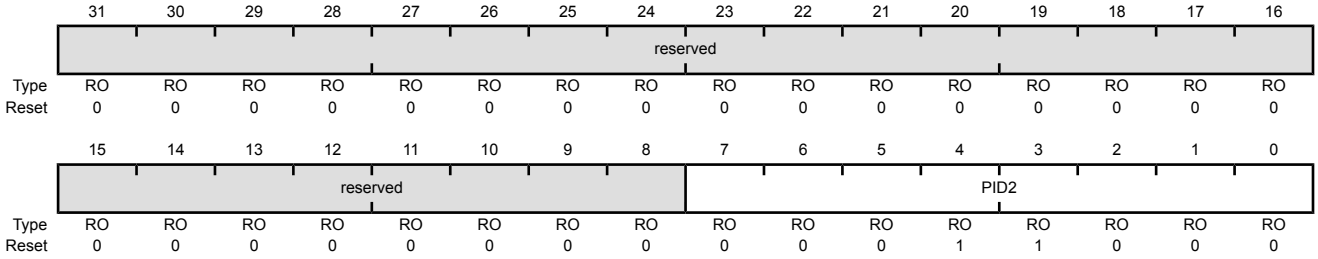
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 16: 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
Offset 0xFE8
Type RO, reset 0x0000.0018



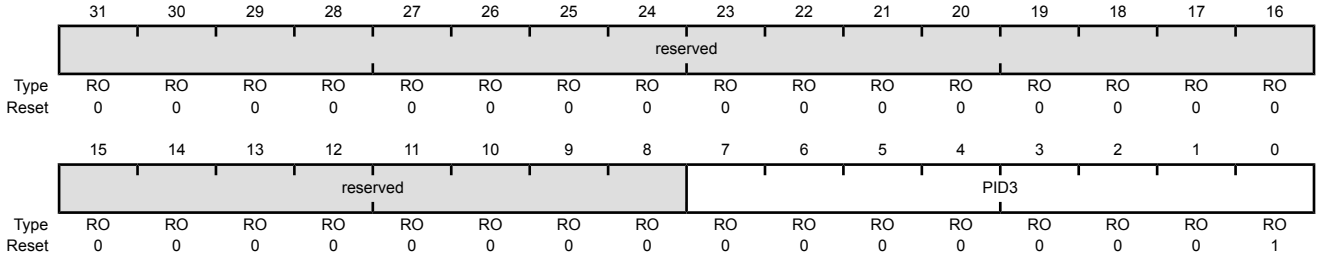
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 17: 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
Offset 0xFEC
Type RO, reset 0x0000.0001



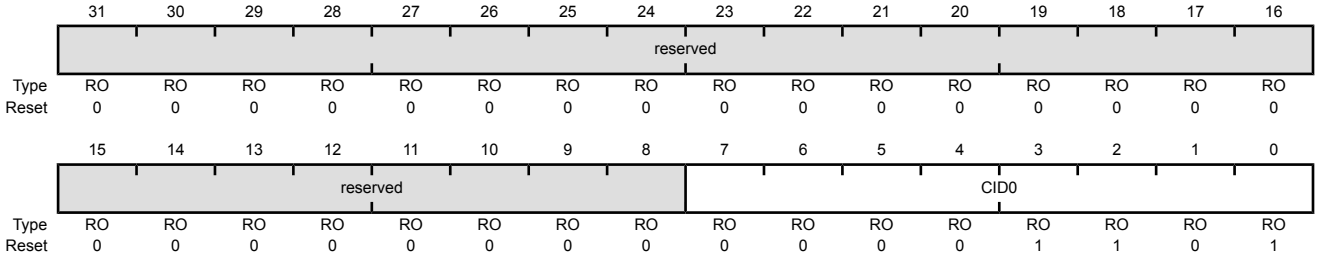
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 18: 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
Offset 0xFF0
Type RO, reset 0x0000.000D



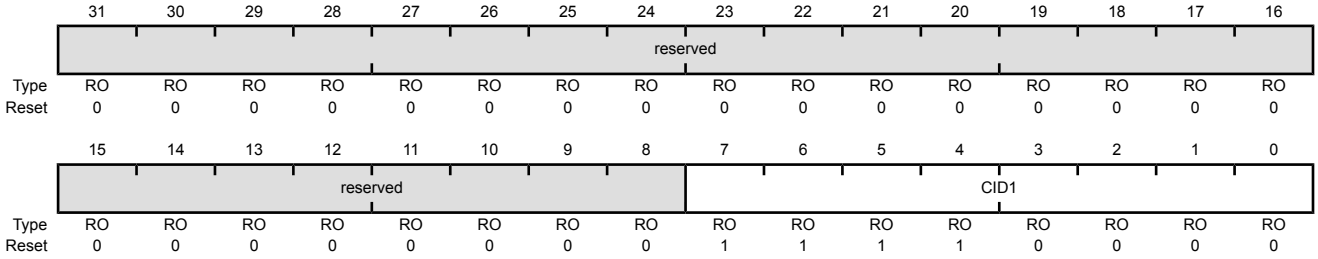
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 19: 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
Offset 0xFF4
Type RO, reset 0x0000.00F0



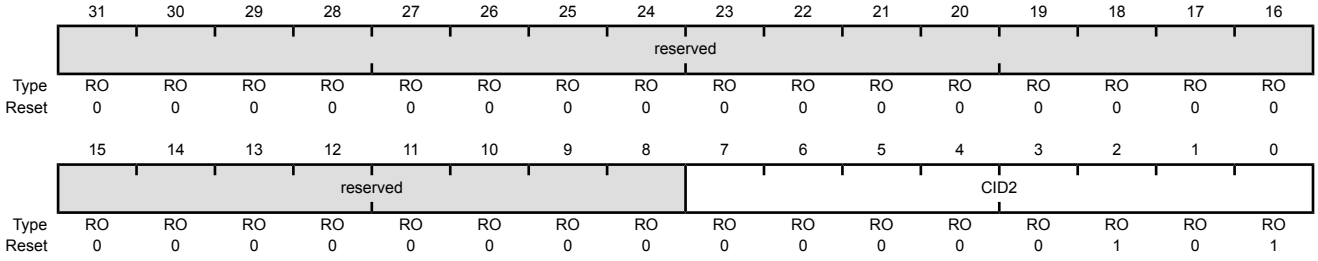
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	SSI PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 20: 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
Offset 0xFF8
Type RO, reset 0x0000.0005



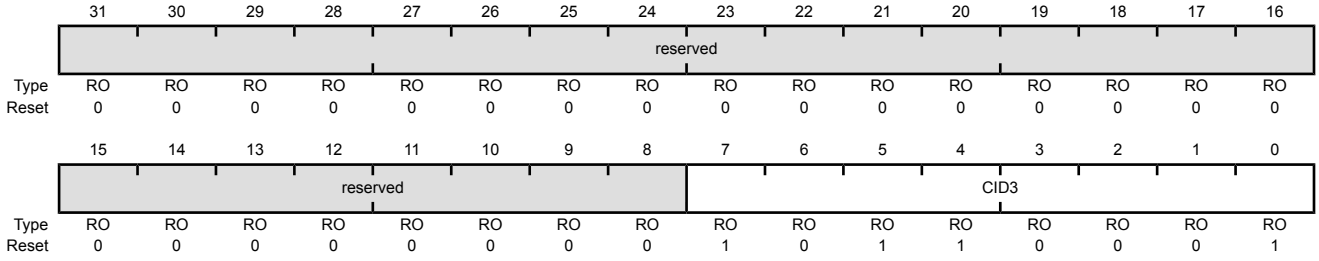
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 21: 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
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	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

14 Inter-Integrated Circuit (I^2C) Interface

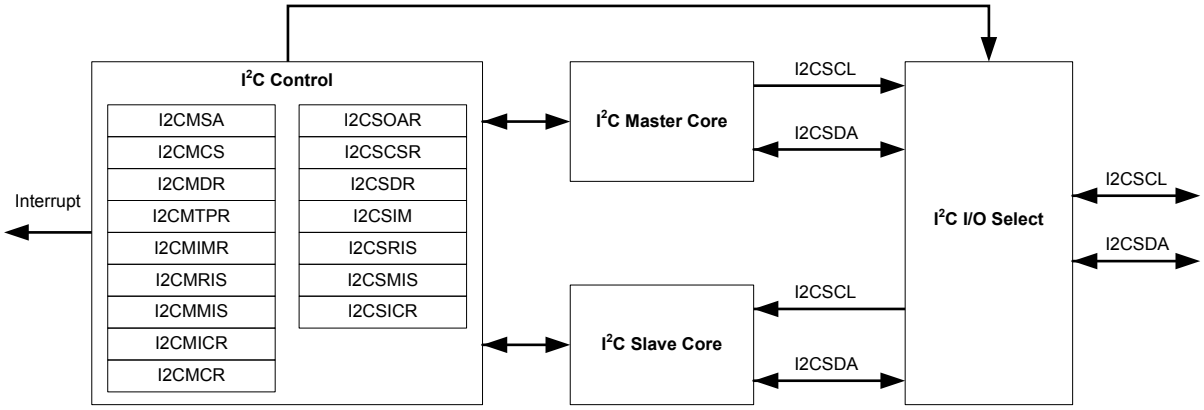
The Inter-Integrated Circuit (I^2C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I^2C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^2C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S8938 microcontroller includes two I^2C modules, providing the ability to interact (both send and receive) with other I^2C devices on the bus.

The Stellaris® I^2C interface has the following features:

- Two I^2C modules, each with the following features:
- Devices on the I^2C bus can be designated as either a master or a slave
  - Supports both sending 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 sent or requested by a master
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

14.1 Block Diagram

Figure 14-1. I2C Block Diagram



14.2 Signal Description

Table 14-1 on page 512 and Table 14-2 on page 512 list the external signals of the I2C interface and describe 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 Assignment" lists the possible GPIO pin placements for the I2C signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the I2C function. 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 286.

Table 14-1. I2C Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
I2C0SCL	70	I/O	OD	I2C module 0 clock.
I2C0SDA	71	I/O	OD	I2C module 0 data.
I2C1SCL	34	I/O	OD	I2C module 1 clock.
I2C1SDA	35	I/O	OD	I2C module 1 data.

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

Table 14-2. I2C Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
I2C0SCL	C11	I/O	OD	I2C module 0 clock.
I2C0SDA	C12	I/O	OD	I2C module 0 data.
I2C1SCL	L6	I/O	OD	I2C module 1 clock.
I2C1SDA	M6	I/O	OD	I2C module 1 data.

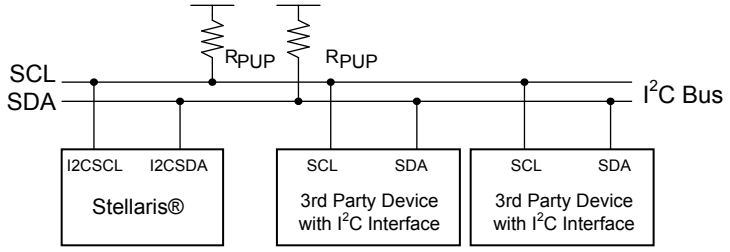
a. The TTL designation indicates the pin has TTL-compatible voltage levels.

14.3 Functional Description

Each I²C module is comprised of both master and slave functions which are implemented as separate peripherals. For proper operation, the SDA and SCL pins must be connected to bi-directional open-drain pads. A typical I²C bus configuration is shown in Figure 14-2 on page 513.

See “Inter-Integrated Circuit (I²C) Interface” on page 702 for I²C timing diagrams.

Figure 14-2. I²C Bus Configuration



14.3.1 I²C Bus Functional Overview

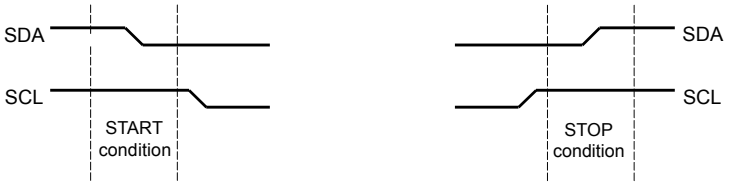
The I²C bus uses only two signals: SDA and SCL, named `I2CSDA` and `I2CSCL` on Stellaris microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

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 513) 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.

14.3.1.1 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 14-3 on page 513.

Figure 14-3. START and STOP Conditions

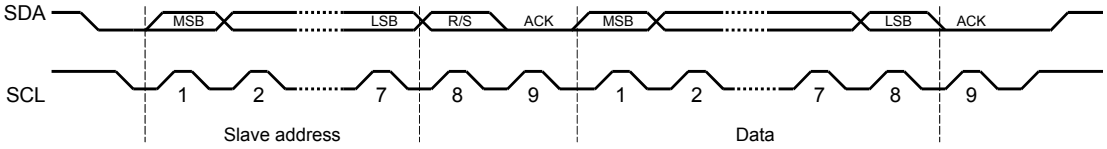


14.3.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 14-4 on page 514. After the START condition, a slave address is sent. This address is 7-bits long followed by an eighth bit, which is a data direction

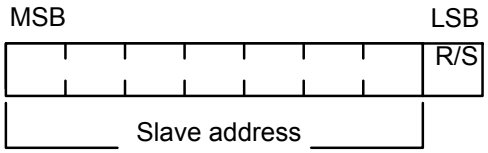
bit (R/S bit in the **I2CMSA** register). A zero indicates a transmit operation (send), and a one indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/send formats are then possible within a single transfer.

Figure 14-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 14-5 on page 514). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master will write (send) data to the selected slave, and a one in this position means that the master will receive data from the slave.

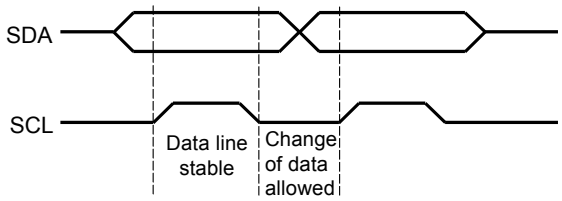
Figure 14-5. R/S Bit in First Byte



14.3.1.3 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 14-6 on page 514).

Figure 14-6. Data Validity During Bit Transfer on the I2C Bus



14.3.1.4 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 sent out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 514.

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. Since 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.

14.3.1.5 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) will switch off its data output stage and retire 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.

14.3.2 Available Speed Modes

The I^2C 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 **I^2C Master Timer Period (I2CMTPR)** register (see page 533).

The I^2C 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/T = 333 Khz
```

Table 14-3 on page 515 gives examples of timer period, system clock, and speed mode (Standard or Fast).

Table 14-3. Examples of I^2C 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

Table 14-3. Examples of I2C Master Timer Period versus Speed Mode *(continued)*

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
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

14.3.3 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

There is a separate interrupt signal for the I2C master and I2C slave modules. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

14.3.3.1 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.

14.3.3.2 I2C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by writing a 1 to 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 writing a 1 to the DATAIC bit in 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.

14.3.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished 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.

14.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I²C transfer types in both master and slave mode.

14.3.5.1 I²C Master Command Sequences

The figures that follow show the command sequences available for the I²C master.

Figure 14-7. Master Single SEND

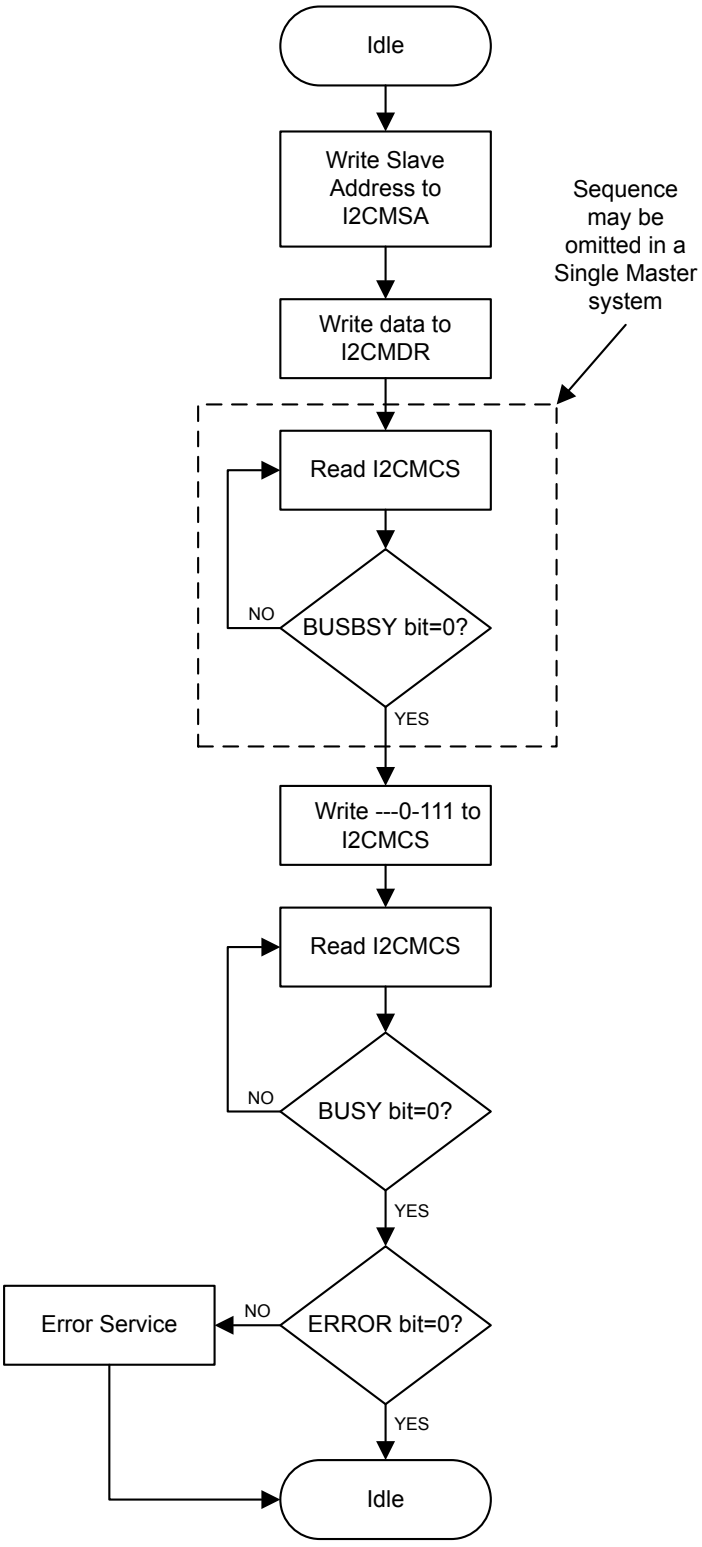


Figure 14-8. Master Single RECEIVE

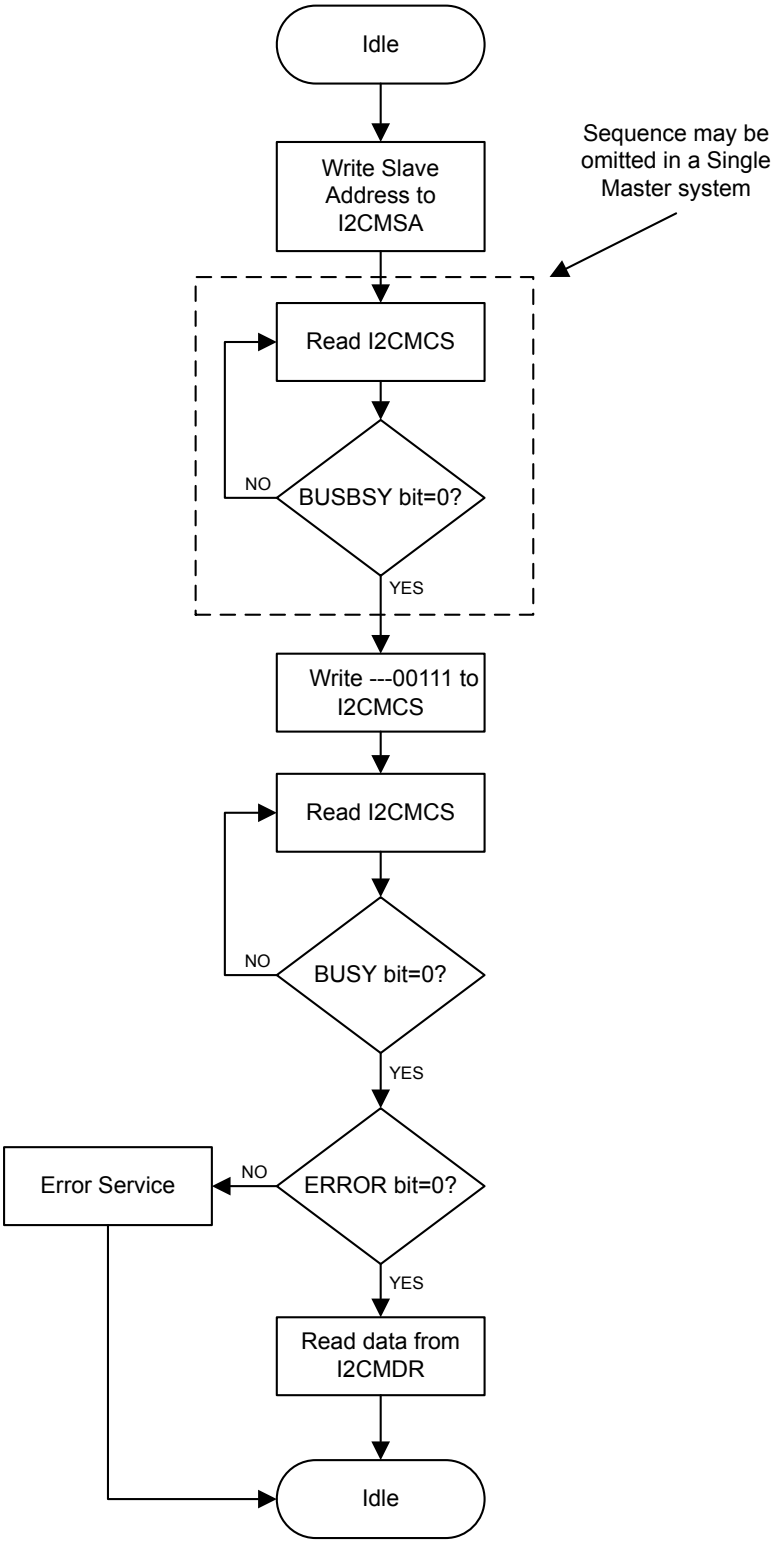


Figure 14-9. Master Burst SEND

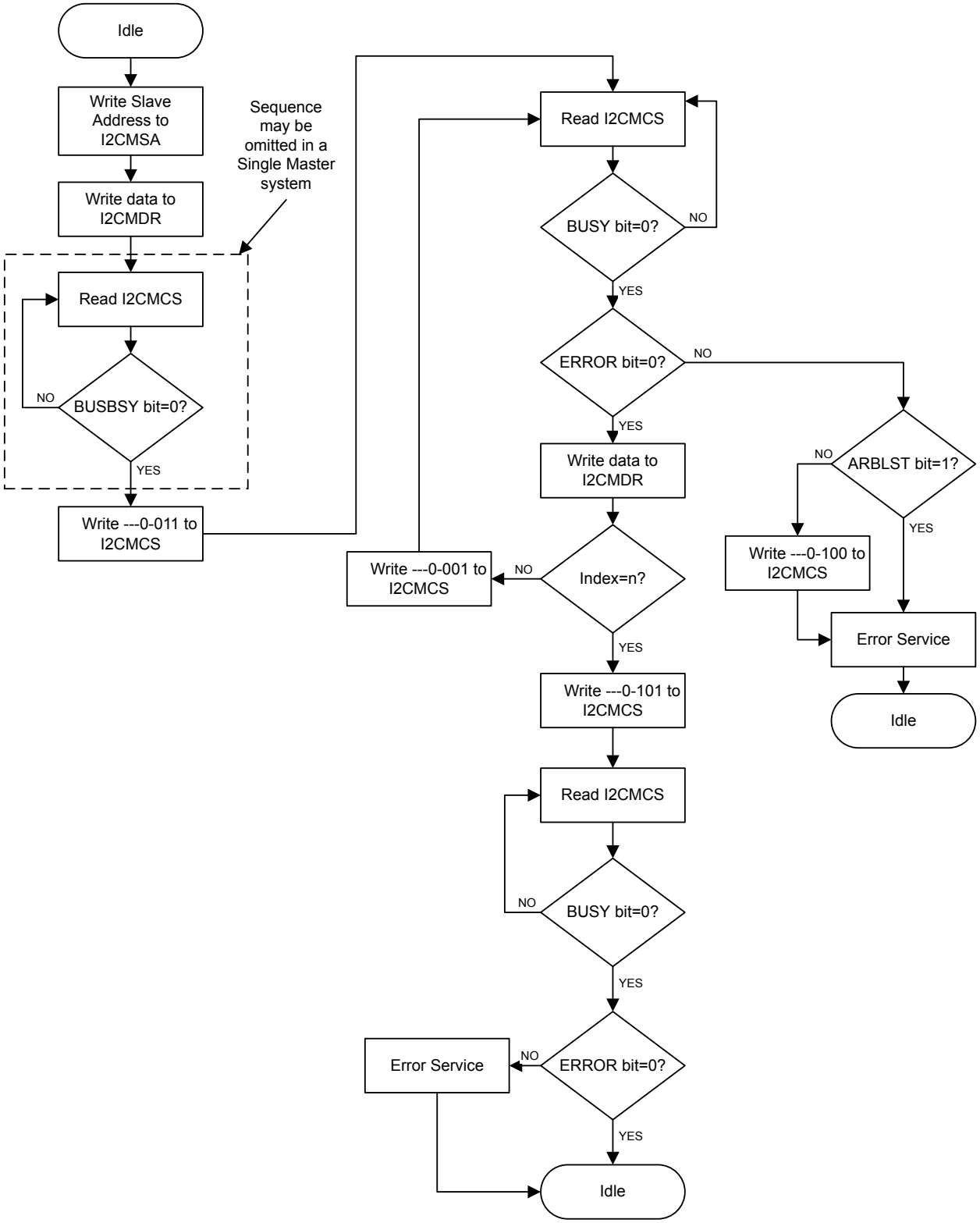


Figure 14-10. Master Burst RECEIVE

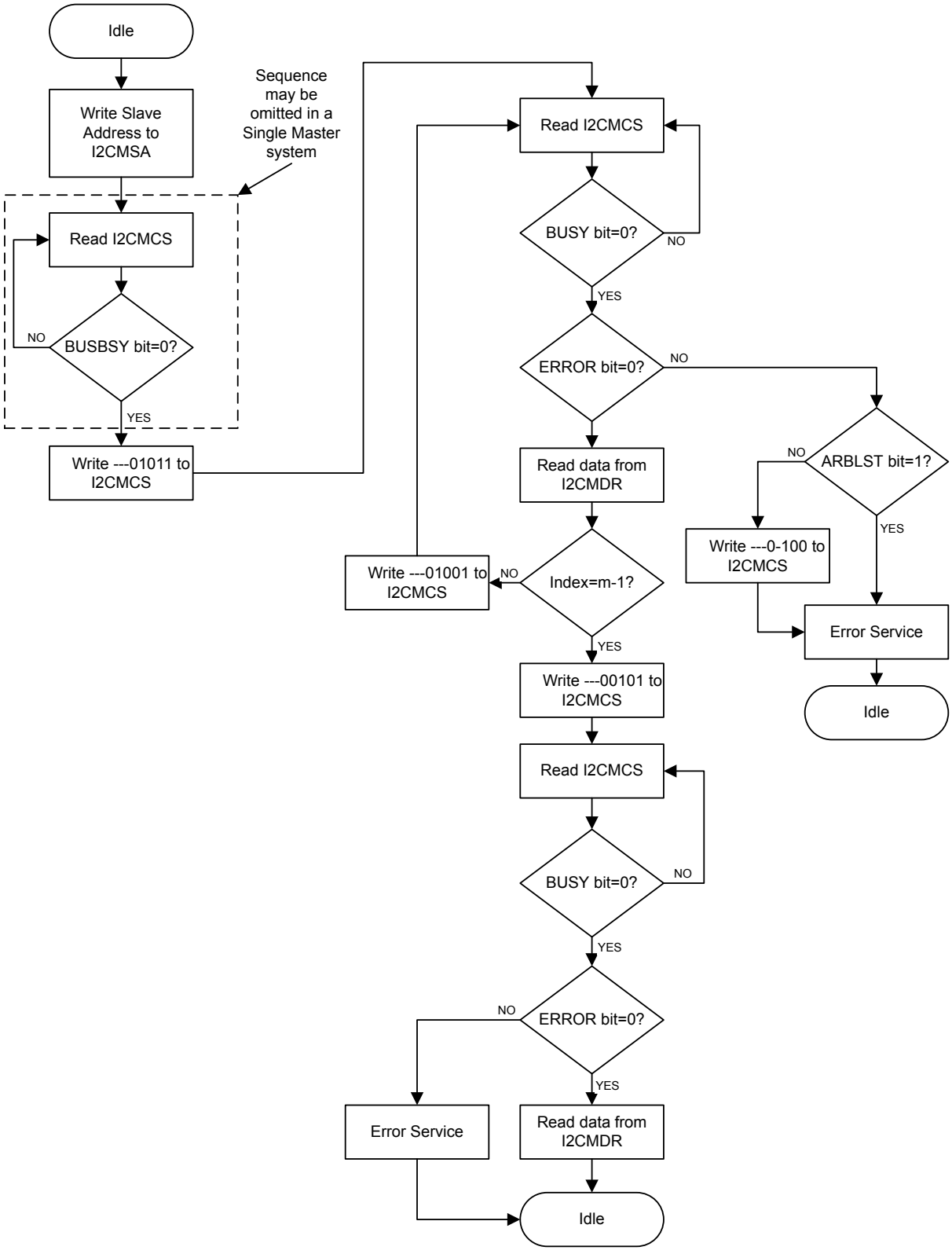


Figure 14-11. Master Burst RECEIVE after Burst SEND

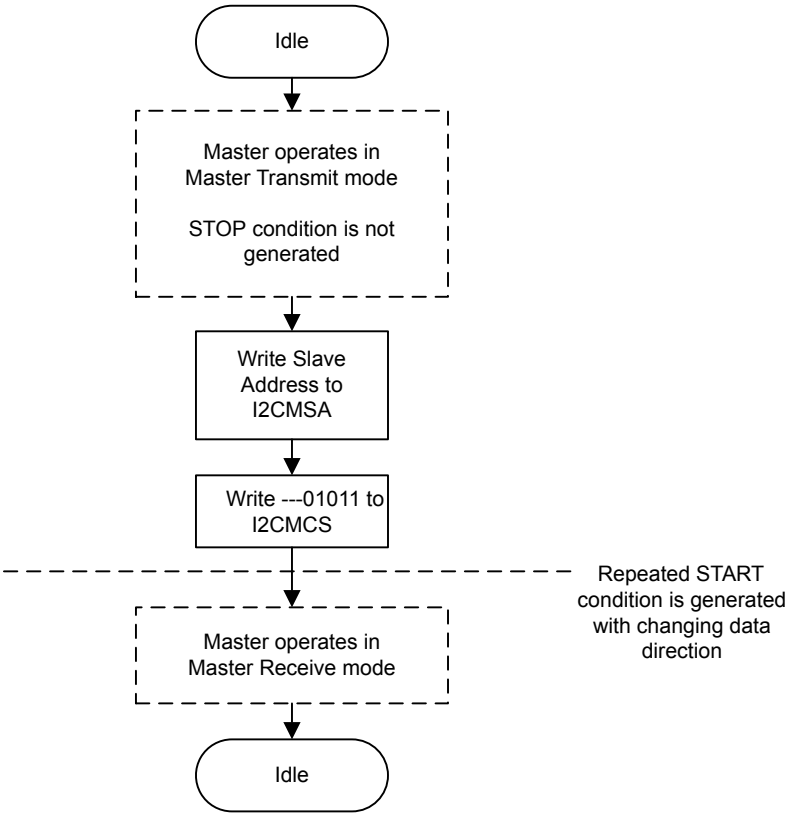
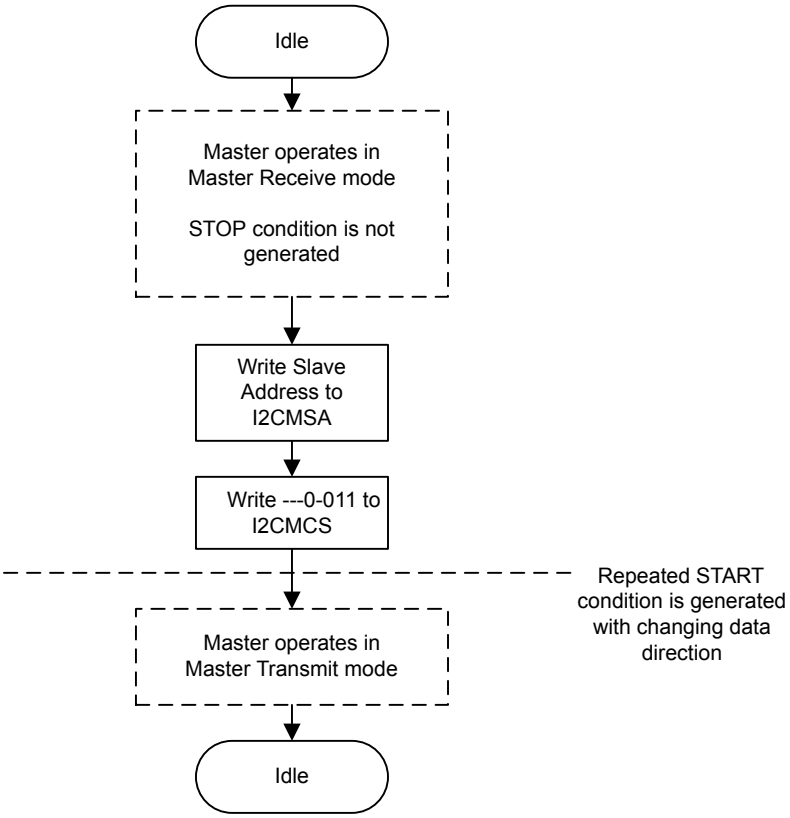


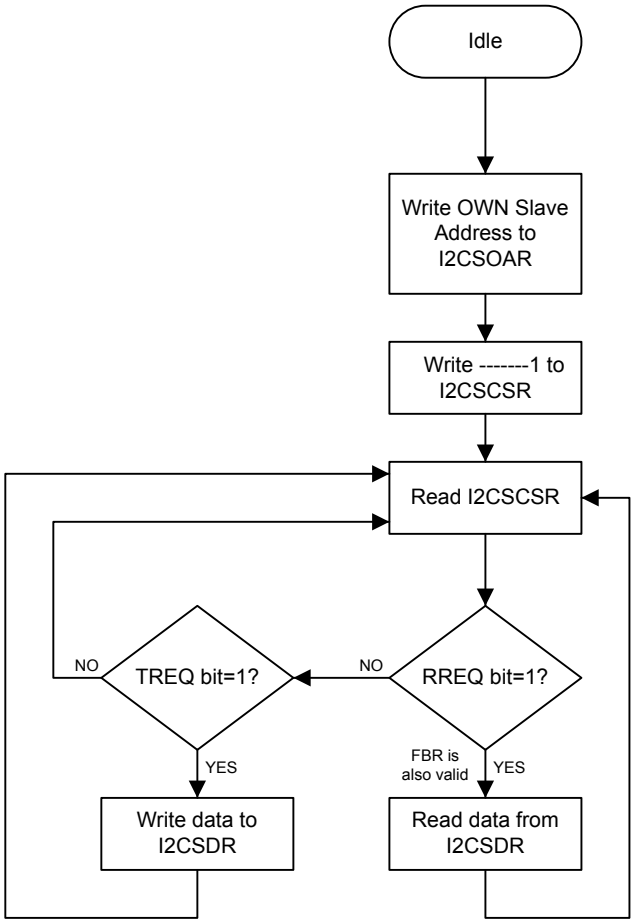
Figure 14-12. Master Burst SEND after Burst RECEIVE



14.3.5.2 I²C Slave Command Sequences

Figure 14-13 on page 524 presents the command sequence available for the I²C slave.

Figure 14-13. Slave Command Sequence



14.4 Initialization and Configuration

The following example shows how to configure the I2C module to send a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I2C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. Also, be sure to enable the same pins for Open Drain operation.
4. Initialize the I2C Master by writing the **I2CMCR** register with a value of 0x0000.0020.
5. 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.

6. Specify the slave address of the master and that the next operation will be a Send by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
7. Place data (byte) to be sent in the data register by writing the **I2CMDR** register with the desired data.
8. Initiate a single byte send of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
9. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

14.5 Register Map

Table 14-4 on page 525 lists the I^2C registers. All addresses given are relative to the I^2C base addresses for the master and slave:

- I^2C 0: 0x4002.0000
- I^2C 1: 0x4002.1000

Note that the I^2C module clock must be enabled before the registers can be programmed (see page 219). There must be a delay of 3 system clocks after the I^2C module clock is enabled before any I^2C module registers are accessed.

The hw_i2c.h file in the StellarisWare® Driver Library uses a base address of 0x800 for the I^2C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that StellarisWare uses an offset between 0x000 and 0x018 with the slave base address.

Table 14-4. Inter-Integrated Circuit (I^2C) Interface Register Map

Offset	Name	Type	Reset	Description	See page
I^2C Master					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	527
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	528
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	532
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	533
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	534
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	535
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	536
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	537
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	538

Table 14-4. Inter-Integrated Circuit (I2C) Interface Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
I2C Slave					
0x800	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	540
0x804	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	541
0x808	I2CSDR	R/W	0x0000.0000	I2C Slave Data	543
0x80C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	544
0x810	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	545
0x814	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	546
0x818	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	547

14.6 Register Descriptions (I2C Master)

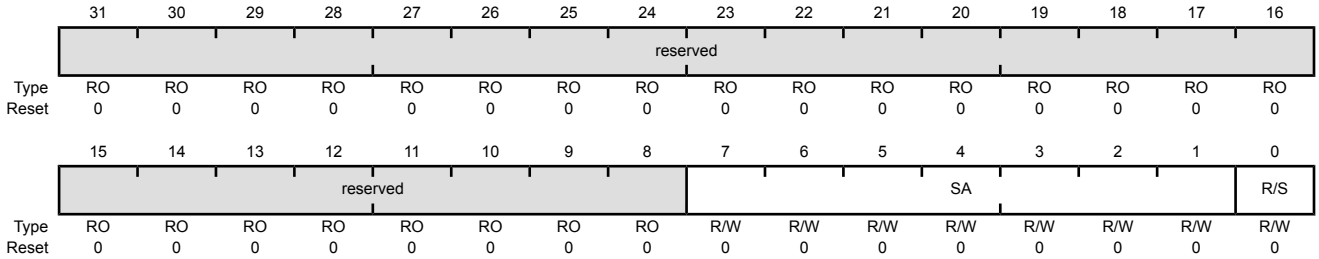
The remainder of this section lists and describes the I2C master registers, in numerical order by address offset. See also “Register Descriptions (I2C Slave)” on page 539.

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 Send (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	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.
7:1	SA	R/W	0	I²C Slave Address This field specifies bits A6 through A0 of the slave address.
0	R/S	R/W	0	Receive/Send The R/S bit specifies if the next operation is a Receive (High) or Send (Low). Value Description 0 Send. 1 Receive.

Register 2: I2C Master Control/Status (I2CMCS), offset 0x004

This register accesses four control bits when written, and accesses seven status bits when read.

The status register consists of seven bits, which when read determine the state of the I2C bus controller.

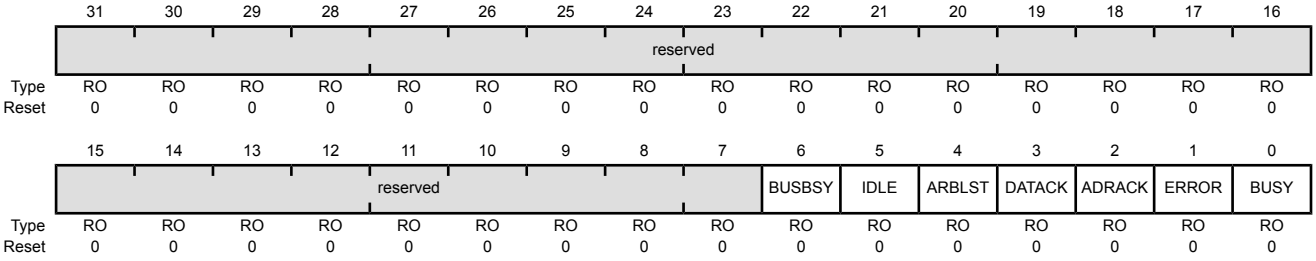
The control register consists of four bits: the RUN, START, STOP, and ACK bits. The START bit causes the generation of the START, or REPEATED START condition.

The STOP bit determines if the cycle stops at the end of the data cycle, or continues on to a burst. To generate a single send cycle, the **I2C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is set to 0, 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 **I2CMDR** register. When the I2C module operates in Master receiver mode, the ACK bit must be set normally to logic 1. This causes the I2C bus controller to send an acknowledge automatically after each byte. This bit must be reset when the I2C bus controller requires no further data to be sent from the slave transmitter.

Reads

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x004
Type RO, reset 0x0000.0000



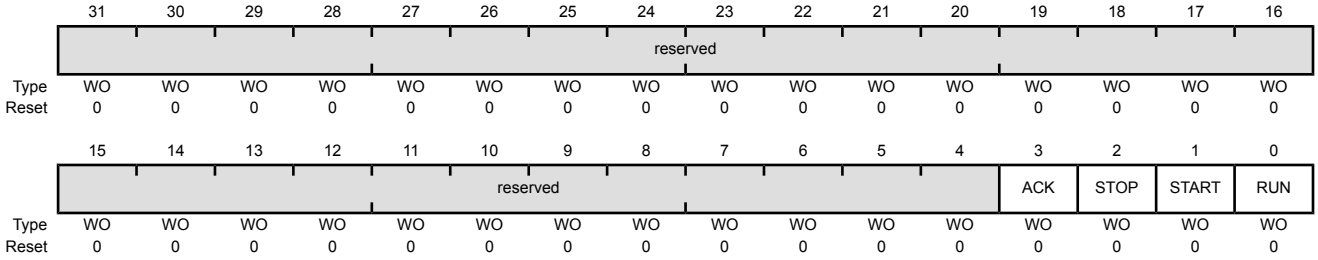
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy This bit specifies the state of the I2C bus. If set, the bus is busy; otherwise, the bus is idle. The bit changes based on the START and STOP conditions.
5	IDLE	RO	0	I2C Idle This bit specifies the I2C controller state. If set, the controller is idle; otherwise the controller is not idle.
4	ARBLST	RO	0	Arbitration Lost This bit specifies the result of bus arbitration. If set, the controller lost arbitration; otherwise, the controller won arbitration.

Bit/Field	Name	Type	Reset	Description
3	DATACK	RO	0	Acknowledge Data This bit specifies the result of the last data operation. If set, the transmitted data was not acknowledged; otherwise, the data was acknowledged.
2	ADRACK	RO	0	Acknowledge Address This bit specifies the result of the last address operation. If set, the transmitted address was not acknowledged; otherwise, the address was acknowledged.
1	ERROR	RO	0	Error This bit specifies the result of the last bus operation. If set, an error occurred on the last operation; otherwise, no error was detected. The error can be from the slave address not being acknowledged or the transmit data not being acknowledged.
0	BUSY	RO	0	I^2C Busy This bit specifies the state of the controller. If set, the controller is busy; otherwise, the controller is idle. When the BUSY bit is set, the other status bits are not valid.

Writes

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	WO	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.
3	ACK	WO	0	Data Acknowledge Enable When set, causes received data byte to be acknowledged automatically by the master. See field decoding in Table 14-5 on page 530.
2	STOP	WO	0	Generate STOP When set, causes the generation of the STOP condition. See field decoding in Table 14-5 on page 530.
1	START	WO	0	Generate START When set, causes the generation of a START or repeated START condition. See field decoding in Table 14-5 on page 530.

Bit/Field	Name	Type	Reset	Description
0	RUN	WO	0	I^2C Master Enable When set, allows the master to send or receive data. See field decoding in Table 14-5 on page 530.

Table 14-5. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)

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 SEND (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a SEND 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.
Master Transmit	X	X	0	0	1	SEND operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	SEND followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a SEND (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND 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 SEND 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.

Table 14-5. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3) *(continued)*

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
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 SEND (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND 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.

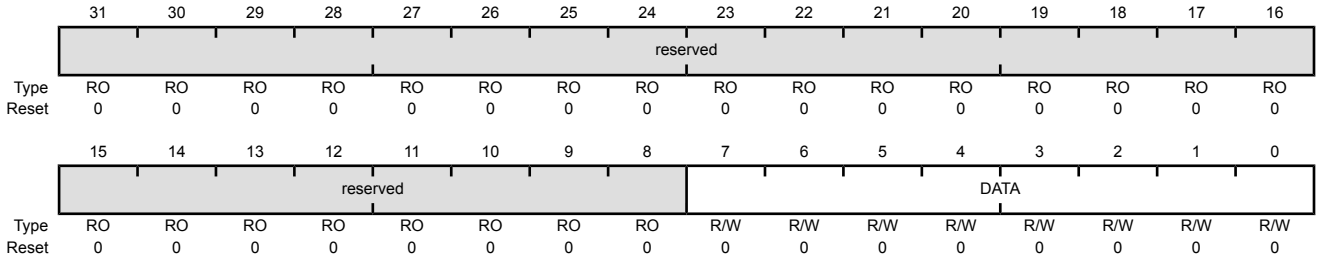
Register 3: I^2C Master Data (I2CMDR), offset 0x008

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	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	DATA	R/W	0x00	Data Transferred Data transferred during transaction.

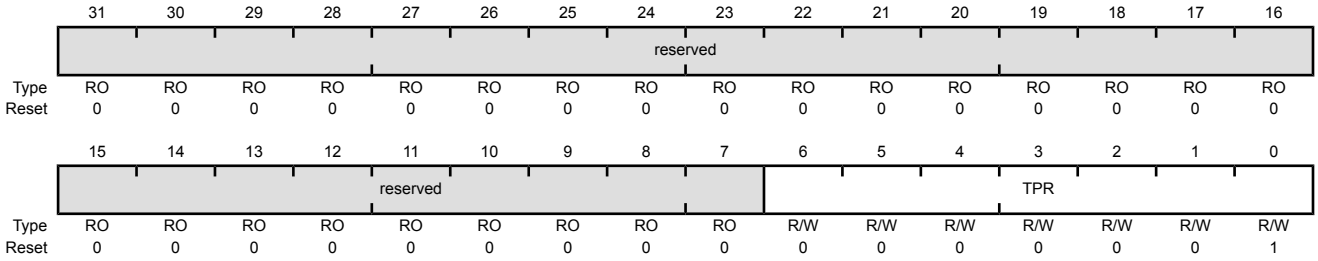
Register 4: I²C Master Timer Period (I2CMTPR), offset 0x00C

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



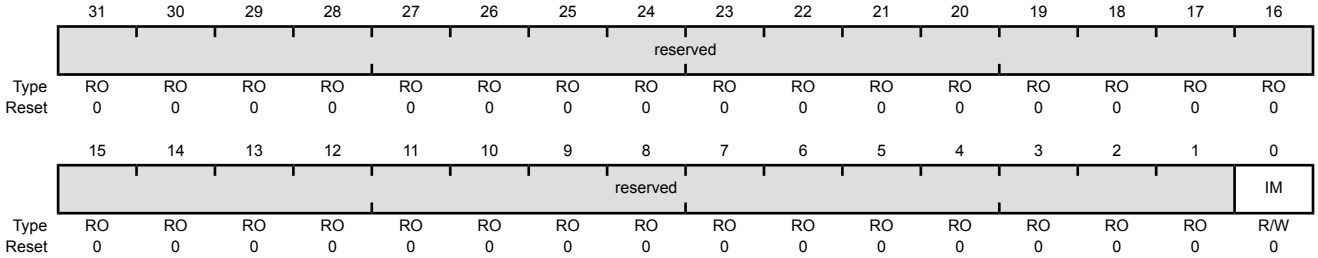
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	TPR	R/W	0x1	SCL Clock Period This field specifies the period of the SCL clock. `SCL_PRD = 2*(1 + TPR)*(SCL_LP + SCL_HP)*CLK_PRD` where: SCL_PRD is the SCL line period (I²C clock). TPR is the Timer Period register value (range of 1 to 127). SCL_LP is the SCL Low period (fixed at 6). SCL_HP is the SCL High period (fixed at 4).

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



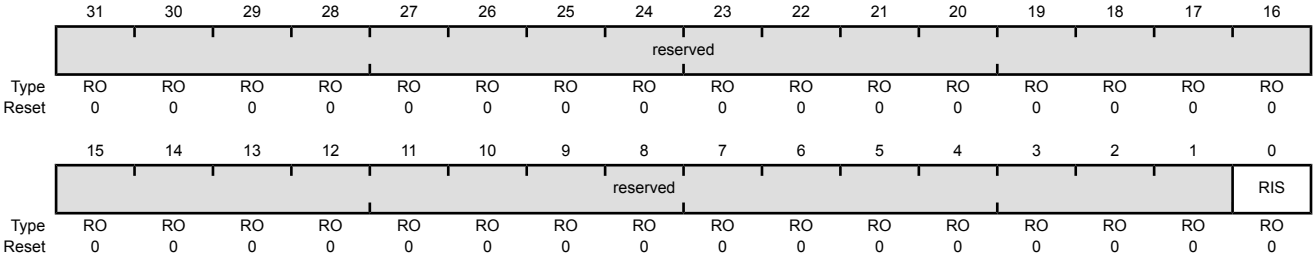
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IM	R/W	0	Interrupt Mask This bit controls whether a raw interrupt is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

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



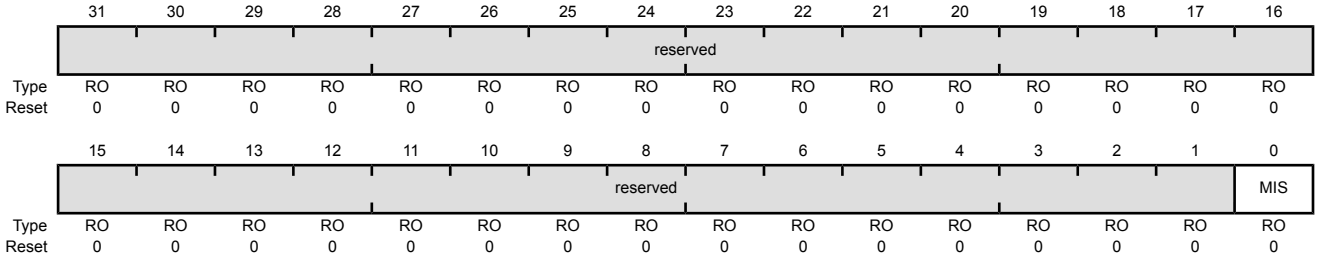
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status This bit specifies the raw interrupt state (prior to masking) of the I²C master block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

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



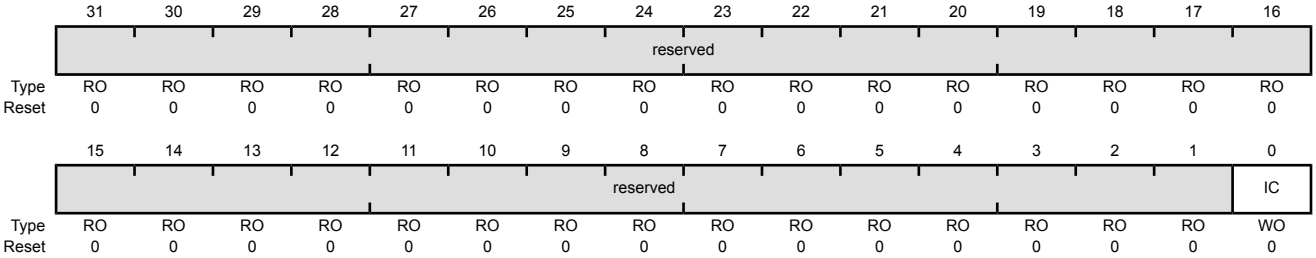
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	Masked Interrupt Status This bit specifies the raw interrupt state (after masking) of the I2C master block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

Register 8: I²C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

I2C Master Interrupt Clear (I2CMICR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x01C
Type WO, reset 0x0000.0000



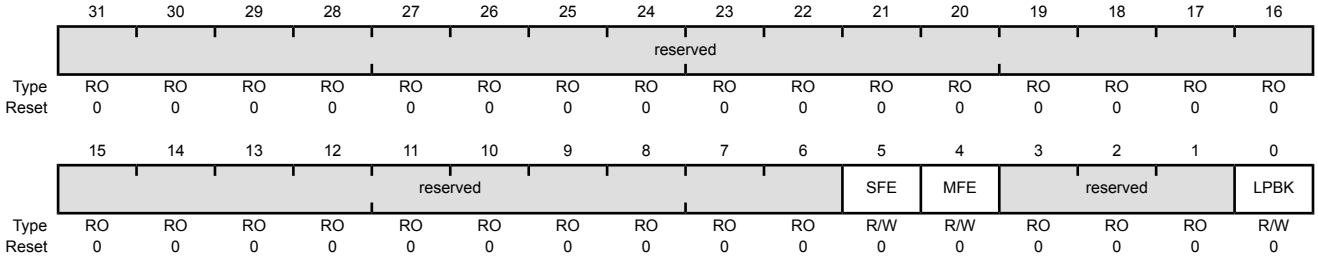
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise, a write of 0 has no affect on the interrupt state. 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	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	SFE	R/W	0	I²C Slave Function Enable This bit specifies whether the interface may operate in Slave mode. If set, Slave mode is enabled; otherwise, Slave mode is disabled.
4	MFE	R/W	0	I²C Master Function Enable This bit specifies whether the interface may operate in Master mode. If set, Master mode is enabled; otherwise, Master mode is disabled and the interface clock is disabled.
3:1	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	LPBK	R/W	0	I²C Loopback This bit specifies whether the interface is operating normally or in Loopback mode. If set, the device is put in a test mode loopback configuration; otherwise, the device operates normally.

14.7 Register Descriptions (I^2C Slave)

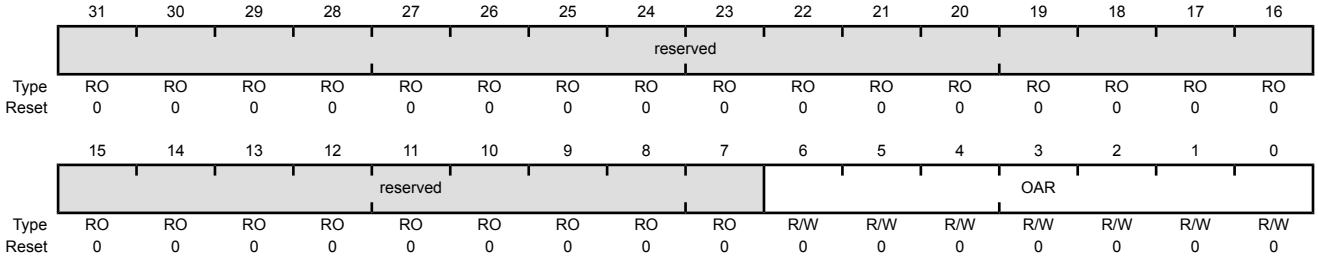
The remainder of this section lists and describes the I^2C slave registers, in numerical order by address offset. See also “Register Descriptions (I^2C Master)” on page 526.

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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	R/W	0x00	I2C Slave Own Address This field specifies bits A6 through A0 of the slave address.

Register 11: I²C Slave Control/Status (I2CSCSR), offset 0x804

This register accesses one control bit when written, and three status bits when read.

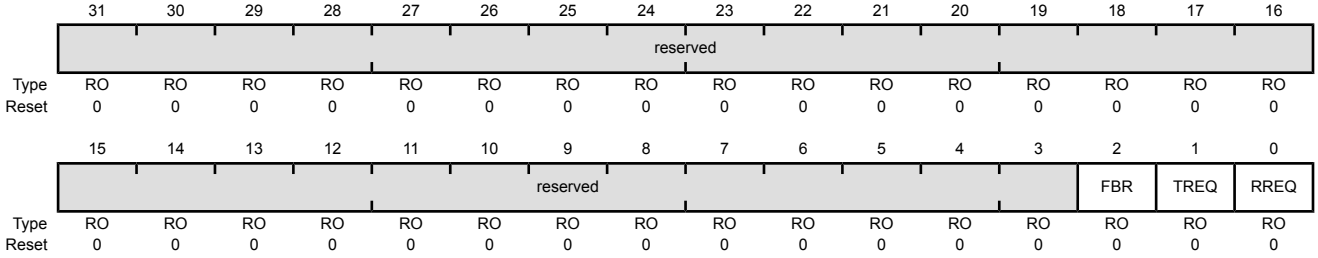
The read-only Status register consists of three bits: the FBR, RREQ, and TREQ bits. The First Byte Received (FBR) bit is set only after the Stellaris device detects its own slave address and receives the first data byte from the I²C master. The Receive Request (RREQ) bit indicates that the Stellaris I²C device has received a data byte from an I²C master. Read one data byte from the **I²C Slave Data (I2CSDR)** register to clear the RREQ bit. The Transmit Request (TREQ) bit indicates that the Stellaris I²C device is addressed as a Slave Transmitter. Write one data byte into the **I²C Slave Data (I2CSDR)** register to clear the TREQ bit.

The write-only Control register consists of one bit: the DA bit. The DA bit enables and disables the Stellaris I²C slave operation.

Reads

I2C Slave Control/Status (I2CSCSR)

I2C 0 base: 0x4002.0000
I2C 1 base: 0x4002.1000
Offset 0x804
Type RO, reset 0x0000.0000

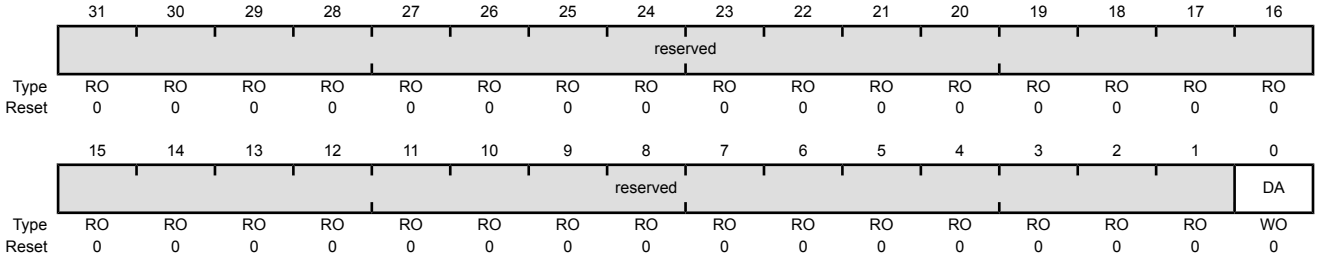


Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FBR	RO	0	First Byte Received Indicates that the first byte following the slave's own address is received. This bit is only valid when the RREQ bit is set, and is automatically cleared when data has been read from the **I2CSDR** register. **Note:** This bit is not used for slave transmit operations.
1	TREQ	RO	0	Transmit Request This bit specifies the state of the I²C slave with regards to outstanding transmit requests. If set, the I²C unit has been addressed as a slave transmitter and uses clock stretching to delay the master until data has been written to the **I2CSDR** register. Otherwise, there is no outstanding transmit request.
0	RREQ	RO	0	Receive Request This bit specifies the status of the I²C slave with regards to outstanding receive requests. If set, the I²C unit has outstanding receive data from the I²C master and uses clock stretching to delay the master until the data has been read from the **I2CSDR** register. Otherwise, no receive data is outstanding.

Writes

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	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	DA	WO	0	Device Active Value Description 0 Disables the I2C slave operation. 1 Enables the I2C slave operation. Once this bit has been set, it should not be set again unless it has been cleared by writing a 0 or by a reset, otherwise transfer failures may occur.

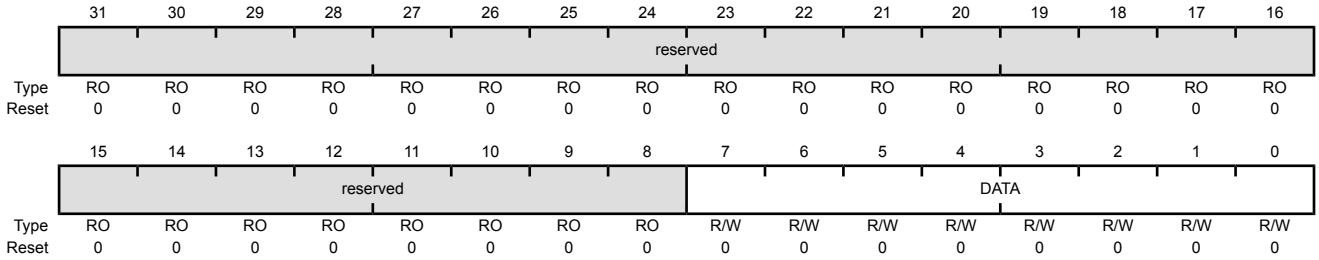
Register 12: I²C Slave Data (I2CSDR), offset 0x808

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



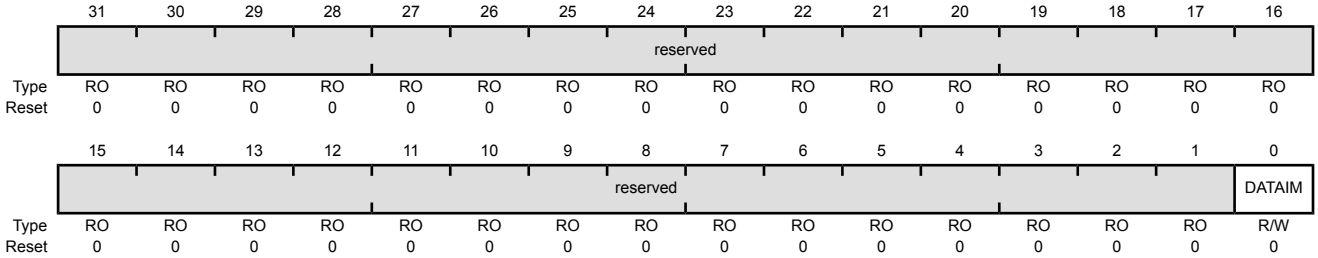
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x0	Data for Transfer This field contains 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



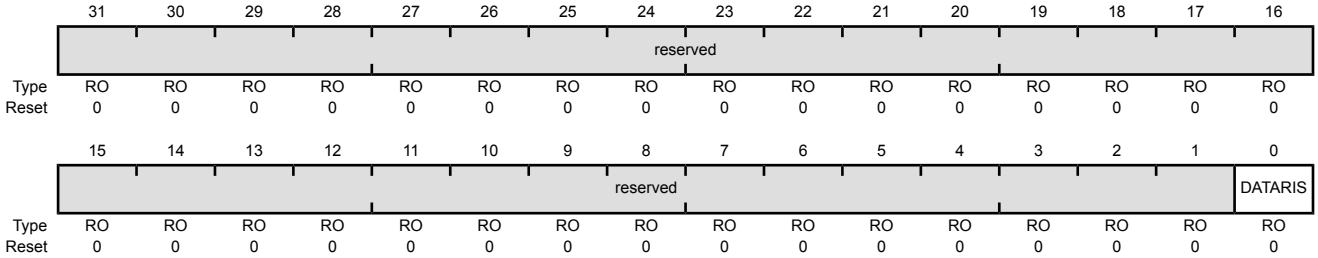
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DATAIM	R/W	0	Data Interrupt Mask This bit controls whether the raw interrupt for data received and data requested is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

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



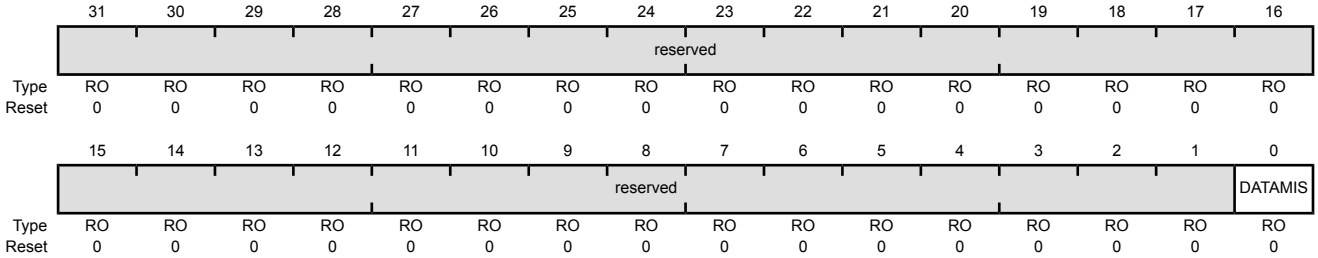
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DATARIS	RO	0	Data Raw Interrupt Status This bit specifies the raw interrupt state for data received and data requested (prior to masking) of the I²C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

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



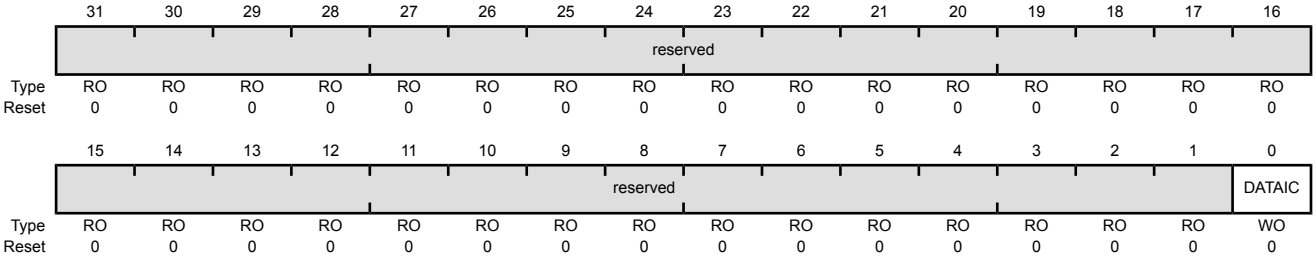
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DATAMIS	RO	0	Data Masked Interrupt Status This bit specifies the interrupt state for data received and data requested (after masking) of the I²C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

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
31:1	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	DATAIC	WO	0	Data Interrupt Clear This bit controls the clearing of the raw interrupt for data received and data requested. When set, it clears the DATARIS interrupt bit; otherwise, it has no effect on the DATARIS bit value.

15 Controller Area Network (CAN) Module

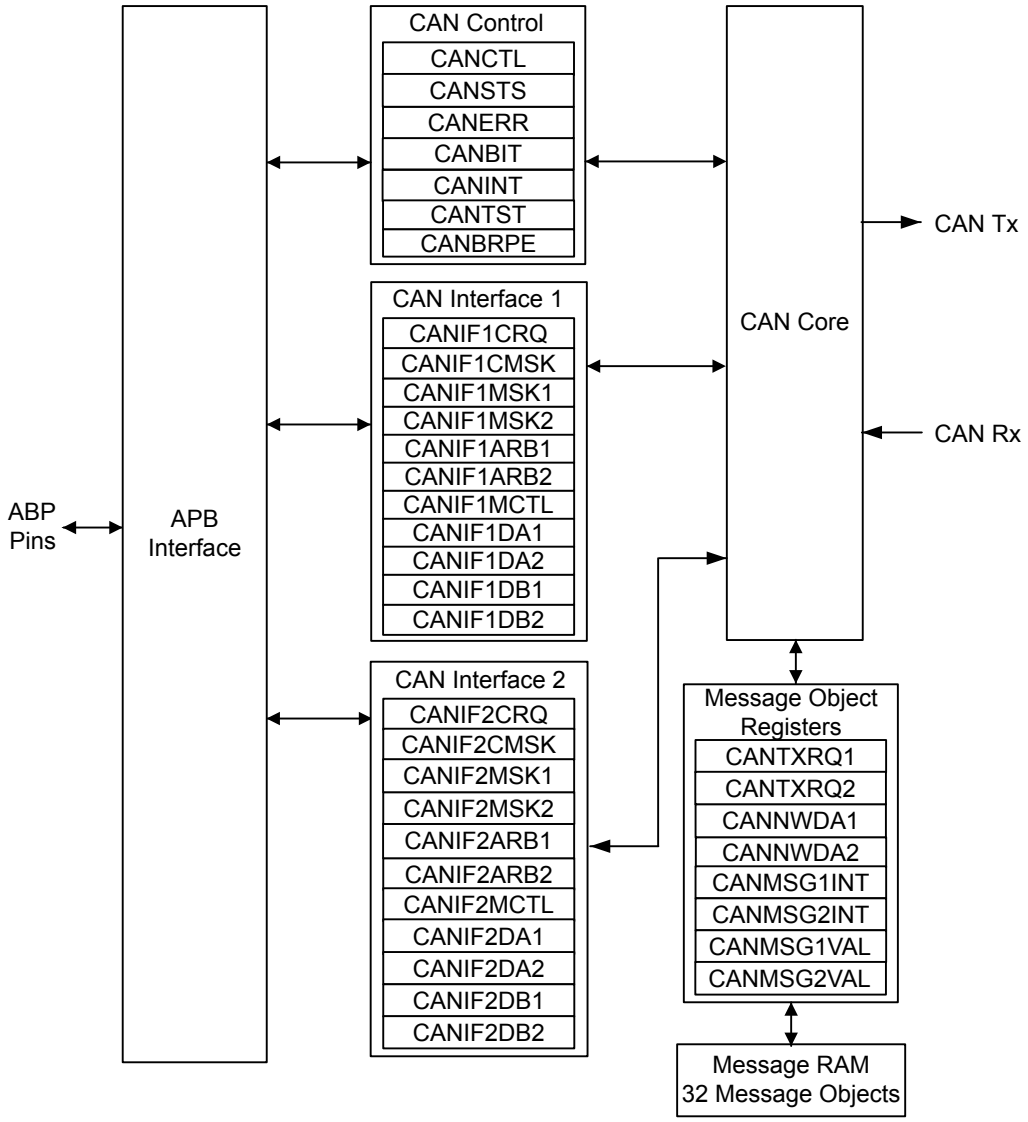
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally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

The Stellaris® CAN controller supports the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable Loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN interface through the CANnTX and CANnRX signals

15.1 Block Diagram

Figure 15-1. CAN Controller Block Diagram



15.2 Signal Description

Table 15-1 on page 550 and Table 15-2 on page 550 list the external signals of the CAN controller and describe the function of each. The CAN controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Assignment" lists the possible GPIO pin placements for the CAN signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the CAN controller function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286.

Table 15-1. Controller Area Network Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
CAN0Rx	10	I	TTL	CAN module 0 receive.
CAN0Tx	11	O	TTL	CAN module 0 transmit.

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

Table 15-2. Controller Area Network Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
CAN0Rx	G1	I	TTL	CAN module 0 receive.
CAN0Tx	G2	O	TTL	CAN module 0 transmit.

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

15.3 Functional Description

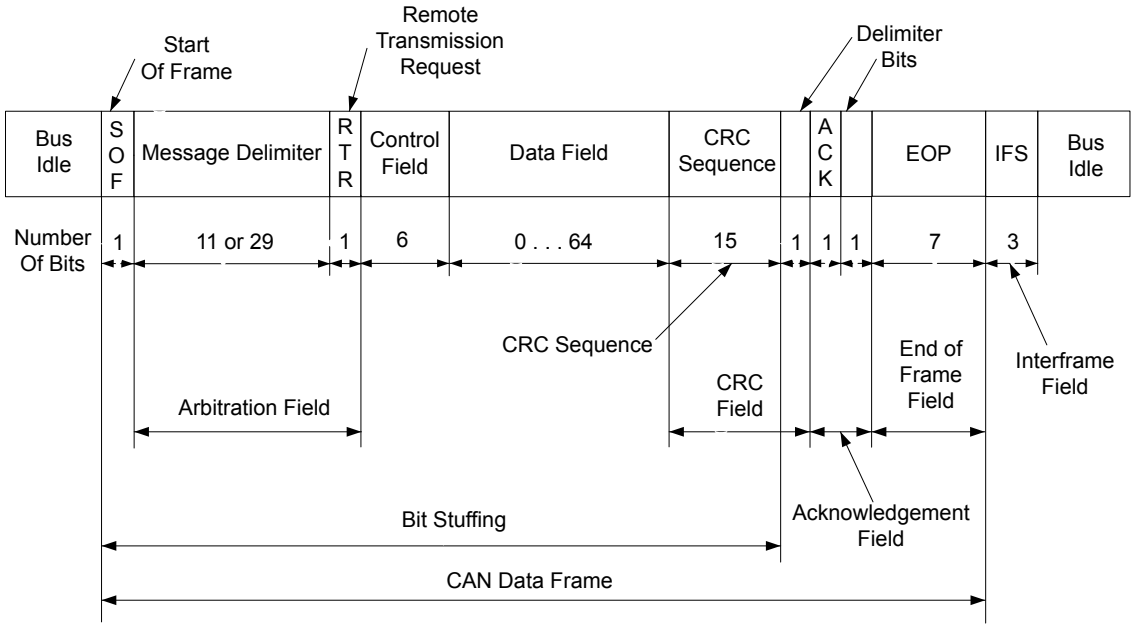
The Stellaris CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

A data frame contains data for transmission, whereas a remote frame contains no data and is used to request the transmission of a specific message object. The CAN data/remote frame is constructed as shown in Figure 15-2 on page 550.

Figure 15-2. CAN Data/Remote Frame



The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These are accessed via either of the CAN message object register interfaces.

The message memory is not directly accessible in the Stellaris memory map, so the Stellaris CAN controller provides an interface to communicate with the message memory via two CAN interface register sets for communicating with the message objects. As there is no direct access to the message object memory, these two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that must be processed. In general, one interface is used for transmit data and one for receive data.

15.3.1 Initialization

Software initialization is started by setting the INIT bit in the **CAN Control (CANCTL)** register (with software or by a hardware reset) or by going bus-off, which occurs when the transmitter's error counter exceeds a count of 255. While INIT is set, all message transfers to and from the CAN bus are stopped and the CANnTX signal is held High. Entering the initialization state does not change the configuration of the CAN controller, the message objects, or the error counters. However, some configuration registers are only accessible while in the initialization state.

To initialize the CAN controller, set the **CAN Bit Timing (CANBIT)** register and configure each message object. If a message object is not needed, label it as not valid by clearing the MSGVAL bit in the **CAN IFn Arbitration 2 (CANIFnARB2)** register. Otherwise, the whole message object must be initialized, as the fields of the message object may not have valid information, causing unexpected results. Both the INIT and CCE bits in the **CANCTL** register must be set in order to access the **CANBIT** register and the **CAN Baud Rate Prescaler Extension (CANBRPE)** register to configure the bit timing. To leave the initialization state, the INIT bit must be cleared. Afterwards, the internal Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (indicating a bus idle condition) before it takes part in bus activities and starts message transfers. Message object initialization does not require the CAN to be in the initialization state and can be done on the fly. However, message objects should all be configured to particular identifiers or set to not valid before message transfer starts. To change the configuration of a message object during normal operation, clear the MSGVAL bit in the **CANIFnARB2** register to indicate that the message object is not valid during the change. When the configuration is completed, set the MSGVAL bit again to indicate that the message object is once again valid.

15.3.2 Operation

There are two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**), which are used to access the message objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The two sets are independent and identical and can be used to queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

Once the CAN module is initialized and the INIT bit in the **CANCTL** register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the message handler's filtering process, and if it passes through the filter, is stored in the message object specified by the MNUM bit in the **CAN IFn Command Request**

(CANIFnCRQ) register. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the MSK bits in the **CAN IFn Mask 1** and **CAN IFn Mask 2 (CANIFnMSKn)** registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time via the CAN Interface Registers. The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects is under the control of the software that is managing the CAN hardware. These can be message objects used for one-time data transfers, or permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. At the start of transmission, the appropriate TXRQST bit in the **CAN Transmission Request n (CANTXRQn)** register and the NEWDAT bit in the **CAN New Data n (CANNWDAn)** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier (MNUM) for the message object, with 1 being the highest priority and 32 being the lowest priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the RMTEN bit in the **CAN IFn Message Control (CANIFnMCTL)** register. A matching received remote frame causes the TXRQST bit to be set and the message object automatically transfers its data or generates an interrupt indicating a remote frame was requested. This can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are identified as remote frame requests. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are identified as a remote frame request. The MXTD bit in the **CANIFnMSK2** register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

15.3.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if there is no data transfer occurring between the CAN Interface Registers and message RAM, the valid message object with the highest priority that has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's NEWDAT bit in the **CANNWDAn** register is cleared. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the TXRQST bit in the **CANTXRQn** register is cleared. If the CAN controller is set up to interrupt upon a successful transmission of a message object, (the TXIE bit in the **CAN IFn Message Control (CANIFnMCTL)** register is set), the INTPND bit in the **CANIFnMCTL** register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

15.3.4 Configuring a Transmit Message Object

The following steps illustrate how to configure a transmit message object.

1. In the **CAN IFn Command Mask (CANIFnCMASK)** register:
   - Set the WRNRD bit to specify a write to the **CANIFnCMASK** register; specify whether to transfer the IDMASK, DIR, and MXTD of the message object into the **CAN IFn** registers using the MASK bit
   - Specify whether to transfer the ID, DIR, XTD, and MSGVAL of the message object into the interface registers using the ARB bit
   - Specify whether to transfer the control bits into the interface registers using the CONTROL bit
   - Specify whether to clear the INTPND bit in the **CANIFnMCTL** register using the CLRINTPND bit
   - Specify whether to clear the NEWDAT bit in the **CANNWDAn** register using the NEWDAT bit
   - Specify which bits to transfer using the DATAA and DATAB bits
2. In the **CANIFnMSK1** register, use the MSK[15:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[15:0] in this register are used for bits [15:0] of the 29-bit message identifier and are not used for an 11-bit identifier. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.
3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.
4. For a 29-bit identifier, configure ID[15:0] in the **CANIFnARB1** register to are used for bits [15:0] of the message identifier and ID[12:0] in the **CANIFnARB2** register to are used for bits [28:16] of the message identifier. Set the XTD bit to indicate an extended identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.
5. For an 11-bit identifier, disregard the **CANIFnARB1** register and configure ID[12:2] in the **CANIFnARB2** register to are used for bits [10:0] of the message identifier. Clear the XTD bit to indicate a standard identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.
6. In the **CANIFnMCTL** register:
   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
   - Optionally set the TXIE bit to enable the INTPND bit to be set after a successful transmission
   - Optionally set the RMTEN bit to enable the TXRQST bit to be set upon the reception of a matching remote frame allowing automatic transmission

- Set the EOB bit for a single message object;
- Set the DLC[3:0] field to specify the size of the data frame. Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

7. Load the data to be transmitted into the CAN IFn Data **(CANIFnDA1, CANIFnDA2, CANIFnDB1, CANIFnDB2)** or **(CANIFnDATAA and CANIFnDATAB)** registers. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register.
8. Program the number of the message object to be transmitted in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register.
9. When everything is properly configured, set the TXRQST bit in the **CANIFnMCTL** register. Once this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Note that setting the RMTEN bit in the **CANIFnMCTL** register can also start message transmission if a matching remote frame has been received.

15.3.5 Updating a Transmit Message Object

The CPU may update the data bytes of a Transmit Message Object any time via the CAN Interface Registers and neither the MSGVAL bit in the **CANIFnARB2** register nor the TXRQST bits in the **CANIFnMCTL** register have to be cleared before the update.

Even if only some of the data bytes are to be updated, all four bytes of the corresponding **CANIFnDAn**/**CANIFnDBn** register have to be valid before the content of that register is transferred to the message object. Either the CPU must write all four bytes into the **CANIFnDAn**/**CANIFnDBn** register or the message object is transferred to the **CANIFnDAn**/**CANIFnDBn** register before the CPU writes the new data bytes.

In order to only update the data in a message object, the WRNRD, DATAA and DATAB bits in the **CANIFnMSKn** register are set, followed by writing the updated data into **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** registers, and then the number of the message object is written to the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. To begin transmission of the new data as soon as possible, set the TXRQST bit in the **CANIFnMSKn** register.

To prevent the clearing of the TXRQST bit in the **CANIFnMCTL** register at the end of a transmission that may already be in progress while the data is updated, the NEWDAT and TXRQST bits have to be set at the same time in the **CANIFnMCTL** register. When these bits are set at the same time, NEWDAT is cleared as soon as the new transmission has started.

15.3.6 Accepting Received Message Objects

When the arbitration and control field (the ID and XTD bits in the **CANIFnARB2** and the RMTEN and DLC[3:0] bits of the **CANIFnMCTL** register) of an incoming message is completely shifted into the CAN controller, the message handling capability of the controller starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the controller uses the acceptance filtering programmed through the mask bits in the **CANIFnMSKn** register and enabled using the UMASK bit in the **CANIFnMCTL** register. Each valid message object, starting with object 1, is compared with the incoming message to locate a matching message object in the message RAM. If a match occurs, the scanning is stopped and the message handler proceeds depending on whether it is a data frame or remote frame that was received.

15.3.7 Receiving a Data Frame

The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the DLC bits

are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used. The NEWDAT bit of the **CANIFnMCTL** register is set to indicate that new data has been received. The CPU should clear this bit when it reads the message object to indicate to the controller that the message has been received, and the buffer is free to receive more messages. If the CAN controller receives a message and the NEWDAT bit is already set, the MSGLST bit in the **CANIFnMCTL** register is set to indicate that the previous data was lost. If the system requires an interrupt upon successful reception of a frame, the RXIE bit of the **CANIFnMCTL** register should be set. In this case, the INTPND bit of the same register is set, causing the **CANINT** register to point to the message object that just received a message. The TXRQST bit of this message object should be cleared to prevent the transmission of a remote frame.

15.3.8 Receiving a Remote Frame

A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object have to be considered:

Configuration in CANIFnMCTL	Description
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 1 (set the TXRQST bit of the **CANIFnMCTL** register at reception of the frame to enable transmission) ■ UMASK = 1 or 0	At the reception of a matching remote frame, the TXRQST bit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible.
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 0 (ignore mask in the **CANIFnMSKn** register)	At the reception of a matching remote frame, the TXRQST bit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and there is no indication that the remote frame ever happened.
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 1 (use mask (MSK, MXTD, and MDIR in the **CANIFnMSKn** register) for acceptance filtering)	At the reception of a matching remote frame, the TXRQST bit of this message object is cleared. The arbitration and control field (ID + XTD + RMTEN + DLC) from the shift register is stored into the message object in the message RAM and the NEWDAT bit of this message object is set. The data field of the message object remains unchanged; the remote frame is treated similar to a received data frame. This is useful for a remote data request from another CAN device for which the Stellaris controller does not have readily available data. The software must fill the data and answer the frame manually.

15.3.9 Receive/Transmit Priority

The receive/transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This should not be confused with the message identifier as that priority is enforced by the CAN bus. This means that if message object 1 and message object 2 both have valid messages that need to be transmitted, message object 1 will always be transmitted first regardless of the message identifier in the message object itself.

15.3.10 Configuring a Receive Message Object

The following steps illustrate how to configure a receive message object.

1. Program the **CAN IFn Command Mask (CANIFnCMASK)** register as described in the "Configuring a Transmit Message Object" on page 552 section, except that the WRNRD bit is set to specify a write to the message RAM.

2. Program the **CANIFnMSK1**and **CANIFnMSK2** registers as described in the "Configuring a Transmit Message Object" on page 552 section to configure which bits are used for acceptance filtering. Note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

4. Program the **CANIFnARB1** and **CANIFnARB2** registers as described in the "Configuring a Transmit Message Object" on page 552 section to program XTD and ID bits for the message identifier to be received; set the MSGVAL bit to indicate a valid message; and clear the DIR bit to specify receive.

5. In the **CANIFnMCTL** register:

   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
   - Optionally set the RXIE bit to enable the INTPND bit to be set after a successful reception
   - Clear the RMTEN bit to leave the TXRQST bit unchanged
   - Set the EOB bit for a single message object
   - Set the DLC[3:0] field to specify the size of the data frame

   Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

6. Program the number of the message object to be received in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. Reception of the message object begins as soon as a matching frame is available on the CAN bus.

When the message handler stores a data frame in the message object, it stores the received Data Length Code and eight data bytes in the **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** register. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by unspecified values.

The CAN mask registers can be used to allow groups of data frames to be received by a message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are received by a message object. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the

CANIFnMSKn register to filter which frames are received. The MXTD bit in the **CANIFnMSK2** register should be set if only 29-bit extended identifiers are expected by this message object.

15.3.11 Handling of Received Message Objects

The CPU may read a received message any time via the CAN Interface registers because the data consistency is guaranteed by the message handler state machine.

Typically, the CPU first writes 0x007F to the **CANIFnCMSK** register and then writes the number of the message object to the **CANIFnCRQ** register. That combination transfers the whole received message from the message RAM into the Message Buffer registers (**CANIFnMSKn**, **CANIFnARBn**, and **CANIFnMCTL**). Additionally, the NEWDAT and INTPND bits are cleared in the message RAM, acknowledging that the message has been read and clearing the pending interrupt generated by this message object.

If the message object uses masks for acceptance filtering, the **CANIFnARBn** registers show the full, unmasked ID for the received message.

The NEWDAT bit in the **CANIFnMCTL** register shows whether a new message has been received since the last time this message object was read. The MSGLST bit in the **CANIFnMCTL** register shows whether more than one message has been received since the last time this message object was read. MSGLST is not automatically cleared, and should be cleared by software after reading its status.

Using a remote frame, the CPU may request new data from another CAN node on the CAN bus. Setting the TXRQST bit of a receive object causes the transmission of a remote frame with the receive object's identifier. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TXRQST bit is automatically reset. This prevents the possible loss of data when the other device on the CAN bus has already transmitted the data slightly earlier than expected.

15.3.11.1 Configuration of a FIFO Buffer

With the exception of the EOB bit in the **CANIFnMCTL** register, the configuration of receive message objects belonging to a FIFO buffer is the same as the configuration of a single receive message object (see “Configuring a Receive Message Object” on page 556). To concatenate two or more message objects into a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest message object number is the first message object in a FIFO buffer. The EOB bit of all message objects of a FIFO buffer except the last one must be cleared. The EOB bit of the last message object of a FIFO buffer is set, indicating it is the last entry in the buffer.

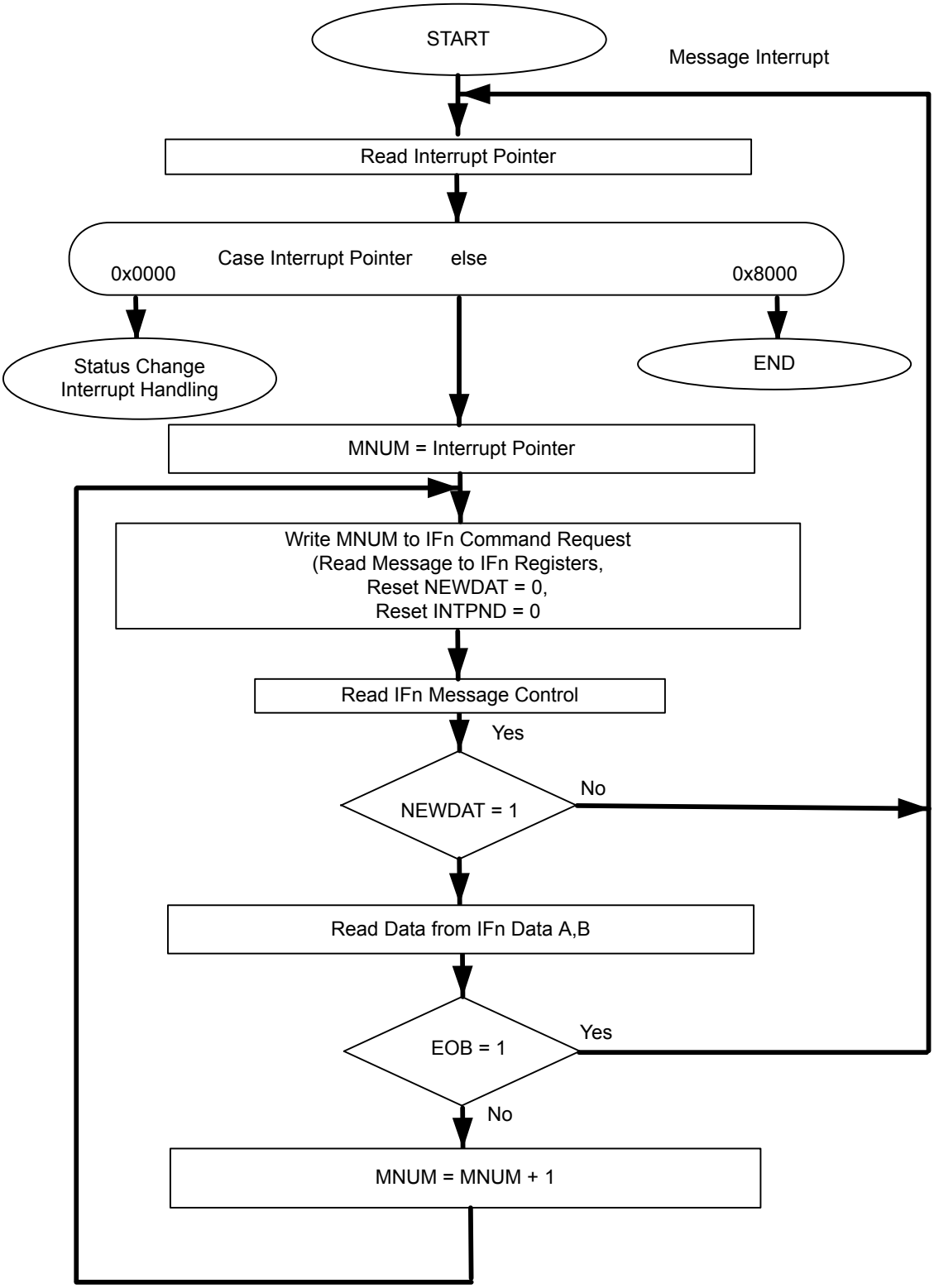
15.3.11.2 Reception of Messages with FIFO Buffers

Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NEWDAT of the **CANIFnMCTL** register bit of this message object is set. By setting NEWDAT while EOB is clear, the message object is locked and cannot be written to by the message handler until the CPU has cleared the NEWDAT bit. Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. If none of the preceding message objects has been released by clearing the NEWDAT bit, all further messages for this FIFO buffer will be written into the last message object of the FIFO buffer and therefore overwrite previous messages.

15.3.11.3 Reading from a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the **CANIFnCRQ**, the TXRQST and CLRINTPND bits in the **CANIFnCMSK** register should be set such that the NEWDAT and INTPEND bits in the **CANIFnMCTL** register are cleared after the read. The values of these bits in the **CANIFnMCTL** register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. When reading from the FIFO buffer, the user should be aware that a new received message is placed in the message object with the lowest message number for which the NEWDAT bit of the **CANIFnMCTL** register. As a result, the order of the received messages in the FIFO is not guaranteed. Figure 15-3 on page 559 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 15-3. Message Objects in a FIFO Buffer



15.3.12 Handling of Interrupts

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. The status interrupt has the highest priority. Among the message interrupts, the message object's interrupt with the lowest message number has the highest priority. A message interrupt is cleared by clearing the message object's INTPND bit in the **CANIFnMCTL** register or by reading the **CAN Status (CANSTS)** register. The status Interrupt is cleared by reading the **CANSTS** register.

The interrupt identifier INTID in the **CANINT** register indicates the cause of the interrupt. When no interrupt is pending, the register reads as 0x0000. If the value of the INTID field is different from 0, then there is an interrupt pending. If the IE bit is set in the **CANCTL** register, the interrupt line to the CPU is active. The interrupt line remains active until the INTID field is 0, meaning that all interrupt sources have been cleared (the cause of the interrupt is reset), or until IE is cleared, which disables interrupts from the CAN controller.

The INTID field of the **CANINT** register points to the pending message interrupt with the highest interrupt priority. The SIE bit in the **CANCTL** register controls whether a change of the RXOK, TXOK, and LEC bits in the **CANSTS** register can cause an interrupt. The EIE bit in the **CANCTL**register controls whether a change of the BOFF and EWARN bits in the **CANSTS** can cause an interrupt. The IE bit in the **CANCTL** controls whether any interrupt from the CAN controller actually generates an interrupt to the microcontroller's interrupt controller. The **CANINT** register is updated even when the IE bit in the **CANCTL** register is clear, but the interrupt will not be indicated to the CPU.

A value of 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed, the **CANSTS** , indicating that either an error or status interrupt has been generated. A write access to the **CANSTS** register can clear the RXOK, TXOK, and LEC bits in that same register; however, the only way to clear the source of a status interrupt is to read the **CANSTS** register.

There are two ways to determine the source of an interrupt during interrupt handling. The first is to read the INTID bit in the **CANINT** register to determine the highest priority interrupt that is pending, and the second is to read the **CAN Message Interrupt Pending (CANMSGnINT)** register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and clear the message object's INTPND bit at the same time by setting the CLRINTPND bit in the **CANIFnCMSK** register. Once the INTPND bit has been cleared, the **CANINT** register contains the message number for the next message object with a pending interrupt.

15.3.13 Test Mode

A Test Mode is provided, which allows various diagnostics to be performed. Test Mode is entered by setting the TEST bit **CANCTL** register. Once in Test Mode, the TX[1:0], LBACK, SILENT and BASIC bits in the **CAN Test (CANTST)** register can be used to put the CAN controller into the various diagnostic modes. The RX bit in the **CANTST** register allows monitoring of the CANnRX signal. All **CANTST** register functions are disabled when the TEST bit is cleared.

15.3.13.1 Silent Mode

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The CAN Controller is put in Silent Mode setting the SILENT bit in the **CANTST** register. In Silent Mode, the CAN controller is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag,

or active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus remains in recessive state.

15.3.13.2 Loopback Mode

Loopback mode is useful for self-test functions. In Loopback Mode, the CAN Controller internally routes the CANnTX signal on to the CANnRX signal and treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into the message buffer. The CAN Controller is put in Loopback Mode by setting the LBACK bit in the **CANTST** register. To be independent from external stimulation, the CAN Controller ignores acknowledge errors (a recessive bit sampled in the acknowledge slot of a data/remote frame) in Loopback Mode. The actual value of the CANnRX signal is disregarded by the CAN Controller. The transmitted messages can be monitored on the CANnTX signal.

15.3.13.3 Loopback Combined with Silent Mode

Loopback Mode and Silent Mode can be combined to allow the CAN Controller to be tested without affecting a running CAN system connected to the CANnTX and CANnRX signals. In this mode, the CANnRX signal is disconnected from the CAN Controller and the CANnTX signal is held recessive. This mode is enabled by setting both the LBACK and SILENT bits in the **CANTST** register.

15.3.13.4 Basic Mode

Basic Mode allows the CAN Controller to be operated without the Message RAM. In Basic Mode, The CANIF1 registers are used as the transmit buffer. The transmission of the contents of the IF1 registers is requested by setting the BUSY bit of the **CANIF1CRQ** register. The CANIF1 registers are locked while the BUSY bit is set. The BUSY bit indicates that a transmission is pending. As soon the CAN bus is idle, the CANIF1 registers are loaded into the shift register of the CAN Controller and transmission is started. When the transmission has completed, the BUSY bit is cleared and the locked CANIF1 registers are released. A pending transmission can be aborted at any time by clearing the BUSY bit in the **CANIF1CRQ** register while the CANIF1 registers are locked. If the CPU has cleared the BUSY bit, a possible retransmission in case of lost arbitration or an error is disabled.

The CANIF2 Registers are used as a receive buffer. After the reception of a message, the contents of the shift register is stored into the CANIF2 registers, without any acceptance filtering. Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read message object is initiated by setting the BUSY bit of the **CANIF2CRQ** register, the contents of the shift register are stored into the CANIF2 registers.

In Basic Mode, all message-object-related control and status bits and of the control bits of the **CANIFnCMSK** registers are not evaluated. The message number of the **CANIFnCRQ** registers is also not evaluated. In the **CANIF2MCTL** register, the NEWDAT and MSGLST bits retain their function, the DLC[3:0] field shows the received DLC, the other control bits are cleared.

Basic Mode is enabled by setting the BASIC bit in the **CANTST** register.

15.3.13.5 Transmit Control

Software can directly override control of the CANnTX signal in four different ways.

- CANnTX is controlled by the CAN Controller
- The sample point is driven on the CANnTX signal to monitor the bit timing
- CANnTX drives a low value

- CANnTX drives a high value

The last two functions, combined with the readable CAN receive pin CANnRX, can be used to check the physical layer of the CAN bus.

The Transmit Control function is enabled by programming the TX[1:0] field in the **CANTST** register. The three test functions for the CANnTX signal interfere with all CAN protocol functions. TX[1:0] must be cleared when CAN message transfer or Loopback Mode, Silent Mode, or Basic Mode are selected.

15.3.14 Bit Timing Configuration Error Considerations

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronization amends a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration, however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive. The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

15.3.15 Bit Time and Bit Rate

The CAN system supports bit rates in the range of lower than 1 Kbps up to 1000 Kbps. Each member of the CAN network has its own clock generator. The timing parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods may be different.

Because of small variations in frequency caused by changes in temperature or voltage and by deteriorating components, these oscillators are not absolutely stable. As long as the variations remain inside a specific oscillator's tolerance range, the CAN nodes are able to compensate for the different bit rates by periodically resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 15-4 on page 563): the Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 15-3 on page 563). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's input clock (fsys) and the Baud Rate Prescaler (BRP):

t_q = BRP / fsys

The fsys input clock is 8 MHz.

The Synchronization Segment Sync is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync and the Sync is called the phase error of that edge.

The Propagation Time Segment Prop is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase1 and Phase2 surround the Sample Point.

The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

Figure 15-4. CAN Bit Time

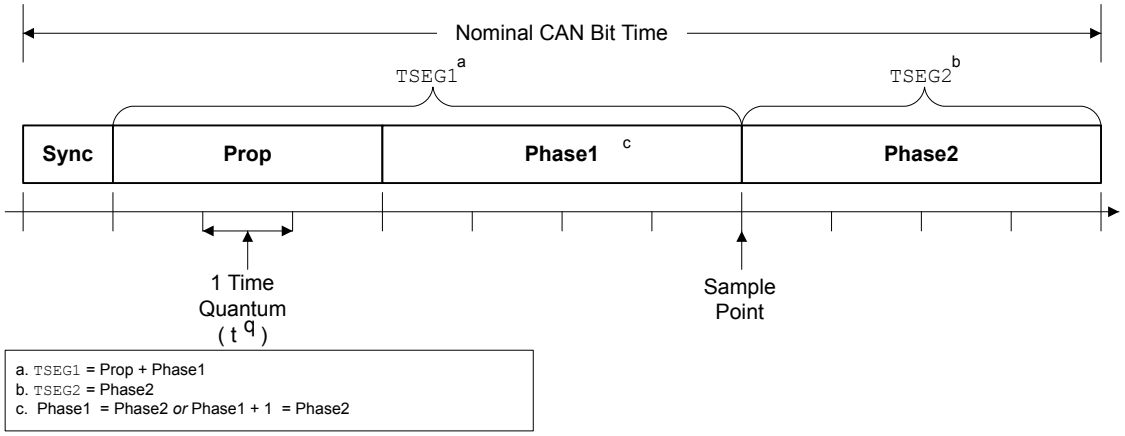


Table 15-3. CAN Protocol Ranges[a]

Parameter	Range	Remark
BRP	[1 .. 64]	Defines the length of the time quantum t_q. The **CANBRPE** register can be used to extend the range to 1024.
Sync	1 t_q	Fixed length, synchronization of bus input to system clock
Prop	[1 .. 8] t_q	Compensates for the physical delay times
Phase1	[1 .. 8] t_q	May be lengthened temporarily by synchronization
Phase2	[1 .. 8] t_q	May be shortened temporarily by synchronization
SJW	[1 .. 4] t_q	May not be longer than either Phase Buffer Segment

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. In the **CANBIT** register, the four components TSEG2, TSEG1, SJW, and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, for example, SJW (functional range of [1..4]) is represented by only two bits in the SJW bit field. Table 15-4 shows the relationship between the **CANBIT** register values and the parameters.

Table 15-4. CANBIT Register Values

CANBIT Register Field	Setting
TSEG2	Phase2 - 1
TSEG1	Prop + Phase1 - 1
SJW	SJW - 1
BRP	BRP

Therefore, the length of the bit time is (programmed values):

[TSEG1 + TSEG2 + 3] × t_q

or (functional values):

[Sync + Prop + Phase1 + Phase2] × t_q

The data in the **CANBIT** register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) defines the length of the time quantum, the basic time unit of the bit time; the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. In addition, the controller generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. The bit value is received or transmitted at the sample point. The information processing time (IPT) is the time after the sample point needed to calculate the next bit to be transmitted on the CAN bus. The IPT includes any of the following: retrieving the next data bit, handling a CRC bit, determining if bit stuffing is required, generating an error flag or simply going idle.

The IPT is application-specific but may not be longer than 2 t_q; the CAN's IPT is 0 t_q. Its length is the lower limit of the programmed length of Phase2. In case of synchronization, Phase2 may be shortened to a value less than IPT, which does not affect bus timing.

15.3.16 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a required bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the required bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is Prop. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop is converted into time quanta (rounded up to the nearest integer multiple of t_q).

Sync is 1 t_q long (fixed), which leaves (bit time - Prop - 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, that is, Phase2 = Phase1, else Phase2 = Phase1 + 1.

The minimum nominal length of Phase2 has to be regarded as well. Phase2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of [0..2] t_q.

The length of the synchronization jump width is set to the least of 4, Phase1 or Phase2.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times fnom \leq fosc \leq (1 + df) \times fnom$$

where:

- df = Maximum tolerance of oscillator frequency
- fosc = Actual oscillator frequency

- fnom = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(Phase_seg1, Phase_seg2)\,\min}{2 \times (13 \times tbit - Phase_Seg2)}$$

$$df\ \max = 2 \times df \times fnom$$

where:

- Phase1 and Phase2 are from Table 15-3 on page 563
- tbit = Bit Time
- dfmax = Maximum difference between two oscillators

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol-compliant configuration of the CAN bit timing.

15.3.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 8 MHz, and the bit rate is 1 Mbps.

```
bit time = 1 µs = n * tq = 8 * tq
tq = 125 ns
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 125E-9 * 8E6 = 1

tSync = 1 * tq = 125 ns              \\fixed at 1 time quanta

delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 375 ns = 3 * tq                \\375 is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 8 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (8 * tq) - (1 * tq) - (3 * tq)
tPhase 1 + tPhase2 = 4 * tq
```

```
tPhase1 = 2 * tq
tPhase2 = 2 * tq                          \\tPhase2 = tPhase1

tTSeg1 = tProp + tPhase1
tTSeg1 = (3 * tq) + (2 * tq)
tTSeg1 = 5 * tq
tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 2) × tq
tTSeg2 = 2 * tq                           \\Assumes IPT=0

tSJW = 2 * tq                             \\Least of 4, Phase1 and Phase2 = 1
```

In the above example, the bit field values for the **CANBIT** register are:

TSEG2	= TSeg2 -1 = 2-1 = 1
TSEG1	= TSeg1 -1 = 5-1 = 4
SJW	= SJW -1 = 2-1 = 1
BRP	= Baud rate prescaler - 1 = 1-1 =0

The final value programmed into the **CANBIT** register = 0x1440.

15.3.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 8 MHz, and the bit rate is 100 Kbps.

```
bit time = 10 µs = n * tq = 10 * tq
tq = 1 µs
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 1E-6 * 8E6 = 8

tSync = 1 * tq = 1 µs                     \\fixed at 1 time quanta

delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 1 µs = 1 * tq                       \\1 µs is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 10 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (10 * tq) - (1 * tq)  - (1 * tq)
tPhase 1 + tPhase2 = 8 * tq
tPhase1 = 4 * tq
```

```
tPhase2 = 4 * tq                        \\tPhase2 = tPhase1

tTSeg1 = tProp + tPhase1
tTSeg1 = (1 * tq) + (4 * tq)
tTSeg1 = 5 * tq
tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 4) * tq
tTSeg2 = 4 * tq                         \\Assumes IPT=0

tSJW = 4 * tq                           \\Least of 4, Phase1, and Phase2
```

TSEG2	= TSeg2 -1 = 4-1 = 3
TSEG1	= TSeg1 -1 = 5-1 = 4
SJW	= SJW -1 = 4-1 = 3
BRP	= Baud rate prescaler - 1 = 8-1 = 7

The final value programmed into the **CANBIT** register = 0x34C7.

15.4 Register Map

Table 15-5 on page 567 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000

Note that the CAN module clock must be enabled before the registers can be programmed (see page 213). There must be a delay of 3 system clocks after the CAN module clock is enabled before any CAN module registers are accessed.

Table 15-5. CAN Register Map

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	R/W	0x0000.0001	CAN Control	570
0x004	CANSTS	R/W	0x0000.0000	CAN Status	572
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	574
0x00C	CANBIT	R/W	0x0000.2301	CAN Bit Timing	575
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	576
0x014	CANTST	R/W	0x0000.0000	CAN Test	577
0x018	CANBRPE	R/W	0x0000.0000	CAN Baud Rate Prescaler Extension	579

Table 15-5. CAN Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1 Command Request	580
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 Command Mask	581
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 Mask 1	583
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1 Mask 2	584
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 Arbitration 1	585
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 Arbitration 2	586
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 Message Control	588
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 Data A1	590
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 Data A2	590
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 Data B1	590
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 Data B2	590
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 Command Request	580
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 Command Mask	581
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 Mask 1	583
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2 Mask 2	584
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 Arbitration 1	585
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 Arbitration 2	586
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 Message Control	588
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 Data A1	590
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 Data A2	590
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 Data B1	590
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 Data B2	590
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	591
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	591
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	592
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	592
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	593
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	593
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	594
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	594

15.5 CAN Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers that are used to access the Message Objects in the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

Register 1: CAN Control (CANCTL), offset 0x000

This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or clearing INIT. If the device goes bus-off, it sets INIT, stopping all bus activities. Once INIT has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 * 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after INIT is cleared, each time a sequence of 11 High bits has been monitored, a BITERROR0 code is written to the **CANSTS** register (the LEC field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

CAN Control (CANCTL)

CAN0 base: 0x4004.0000
Offset 0x000
Type R/W, reset 0x0000.0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
Type RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								TEST	CCE	DAR	reserved	EIE	SIE	IE	INIT
Type RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	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	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	TEST	R/W	0	Test Mode Enable 0: Normal operation 1: Test mode
6	CCE	R/W	0	Configuration Change Enable 0: Do not allow write access to the **CANBIT** register. 1: Allow write access to the **CANBIT** register if the INIT bit is 1.
5	DAR	R/W	0	Disable Automatic-Retransmission 0: Auto-retransmission of disturbed messages is enabled. 1: Auto-retransmission is disabled.
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EIE	R/W	0	Error Interrupt Enable 0: Disabled. No error status interrupt is generated. 1: Enabled. A change in the BOFF or EWARN bits in the **CANSTS** register generates an interrupt.

Bit/Field	Name	Type	Reset	Description
2	SIE	R/W	0	Status Interrupt Enable 0: Disabled. No status interrupt is generated. 1: Enabled. An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the **CANSTS** register generates an interrupt.
1	IE	R/W	0	CAN Interrupt Enable 0: Interrupts disabled. 1: Interrupts enabled.
0	INIT	R/W	1	Initialization 0: Normal operation. 1: Initialization started.

Register 2: CAN Status (CANSTS), offset 0x004

Important: This register is read-sensitive. See the register description for details.

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

The LEC field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error. The unused error code 7 may be written by the CPU to manually set this field to an invalid error so that it can be checked for a change later.

An error interrupt is generated by the BOFF and EWARN bits and a status interrupt is generated by the RXOK, TXOK, and LEC bits, if the corresponding enable bits in the **CAN Control (CANCTL)** register are set. A change of the EPASS bit or a write to the RXOK, TXOK, or LEC bits does not generate an interrupt.

Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Status (CANSTS)

CAN0 base: 0x4004.0000
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								BOFF	EWARN	EPASS	RXOK	TXOK	LEC		
Type RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	BOFF	RO	0	Bus-Off Status 0: CAN controller is not in bus-off state. 1: CAN controller is in bus-off state.
6	EWARN	RO	0	Warning Status 0: Both error counters are below the error warning limit of 96. 1: At least one of the error counters has reached the error warning limit of 96.
5	EPASS	RO	0	Error Passive 0: The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127. 1: The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.

Bit/Field	Name	Type	Reset	Description
4	RXOK	R/W	0	Received a Message Successfully 0: Since this bit was last cleared, no message has been successfully received. 1: Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering. This bit is never cleared by the CAN module.
3	TXOK	R/W	0	Transmitted a Message Successfully 0: Since this bit was last cleared, no message has been successfully transmitted. 1: Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node. This bit is never cleared by the CAN module.
2:0	LEC	R/W	0x0	Last Error Code This is the type of the last error to occur on the CAN bus.

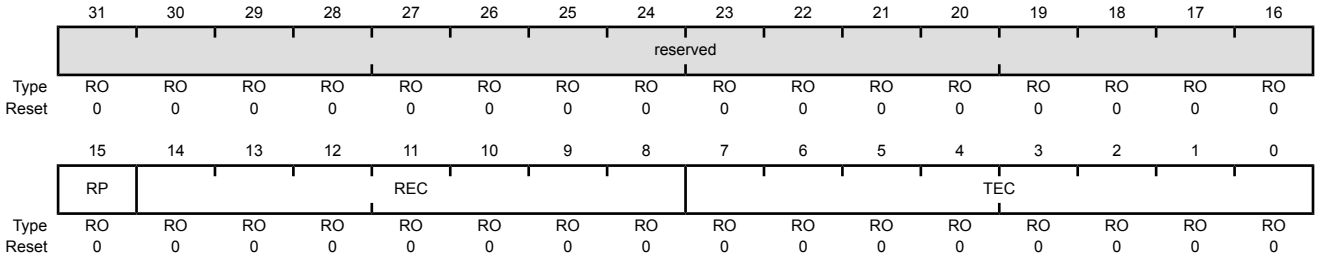
Value	Definition
0x0	No Error
0x1	Stuff Error More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
0x2	Format Error A fixed format part of the received frame has the wrong format.
0x3	ACK Error The message transmitted was not acknowledged by another node.
0x4	Bit 1 Error When a message is transmitted, the CAN controller monitors the data lines to detect any conflicts. When the arbitration field is transmitted, data conflicts are a part of the arbitration protocol. When other frame fields are transmitted, data conflicts are considered errors. A Bit 1 Error indicates that the device wanted to send a High level (logical 1) but the monitored bus value was Low (logical 0).
0x5	Bit 0 Error A Bit 0 Error indicates that the device wanted to send a Low level (logical 0), but the monitored bus value was High (logical 1). During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus.
0x6	CRC Error The CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data.
0x7	No Event When the LEC bit shows this value, no CAN bus event was detected since the CPU wrote this value to LEC.

Register 3: CAN Error Counter (CANERR), offset 0x008

This register contains the error counter values, which can be used to analyze the cause of an error.

CAN Error Counter (CANERR)

CAN0 base: 0x4004.0000
Offset 0x008
Type RO, reset 0x0000.0000



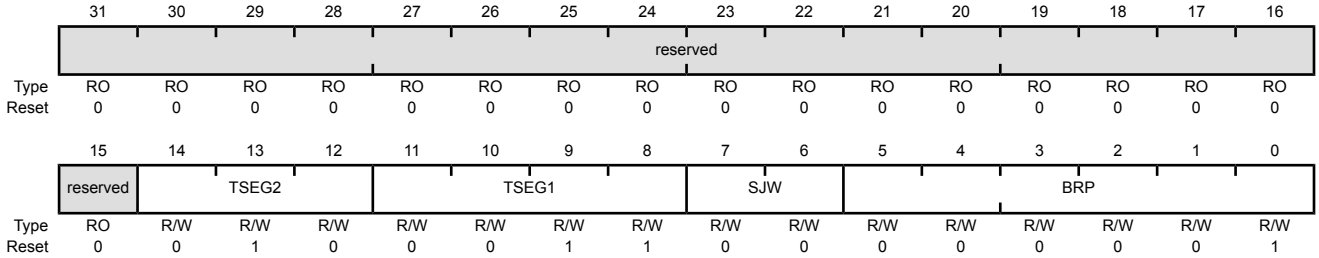
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	RP	RO	0	Received Error Passive 0: The Receive Error counter is below the Error Passive level (127 or less). 1: The Receive Error counter has reached the Error Passive level (128 or greater).
14:8	REC	RO	0x00	Receive Error Counter State of the receiver error counter (0 to 127).
7:0	TEC	RO	0x00	Transmit Error Counter State of the transmit error counter (0 to 255).

Register 4: CAN Bit Timing (CANBIT), offset 0x00C

This register is used to program the bit width and bit quantum. Values are programmed to the system clock frequency. This register is write-enabled by setting the CCE and INIT bits in the **CANCTL** register. See “Bit Time and Bit Rate” on page 562 for more information.

CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000
Offset 0x00C
Type R/W, reset 0x0000.2301



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	TSEG2	R/W	0x2	Time Segment after Sample Point 0x00-0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, a reset value of 0x2 defines that there is 3 (2+1) bit time quanta defined for Phase_Seg2 (see Figure 15-4 on page 563). The bit time quanta is defined by the BRP field.
11:8	TSEG1	R/W	0x3	Time Segment Before Sample Point 0x00-0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, the reset value of 0x3 defines that there is 4 (3+1) bit time quanta defined for Phase_Seg1 (see Figure 15-4 on page 563). The bit time quanta is define by the BRP field.
7:6	SJW	R/W	0x0	(Re)Synchronization Jump Width 0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of TSEG2 or TSEG1 by the value in SJW. So the reset value of 0 adjusts the length by 1 bit time quanta.
5:0	BRP	R/W	0x1	Baud Rate Prescaler The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. 0x00-0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. BRP defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1). The **CANBRPE** register can be used to further divide the bit time.

Register 5: CAN Interrupt (CANINT), offset 0x010

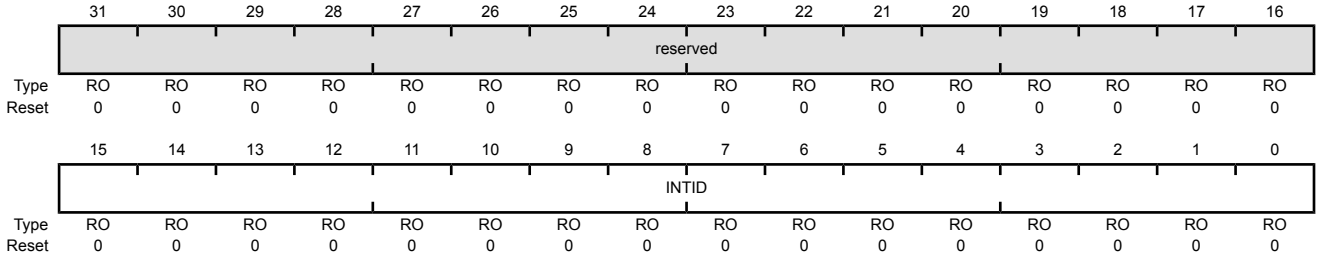
This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the `INTID` field is not 0x0000 (the default) and the `IE` bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the `INTID` field is cleared by reading the **CANSTS** register, or until the `IE` bit in the **CANCTL** register is cleared.

Note: Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000
Offset 0x010
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	INTID	RO	0x0000	Interrupt Identifier The number in this field indicates the source of the interrupt.

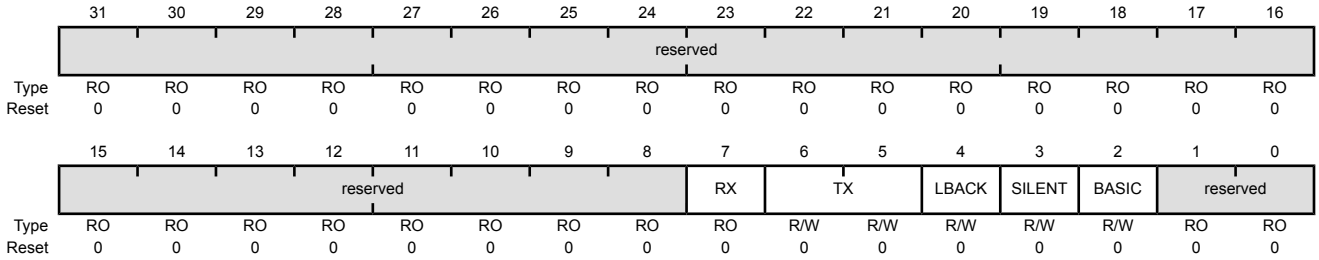
Value	Definition
0x0000	No interrupt pending
0x0001-0x0020	Number of the message object that caused the interrupt
0x0021-0x7FFF	Reserved
0x8000	Status Interrupt
0x8001-0xFFFF	Reserved

Register 6: CAN Test (CANTST), offset 0x014

This is the test mode register for self-test and external pin access. It is write-enabled by setting the TEST bit in the **CANCTL** register. Different test functions may be combined, however, CAN transfers will be affected if the TX bits in this register are not zero.

CAN Test (CANTST)

CAN0 base: 0x4004.0000
Offset 0x014
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	RX	RO	0	Receive Observation Displays the value on the CANnRx pin.
6:5	TX	R/W	0x0	Transmit Control Overrides control of the CANnTx pin.

Value	Description
0x0	CAN Module Control CANnTx is controlled by the CAN module; default operation
0x1	Sample Point The sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing.
0x2	Driven Low CANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus.
0x3	Driven High CANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus.

4	LBACK	R/W	0	Loopback Mode 0: Disabled. 1: Enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.

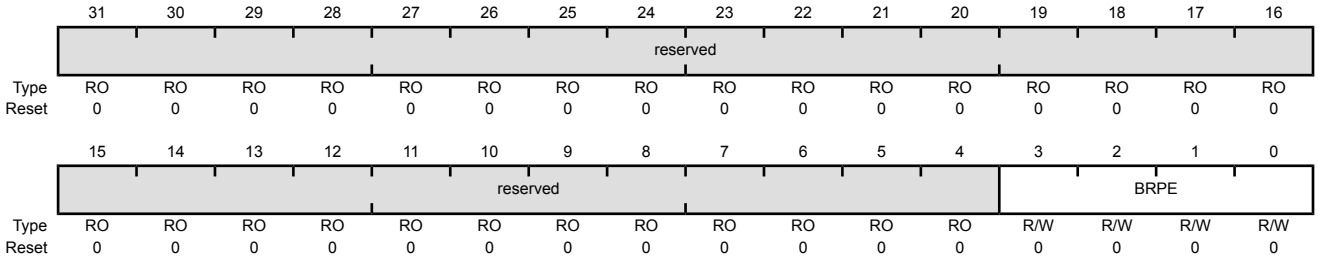
Bit/Field	Name	Type	Reset	Description
3	SILENT	R/W	0	Silent Mode Do not transmit data; monitor the bus. Also known as Bus Monitor mode. 0: Disabled. 1: Enabled.
2	BASIC	R/W	0	Basic Mode 0: Disabled. 1: Use **CANIF1** registers as transmit buffer, and use **CANIF2** registers as receive buffer.
1:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018

This register is used to further divide the bit time set with the BRP bit in the **CANBIT** register. It is write-enabled by setting the CCE bit in the **CANCTL** register.

CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000
Offset 0x018
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	BRPE	R/W	0x0	Baud Rate Prescaler Extension 0x00-0x0F: Extend the BRP bit in the **CANBIT** register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by BRPE (MSBs) and BRP (LSBs).

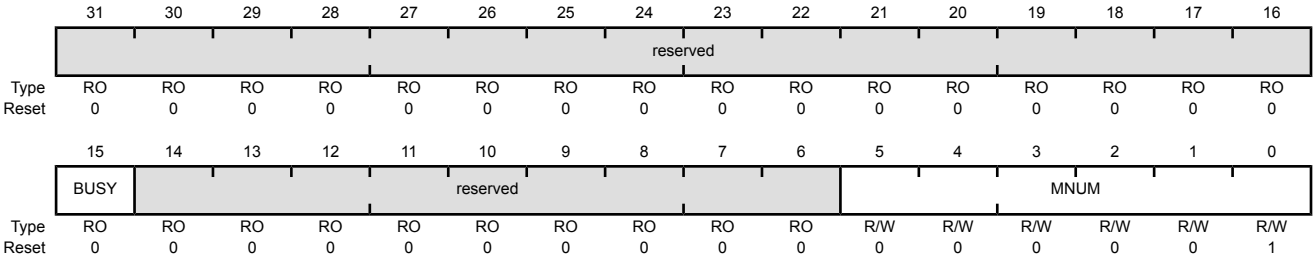
Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080

A message transfer is started as soon as there is a write of the message object number to the MNUM field when the TXRQST bit in the **CANIF1MCTL** register is set. With this write operation, the BUSY bit is automatically set to indicate that a transfer between the CAN Interface Registers and the internal message RAM is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the interface register and the message RAM completes, which then clears the BUSY bit.

CAN IF1 Command Request (CANIF1CRQ)

CAN0 base: 0x4004.0000
Offset 0x020
Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	BUSY	RO	0	Busy Flag 0: Cleared when read/write action has finished. 1: Set when a write occurs to the message number in this register.
14:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	MNUM	R/W	0x01	Message Number Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32.

Value	Description
0x00	Reserved 0 is not a valid message number; it is interpreted as 0x20, or object 32.
0x01-0x20	Message Number Indicates specified message object 1 to 32.
0x21-0x3F	Reserved Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.

Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024

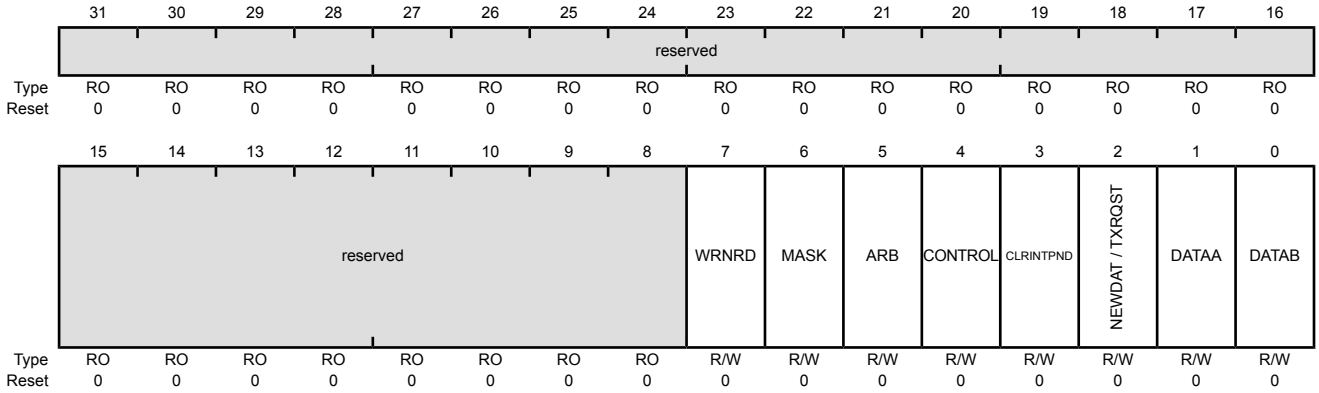
Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084

Reading the Command Mask registers provides status for various functions. Writing to the Command Mask registers specifies the transfer direction and selects which buffer registers are the source or target of the data transfer.

Note that when a read from the message object buffer occurs when the WRNRD bit is clear and the CLRINTPND and/or NEWDAT bits are set, the interrupt pending and/or new data flags in the message object buffer are cleared.

CAN IF1 Command Mask (CANIF1CMSK)

CAN0 base: 0x4004.0000
Offset 0x024
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	WRNRD	R/W	0	Write, Not Read Transfer the message object address specified by the **CAN Command Request (CANIFnCRQ)** register to the CAN message buffer registers. **Note:** Interrupt pending and new data conditions in the message buffer can be cleared by reading from the buffer (WRNRD = 0) when the CLRINTPND and/or NEWDAT bits are set.
6	MASK	R/W	0	Access Mask Bits 0: Mask bits unchanged. 1: Transfer IDMASK + DIR + MXTD of the message object into the Interface registers.
5	ARB	R/W	0	Access Arbitration Bits 0: Arbitration bits unchanged. 1: Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.

Bit/Field	Name	Type	Reset	Description
4	CONTROL	R/W	0	Access Control Bits 0: Control bits unchanged. 1: Transfer control bits from the **CANIFnMCTL** register into the Interface registers.
3	CLRINTPND	R/W	0	Clear Interrupt Pending Bit If WRNRD is set, this bit controls whether the INTPND bit in the **CANIFnMCTL** register is changed. 0: The INTPND bit in the message object remains unchanged. 1: The INTPND bit is cleared in the message object. If WRNRD is clear and this bit is clear, the interrupt pending status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is clear and this bit is set, the interrupt pending status is cleared in the message buffer. Note that the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing.
2	NEWDAT / TXRQST	R/W	0	NEWDAT / TXRQST Bit If WRNRD is set, this bit can act as a TXRQST bit and request a transmission. Note that when this bit is set, the TXRQST bit in the **CANIFnMCTL** register is ignored. 0: Transmission is not requested 1: Begin a transmission If WRNRD is clear and this bit is clear, the value of the new data status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is clear and this bit is set, the new data status is cleared in the message buffer. Note that the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing.
1	DATAA	R/W	0	Access Data Byte 0 to 3 When WRNRD = 1: 0: Data bytes 0-3 are unchanged. 1: Transfer data bytes 0-3 in message object to **CANIFnDA1** and **CANIFnDA2**. When WRNRD = 0: 0: Data bytes 0-3 are unchanged. 1: Transfer data bytes 0-3 in **CANIFnDA1** and **CANIFnDA2** to the message object.
0	DATAB	R/W	0	Access Data Byte 4 to 7 When WRNRD = 1: 0: Data bytes 4-7 are unchanged. 1: Transfer data bytes 4-7 in message object to **CANIFnDB1** and **CANIFnDB2**. When WRNRD = 0: 0: Data bytes 4-7 are unchanged. 1: Transfer data bytes 4-7 in **CANIFnDB1** and **CANIFnDB2** to the message object.

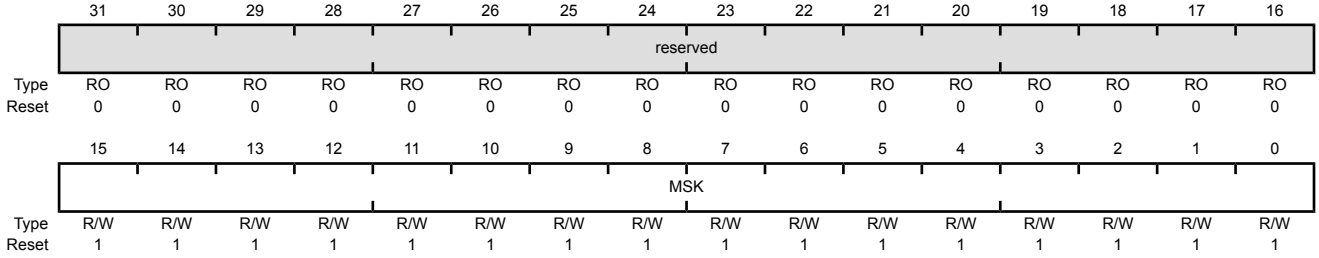
Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the ID bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

CAN IF1 Mask 1 (CANIF1MSK1)

CAN0 base: 0x4004.0000
Offset 0x028
Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MSK	R/W	0xFFFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [15:0] of the ID. The MSK field in the **CANIFnMSK2** register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored. 0: The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering. 1: The corresponding identifier field (ID) is used for acceptance filtering.

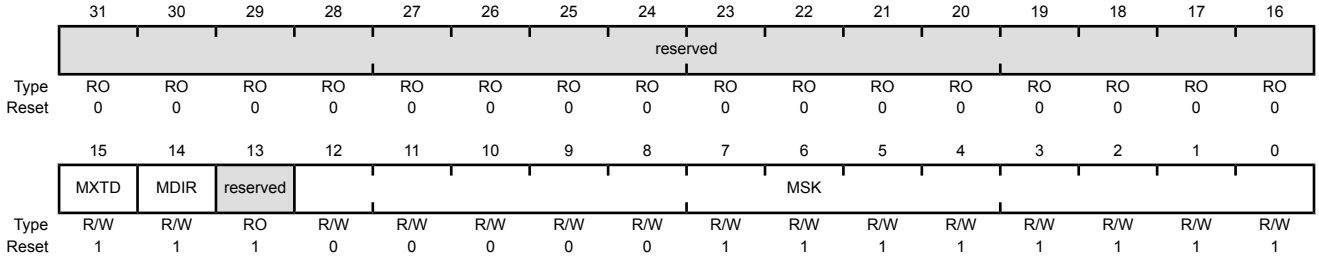
Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C

Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C

This register holds extended mask information that accompanies the **CANIFnMSK1** register.

CAN IF1 Mask 2 (CANIF1MSK2)

CAN0 base: 0x4004.0000
Offset 0x02C
Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MXTD	R/W	0x1	Mask Extended Identifier 0: The extended identifier bit (XTD in the **CANIFnARB2** register) has no effect on the acceptance filtering. 1: The extended identifier bit XTD is used for acceptance filtering.
14	MDIR	R/W	0x1	Mask Message Direction 0: The message direction bit (DIR in the **CANIFnARB2** register) has no effect for acceptance filtering. 1: The message direction bit DIR is used for acceptance filtering.
13	reserved	RO	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:0	MSK	R/W	0xFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The MSK field in the **CANIFnMSK1** register are used for bits [15:0] of the ID. When using an 11-bit identifier, MSK[12:2] are used for bits [10:0] of the ID. 0: The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering. 1: The corresponding identifier field (ID) is used for acceptance filtering.

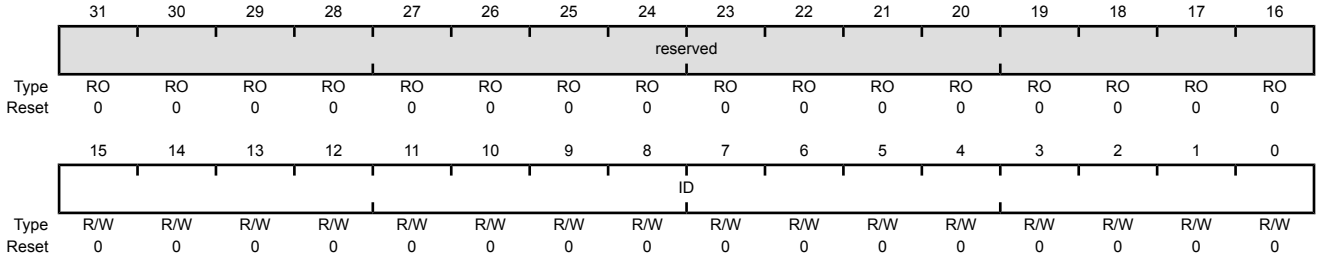
Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030

Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090

These registers hold the identifiers for acceptance filtering.

CAN IF1 Arbitration 1 (CANIF1ARB1)

CAN0 base: 0x4004.0000
Offset 0x030
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	ID	R/W	0x0000	Message Identifier This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, bits 15:0 of the **CANIFnARB1** register are [15:0] of the ID, while bits 12:0 of the **CANIFnARB2** register are [28:16] of the ID. When using an 11-bit identifier, these bits are not used.

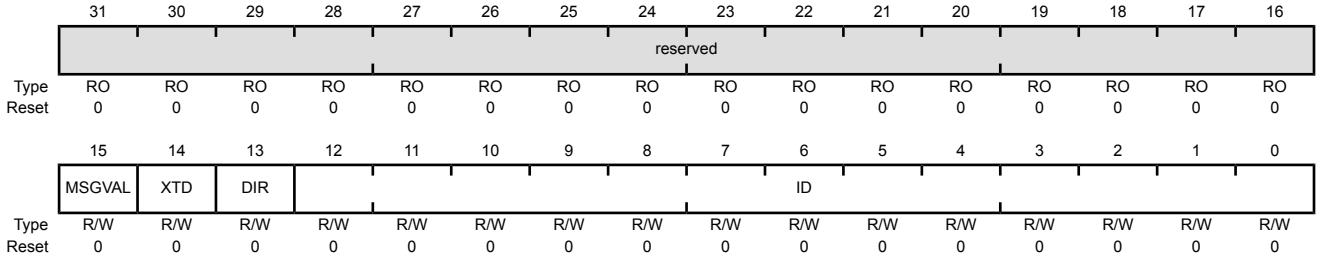
Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034

Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094

These registers hold information for acceptance filtering.

CAN IF1 Arbitration 2 (CANIF1ARB2)

CAN0 base: 0x4004.0000
Offset 0x034
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	MSGVAL	R/W	0	Message Valid 0: The message object is ignored by the message handler. 1: The message object is configured and ready to be considered by the message handler within the CAN controller. All unused message objects should have this bit cleared during initialization and before clearing the INIT bit in the **CANCTL** register. The MSGVAL bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the ID fields in the **CANIFnARBn** registers, the XTD and DIR bits in the **CANIFnARB2** register, or the DLC field in the **CANIFnMCTL** register.
14	XTD	R/W	0	Extended Identifier 0: An 11-bit Standard Identifier is used for this message object. 1: A 29-bit Extended Identifier is used for this message object.
13	DIR	R/W	0	Message Direction 0: Receive. When the TXRQST bit in the **CANIFnMCTL** register is set, a remote frame with the identifier of this message object is received. On reception of a data frame with matching identifier, that message is stored in this message object. 1: Transmit. When the TXRQST bit in the **CANIFnMCTL** register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TXRQST bit of this message object is set (if RMTEN=1).

Bit/Field	Name	Type	Reset	Description
12:0	ID	R/W	0x000	Message Identifier This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, `ID[15:0]` of the **CANIFnARB1** register are [15:0] of the ID, while these bits, `ID[12:0]`, are [28:16] of the ID. When using an 11-bit identifier, `ID[12:2]` are used for bits [10:0] of the ID. The `ID` field in the **CANIFnARB1** register is ignored.

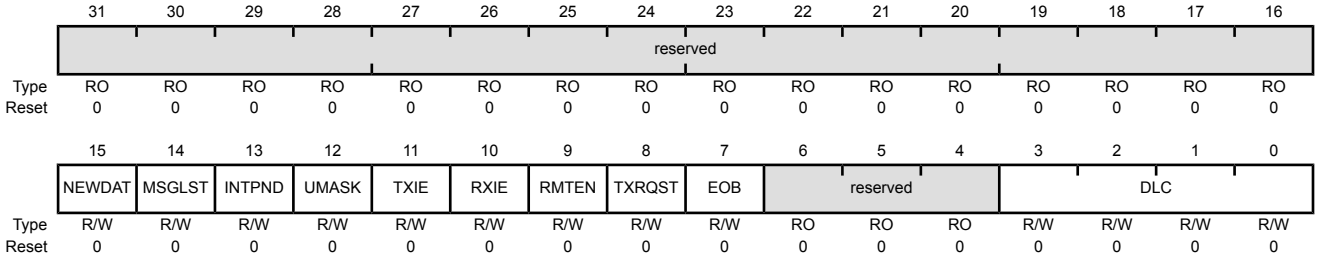
Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038

Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098

This register holds the control information associated with the message object to be sent to the Message RAM.

CAN IF1 Message Control (CANIF1MCTL)

CAN0 base: 0x4004.0000
Offset 0x038
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	NEWDAT	R/W	0	New Data 0: No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU. 1: The message handler or the CPU has written new data into the data portion of this message object.
14	MSGLST	R/W	0	Message Lost 0 : No message was lost since the last time this bit was cleared by the CPU. 1: The message handler stored a new message into this object when NEWDAT was set; the CPU has lost a message. This bit is only valid for message objects when the DIR bit in the **CANIFnARB2** register clear (receive).
13	INTPND	R/W	0	Interrupt Pending 0: This message object is not the source of an interrupt. 1: This message object is the source of an interrupt. The interrupt identifier in the **CANINT** register points to this message object if there is not another interrupt source with a higher priority.
12	UMASK	R/W	0	Use Acceptance Mask 0: Mask ignored. 1: Use mask (MSK, MXTD, and MDIR bits in the **CANIFnMSKn** registers) for acceptance filtering.

Bit/Field	Name	Type	Reset	Description
11	TXIE	R/W	0	Transmit Interrupt Enable 0: The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful transmission of a frame. 1: The INTPND bit in the **CANIFnMCTL** register is set after a successful transmission of a frame.
10	RXIE	R/W	0	Receive Interrupt Enable 0: The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful reception of a frame. 1: The INTPND bit in the **CANIFnMCTL** register is set after a successful reception of a frame.
9	RMTEN	R/W	0	Remote Enable 0: At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is left unchanged. 1: At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is set.
8	TXRQST	R/W	0	Transmit Request 0: This message object is not waiting for transmission. 1: The transmission of this message object is requested and is not yet done.
7	EOB	R/W	0	End of Buffer 0: Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer. 1: Single message object or last message object of a FIFO Buffer. This bit is used to concatenate two or more message objects (up to 32) to build a FIFO buffer. For a single message object (thus not belonging to a FIFO buffer), this bit must be set.
6:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	DLC	R/W	0x0	Data Length Code Value: Description 0x0-0x8: Specifies the number of bytes in the data frame. 0x9-0xF: Defaults to a data frame with 8 bytes. The DLC field in the **CANIFnMCTL** register of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it writes DLC to the value given by the received message.

Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C

Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040

Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044

Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048

Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C

Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0

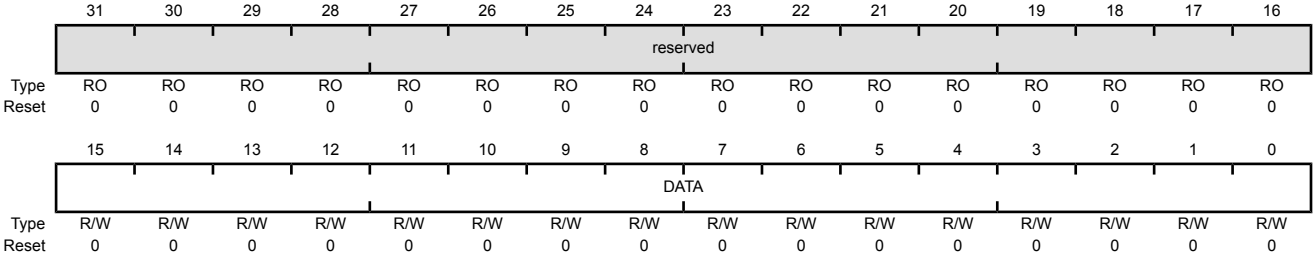
Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4

Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8

These registers contain the data to be sent or that has been received. In a CAN data frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

CAN IF1 Data A1 (CANIF1DA1)

CAN0 base: 0x4004.0000
Offset 0x03C
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	Data The **CANIFnDA1** registers contain data bytes 1 and 0; **CANIFnDA2** data bytes 3 and 2; **CANIFnDB1** data bytes 5 and 4; and **CANIFnDB2** data bytes 7 and 6.

Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100

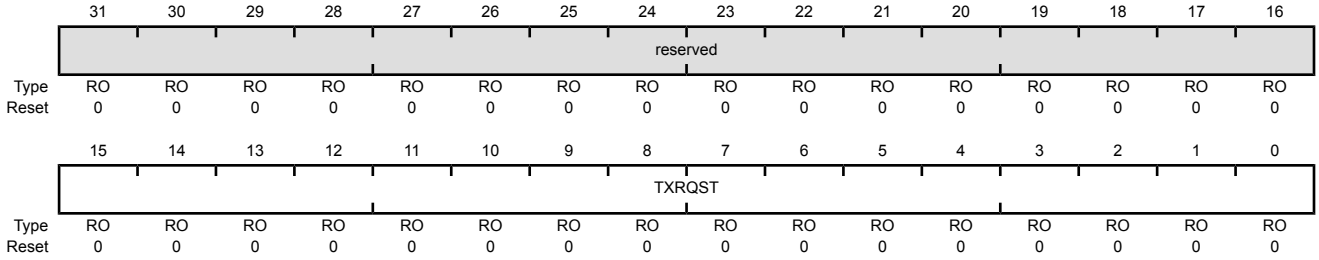
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104

The **CANTXRQ1** and **CANTXRQ2** registers hold the TXRQST bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TXRQST bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a remote frame, or (3) the message handler state machine after a successful transmission.

The **CANTXRQ1** register contains the TXRQST bits of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TXRQST bits of the second 16 message objects.

CAN Transmission Request 1 (CANTXRQ1)

CAN0 base: 0x4004.0000
Offset 0x100
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TXRQST	RO	0x0000	Transmission Request Bits 0: The corresponding message object is not waiting for transmission. 1: The transmission of the corresponding message object is requested and is not yet done.

Register 32: CAN New Data 1 (CANNWDA1), offset 0x120

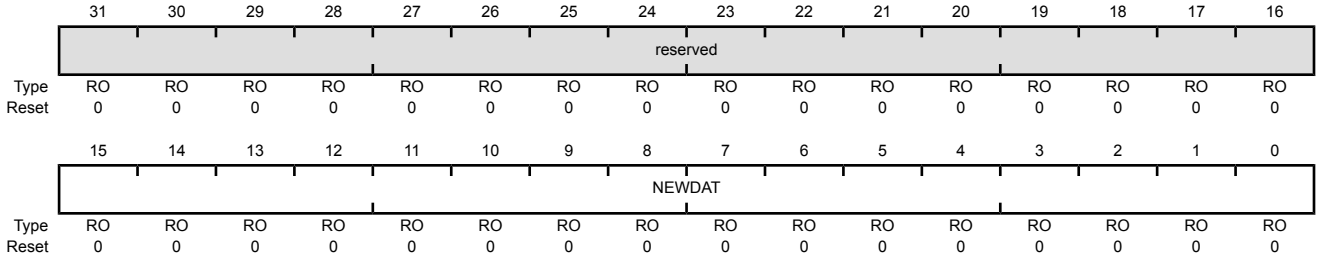
Register 33: CAN New Data 2 (CANNWDA2), offset 0x124

The **CANNWDA1** and **CANNWDA2** registers hold the NEWDAT bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The NEWDAT bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a data frame, or (3) the message handler state machine after a successful transmission.

The **CANNWDA1** register contains the NEWDAT bits of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the NEWDAT bits of the second 16 message objects.

CAN New Data 1 (CANNWDA1)

CAN0 base: 0x4004.0000
Offset 0x120
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	NEWDAT	RO	0x0000	New Data Bits 0: No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU. 1: The message handler or the CPU has written new data into the data portion of the corresponding message object.

Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

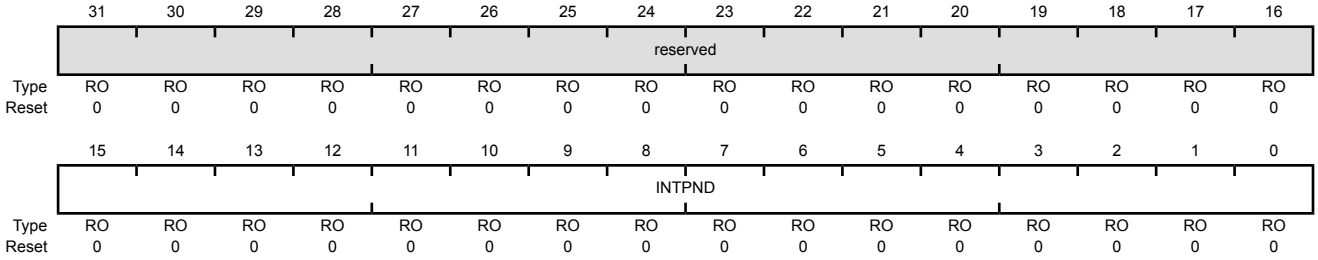
The **CANMSG1INT** and **CANMSG2INT** registers hold the INTPND bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The INTPND bit of a specific message object can be changed through two sources: (1) the CPU via the **CANIFnMCTL** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CANINT** register.

The **CANMSG1INT** register contains the INTPND bits of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the INTPND bits of the second 16 message objects.

CAN Message 1 Interrupt Pending (CANMSG1INT)

CAN0 base: 0x4004.0000
Offset 0x140
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	INTPND	RO	0x0000	Interrupt Pending Bits 0: The corresponding message object is not the source of an interrupt. 1: The corresponding message object is the source of an interrupt.

Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160

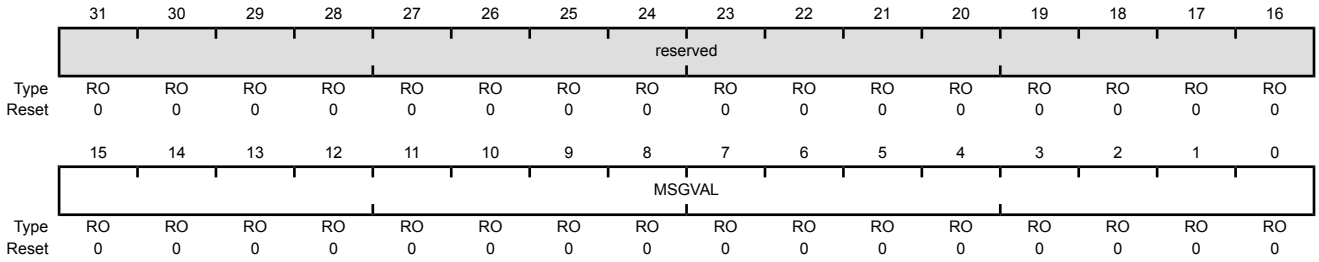
Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the MSGVAL bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message value of a specific message object can be changed with the **CANIFnMCTL** register.

The **CANMSG1VAL** register contains the MSGVAL bits of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the MSGVAL bits of the second 16 message objects in the message RAM.

CAN Message 1 Valid (CANMSG1VAL)

CAN0 base: 0x4004.0000
Offset 0x160
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MSGVAL	RO	0x0000	Message Valid Bits 0: The corresponding message object is not configured and is ignored by the message handler. 1: The corresponding message object is configured and should be considered by the message handler.

16 Ethernet Controller

The Stellaris® Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface. The Ethernet Controller conforms to *IEEE 802.3* specifications and fully supports 10BASE-T and 100BASE-TX standards.

The Stellaris Ethernet Controller module has the following features:

- Conforms to the *IEEE 802.3-2002 specification*
  - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line
  - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
  - Full-featured auto-negotiation
- Multiple operational modes
  - Full- and half-duplex 100 Mbps
  - Full- and half-duplex 10 Mbps
  - Power-saving and power-down modes
- Highly configurable
  - Programmable MAC address
  - LED activity selection
  - Promiscuous mode support
  - CRC error-rejection control
  - User-configurable interrupts
- Physical media manipulation
  - Automatic MDI/MDI-X cross-over correction
  - Register-programmable transmit amplitude
  - Automatic polarity correction and 10BASE-T signal reception
- IEEE 1588 Precision Time Protocol - provides highly accurate time stamps for individual packets

16.1 Block Diagram

As shown in Figure 16-1 on page 596, the Ethernet Controller is functionally divided into two layers: the Media Access Controller (MAC) layer and the Network Physical (PHY) layer. These layers correspond to the OSI model layers 2 and 1. The CPU accesses the Ethernet Controller via the MAC layer. The MAC layer provides transmit and receive processing for Ethernet frames. The MAC layer also provides the interface to the PHY layer via an internal Media Independent Interface (MII). The PHY layer communicates with the Ethernet bus.

Figure 16-1. Ethernet Controller

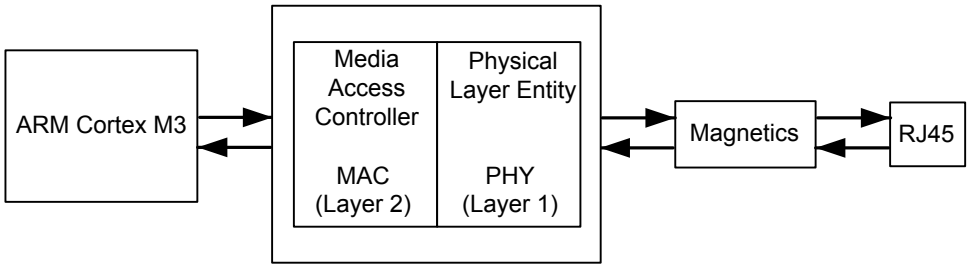
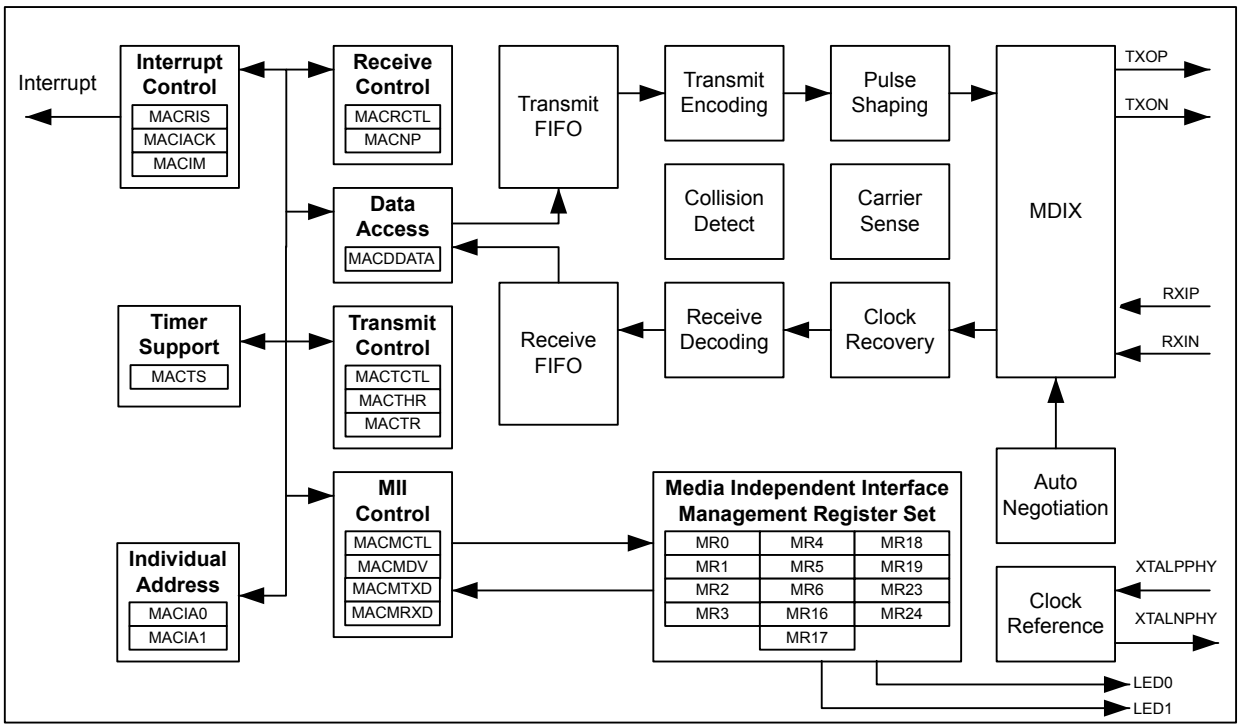


Figure 16-2 on page 596 shows more detail of the internal structure of the Ethernet Controller and how the register set relates to various functions.

Figure 16-2. Ethernet Controller Block Diagram



16.2 Signal Description

Table 16-1 on page 597 and Table 16-2 on page 597 list the external signals of the Ethernet Controller and describe the function of each. The Ethernet LED signals are alternate functions for GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Assignment" lists the GPIO pin placement for the LED signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the LED function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 16-1. Ethernet Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ERBIAS	41	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
GNDPHY	42 85 86	-	Power	GND of the Ethernet PHY.
LED0	59	O	TTL	Ethernet LED 0.
LED1	60	O	TTL	Ethernet LED 1.
MDIO	58	I/O	TTL	MDIO of the Ethernet PHY.
RXIN	37	I	Analog	RXIN of the Ethernet PHY.
RXIP	40	I	Analog	RXIP of the Ethernet PHY.
TXON	46	O	Analog	TXON of the Ethernet PHY.
TXOP	43	O	Analog	TXOP of the Ethernet PHY.
VCCPHY	36 83 84	-	Power	VCC of the Ethernet PHY.
XTALNPHY	17	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
XTALPPHY	16	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

Table 16-2. Ethernet Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ERBIAS	K3	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
GNDPHY	C8 C9 K4	-	Power	GND of the Ethernet PHY.
LED0	J12	O	TTL	Ethernet LED 0.
LED1	J11	O	TTL	Ethernet LED 1.
MDIO	L9	I/O	TTL	MDIO of the Ethernet PHY.
RXIN	L7	I	Analog	RXIN of the Ethernet PHY.
RXIP	M7	I	Analog	RXIP of the Ethernet PHY.
TXON	L8	O	Analog	TXON of the Ethernet PHY.
TXOP	M8	O	Analog	TXOP of the Ethernet PHY.
VCCPHY	C10 D10 D11	-	Power	VCC of the Ethernet PHY.
XTALNPHY	J1	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
XTALPPHY	J2	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

16.3 Functional Description

Note: A 12.4-kΩ resistor should be connected between the ERBIAS and ground. The 12.4-kΩ resistor should have a 1% tolerance and should be located in close proximity to the ERBIAS pin. Power dissipation in the resistor is low, so a chip resistor of any geometry may be used.

The functional description of the Ethernet Controller is discussed in the following sections.

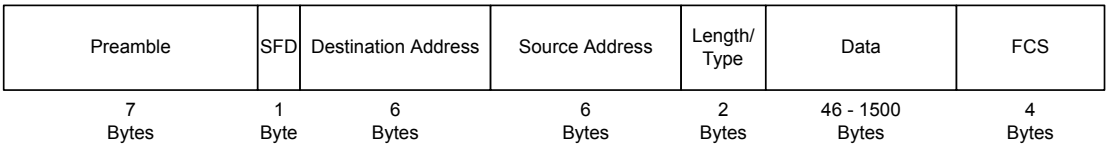
16.3.1 MAC Operation

The following sections decribe the operation of the MAC unit, including an overview of the Ethernet frame format, the MAC layer FIFOs, Ethernet transmission and reception options, packet timestamps, and LED indicators.

16.3.1.1 Ethernet Frame Format

Ethernet data is carried by Ethernet frames. The basic frame format is shown in Figure 16-3 on page 598.

Figure 16-3. Ethernet Frame



The seven fields of the frame are transmitted from left to right. The bits within the frame are transmitted from least to most significant bit.

- Preamble

  The Preamble field is used to synchronize with the received frame's timing. The preamble is 7 octets long.

- Start Frame Delimiter (SFD)

  The SFD field follows the preamble pattern and indicates the start of the frame. Its value is 1010.1011.

- Destination Address (DA)

  This field specifies destination addresses for which the frame is intended. The LSB (bit 16 of DA oct 1 in the frame, see Table 16-3 on page 600) of the DA determines whether the address is an individual (0), or group/multicast (1) address.

- Source Address (SA)

  The source address field identifies the station from which the frame was initiated.

- Length/Type Field

  The meaning of this field depends on its numeric value. This field can be interpreted as length or type code. The maximum length of the data field is 1500 octets. If the value of the Length/Type field is less than or equal to 1500 decimal, it indicates the number of MAC client data octets. If the value of this field is greater than or equal to 1536 decimal, then it is type interpretation. The meaning of the Length/Type field when the value is between 1500 and 1536 decimal is unspecified by the IEEE 802.3 standard. However, the Ethernet Controller assumes type interpretation if the

value of the Length/Type field is greater than 1500 decimal. The definition of the Type field is specified in the IEEE 802.3 standard. The first of the two octets in this field is most significant.

- Data

  The data field is a sequence of octets that is at least 46 in length, up to 1500 in length. Full data transparency is provided so any values can appear in this field. A minimum frame size of 46 octets is required to meet the IEEE standard. If the frame size is too small, the Ethernet Controller automatically appends extra bits (a pad), thus the pad can have a size of 0 to 46 octets. Data padding can be disabled by clearing the PADEN bit in the **Ethernet MAC Transmit Control (MACTCTL)** register.

  For the Ethernet Controller, data sent/received can be larger than 1500 bytes without causing a Frame Too Long error. Instead, a FIFO overrun error is reported using the FOV bit in the **Ethernet MAC Raw Interrupt Status(MACRIS)** register when the frame received is too large to fit into the Ethernet Controller's 2K RAM.

- Frame Check Sequence (FCS)

  The frame check sequence carries the cyclic redundancy check (CRC) value. The CRC is computed over the destination address, source address, length/type, and data (including pad) fields using the CRC-32 algorithm. The Ethernet Controller computes the FCS value one nibble at a time. For transmitted frames, this field is automatically inserted by the MAC layer, unless disabled by clearing the CRC bit in the **MACTCTL** register. For received frames, this field is automatically checked. If the FCS does not pass, the frame is not placed in the RX FIFO, unless the FCS check is disabled by clearing the BADCRC bit in the **MACRCTL** register.

16.3.1.2 MAC Layer FIFOs

The Ethernet Controller is capable of simultaneous transmission and reception. This feature is enabled by setting the DUPLEX bit in the **MACTCTL** register.

For Ethernet frame transmission, a 2 KB transmit FIFO is provided that can be used to store a single frame. While the *IEEE 802.3 specification* limits the size of an Ethernet frame's payload section to 1500 Bytes, the Ethernet Controller places no such limit. The full buffer can be used, for a payload of up to 2032 bytes (as the first 16 bytes in the FIFO are reserved for destination address, source address and length/type information).

For Ethernet frame reception, a 2-KB receive FIFO is provided that can be used to store multiple frames, up to a maximum of 31 frames. If a frame is received, and there is insufficient space in the RX FIFO, an overflow error is indicated using the FOV bit in the **MACRIS** register.

For details regarding the TX and RX FIFO layout, refer to Table 16-3 on page 600. Please note the following difference between TX and RX FIFO layout. For the TX FIFO, the Data Length field in the first FIFO word refers to the Ethernet frame data payload, as shown in the 5th to nth FIFO positions. For the RX FIFO, the Frame Length field is the total length of the received Ethernet frame, including the Length/Type bytes and the FCS bits.

If FCS generation is disabled by clearing the CRC bit in the **MACTCTL** register, the last word in the TX FIFO must contain the FCS bytes for the frame that has been written to the FIFO.

Also note that if the length of the data payload section is not a multiple of 4, the FCS field is not be aligned on a word boundary in the FIFO. However, for the RX FIFO the beginning of the next frame is always on a word boundary.

Table 16-3. TX & RX FIFO Organization

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
1st	7:0	Data Length Least Significant Byte	Frame Length Least Significant Byte
	15:8	Data Length Most Significant Byte	Frame Length Most Significant Byte
	23:16	DA oct 1	
	31:24	DA oct 2	
2nd	7:0	DA oct 3	
	15:8	DA oct 4	
	23:16	DA oct 5	
	31:24	DA oct 6	
3rd	7:0	SA oct 1	
	15:8	SA oct 2	
	23:16	SA oct 3	
	31:24	SA oct 4	
4th	7:0	SA oct 5	
	15:8	SA oct 6	
	23:16	Len/Type Most Significant Byte	
	31:24	Len/Type Least Significant Byte	
5th to nth	7:0	data oct n	
	15:8	data oct n+1	
	23:16	data oct n+2	
	31:24	data oct n+3	
last	7:0	FCS 1	
	15:8	FCS 2	
	23:16	FCS 3	
	31:24	FCS 4	

Note: If the CRC bit in the **MACTCTL** register is clear, the FCS bytes must be written with the correct CRC. If the CRC bit is set, the Ethernet Controller automatically writes the FCS bytes.

16.3.1.3 Ethernet Transmission Options

At the MAC layer, the transmitter can be configured for both full-duplex and half-duplex operation by using the DUPLEX bit in the **MACTCTL** register.

The Ethernet Controller automatically generates and inserts the Frame Check Sequence (FCS) at the end of the transmit frame when the CRC bit in the **MACTCTL** register is set. However, for test purposes, this feature can be disabled in order to generate a frame with an invalid CRC by clearing the CRC bit.

The *IEEE 802.3 specification* requires that the Ethernet frame payload section be a minimum of 46 bytes. The Ethernet Controller automatically pads the data section if the payload data section loaded into the FIFO is less than the minimum 46 bytes when the PADEN bit in the **MACTCTL** register is set. This feature can be disabled by clearing the PADEN bit.

The transmitter must be enabled by setting the TXEN bit in the **TCTL** register.

16.3.1.4 Ethernet Reception Options

The Ethernet Controller RX FIFO should be cleared during software initialization. The receiver should first be disabled by clearing the RXEN bit in the **Ethernet MAC Receive Control (MACRCTL)** register, then the FIFO can be cleared by setting the RSTFIFO bit in the **MACRCTL** register.

The receiver automatically rejects frames that contain bad CRC values in the FCS field. In this case, a Receive Error interrupt is generated and the receive data is lost. To accept all frames, clear the BADCRC bit in the **MACRCTL** register.

In normal operating mode, the receiver accepts only those frames that have a destination address that matches the address programmed into the **Ethernet MAC Individual Address 0 (MACIA0)** and **Ethernet MAC Individual Address 1 (MACIA1)** registers. However, the Ethernet receiver can also be configured for Promiscuous and Multicast modes by setting the PRMS and AMUL bits in the **MACRCTL** register.

16.3.1.5 Packet Timestamps

For applications requiring very high-precision synchronization packets, the Ethernet Controller provides a means of generating precision timestamps in support of the IEEE Precision Time Protocol (IEEE-1588). This feature is enabled by setting the TSEN bit in the **Ethernet MAC Timer Support (MATCS)** register. Note that when this feature is enabled, General-Purpose Timer 3 (GPT3) must be dedicated to the Ethernet Controller. GPT3 must be configured to 16-bit edge capture mode, see page 347. Timer A of GPT3 stores the transmit time, and Timer B stores the receive time. One other General-Purpose Timer can be set up as a 16-bit free-running timer to synchronize the receiver and transmitter timers and provide a timestamp with which to compare the timestamps stored in GPT3.

16.3.2 Internal MII Operation

For the MII management interface to function properly, the MDIO signal must be connected through a 10k Ω pull-up resistor to the +3.3 V supply. Failure to connect this pull-up resistor prevents management transactions on this internal MII to function. Note that it is possible for data transmission across the MII to still function since the PHY layer auto-negotiates the link parameters by default.

For the MII management interface to function properly, the internal clock must be divided down from the system clock to a frequency no greater than 2.5 MHz. The **Ethernet MAC Management Divider (MACMDV)** register contains the divider used for scaling down the system clock. See page 620 for more details about the use of this register.

16.3.3 PHY Operation

The Physical Layer (PHY) in the Ethernet Controller includes integrated ENDECs, scrambler/descrambler, dual-speed clock recovery, and full-featured auto-negotiation functions. The transmitter includes an on-chip pulse shaper and a low-power line driver. The receiver has an adaptive equalizer and a baseline restoration circuit required for accurate clock and data recovery. The transceiver interfaces to Category-5 unshielded twisted pair (Cat-5 UTP) cabling for 100BASE-TX applications, and Category-3 unshielded twisted pair (Cat-3 UTP) for 10BASE-T applications. The Ethernet Controller is connected to the line media via dual 1:1 isolation transformers. No external filter is required.

16.3.3.1 Clock Selection

The Ethernet Controller has an on-chip crystal oscillator which can also be driven by an external oscillator. In this mode of operation, a 25-MHz crystal should be connected between the XTALPPHY

and XTALNPHY pins. Alternatively, an external 25-MHz clock input can be connected to the XTALPPHY pin. In this mode of operation, a crystal is not required and the XTALNPHY pin must be tied to ground.

16.3.3.2 Auto-Negotiation

The Ethernet Controller supports the auto-negotiation functions of Clause 28 of the *IEEE 802.3* standard for 10/100 Mbps operation over copper wiring. This function is controlled via register settings. The auto-negotiation function is turned on by default, and the ANEGEN bit in the **Ethernet PHY Management Register 0 - Control (MR0)** is set after reset. Software can disable the auto-negotiation function by clearing the ANEGEN bit. The contents of the **Ethernet PHY Management Register - Auto-Negotiation Advertisement (MR4)** are reflected to the Ethernet Controller's link partner during auto-negotiation via fast-link pulse coding.

Once auto-negotiation is complete, the DPLX and RATE bits in the **Ethernet PHY Management Register 18 - Diagnostic (MR18)** register reflect the actual speed and duplex condition. If auto-negotiation fails to establish a link for any reason, the ANEGF bit in the **MR18** register reflects this and auto-negotiation restarts from the beginning. Setting the RANEG bit in the **MR0** register also causes auto-negotiation to restart.

16.3.3.3 Polarity Correction

The Ethernet Controller is capable of either automatic or manual polarity reversal for 10BASE-T and auto-negotiation functions. Bits 4 and 5 (RVSPOL and APOL) in the **Ethernet PHY Management Register 16 - Vendor-Specific (MR16)** control this feature. The default is automatic mode, where APOL is clear and RVSPOL indicates if the detection circuitry has inverted the input signal. To enter manual mode, APOL should be set. In manual mode RVSPOL controls the signal polarity.

16.3.3.4 MDI/MDI-X Configuration

The Ethernet Controller supports the MDI/MDI-X configuration as defined in *IEEE 802.3-2002 specification*. The MDI/MDI-X configuration eliminates the need for cross-over cables when connecting to another device, such as a hub. The algorithm is controlled via settings in the **Ethernet PHY Management Register 24 - MDI/MIDIX Control (MR24)**. Refer to page 643 for additional details about these settings.

16.3.3.5 Power Management

The PHY has two power-saving modes:

- Power-Down
- Receive Power Management

Power-down mode is activated by setting the PWRDN bit in the **MR0** register. When the PHY is in power-down mode, it consumes minimum power. While in the power-down state, the Ethernet Controller still responds to management transactions.

Receive power management (RXCC mode) is activated by setting the RXCC bit in the **MR16** register. In this mode of operation, the adaptive equalizer, the clock recovery phase lock loop (PLL), and all other receive circuitry are powered down. As soon as a valid signal is detected, all circuits are automatically powered up to resume normal operation. Note that the RXCC mode is not supported during 10BASE-T operation.

16.3.3.6 LED Indicators

The Ethernet Controller supports two LED signals that can be used to indicate various states of operation. These signals are mapped to the LED0 and LED1 pins. By default, these pins are configured as GPIO signals (PF3 and PF2). For the PHY layer to drive these signals, they must be reconfigured to their alternate function. See “General-Purpose Input/Outputs (GPIOs)” on page 286 for additional details. The function of these pins is programmable via the PHY layer **Ethernet PHY Management Register 23 - LED Configuration (MR23)**. Refer to page 642 for additional details on how to program these LED functions.

16.3.4 Interrupts

The Ethernet Controller can generate an interrupt for one or more of the following conditions:

- A frame has been received into an empty RX FIFO
- A frame transmission error has occurred
- A frame has been transmitted successfully
- A frame has been received with inadequate room in the RX FIFO (overrun)
- A frame has been received with one or more error conditions (for example, FCS failed)
- An MII management transaction between the MAC and PHY layers has completed
- One or more of the following PHY layer conditions occurs:
  - Auto-Negotiate Complete
  - Remote Fault
  - Link Status Change
  - Link Partner Acknowledge
  - Parallel Detect Fault
  - Page Received
  - Receive Error
  - Jabber Event Detected

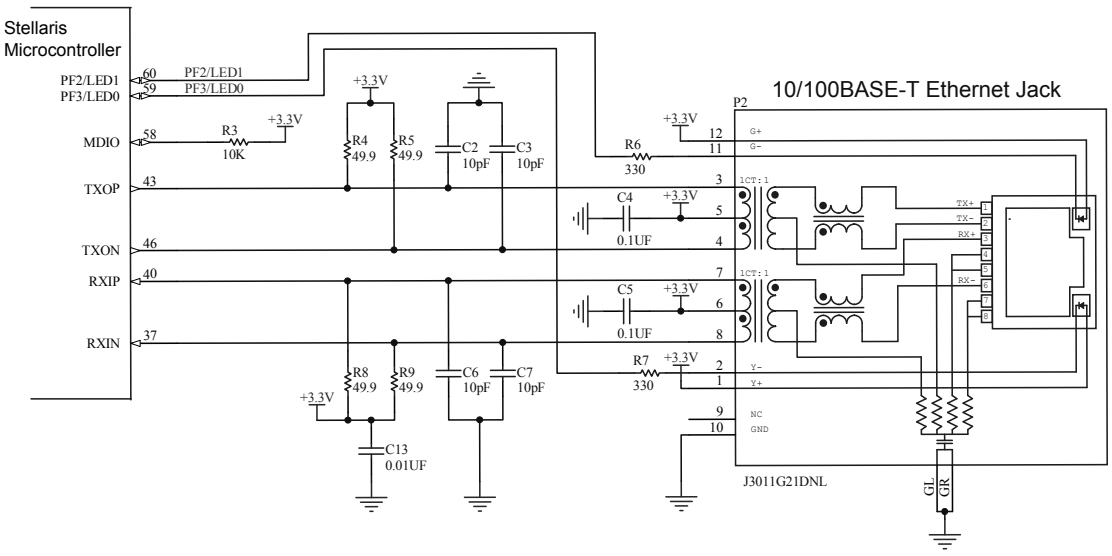
16.4 Initialization and Configuration

The following sections describe the hardware and software configuration required to set up the Ethernet Controller.

16.4.1 Hardware Configuration

Figure 16-4 on page 604 shows the proper method for interfacing the Ethernet Controller to a 10/100BASE-T Ethernet jack.

Figure 16-4. Interface to an Ethernet Jack



The following isolation transformers have been tested and are known to successfully interface to the Ethernet PHY layer.

- Isolation Transformers
  - TDK TLA-6T103
  - Bel-Fuse S558-5999-46
  - Halo TG22-3506ND
  - Pulse PE-68515
  - Valor ST6118
  - YCL 20PMT04
- Isolation transformers in low profile packages (0.100 in/2.5 mm or less)
  - TDK TLA-6T118
  - Halo TG110-S050
  - PCA EPF8023G
- Isolation transformers with integrated RJ45 connector
  - TDK TLA-6T704
  - Delta RJS-1A08T089A
- Isolation transformers with integrated RJ45 connector, LEDs and termination resistors
  - Pulse J0011D21B/E
  - Pulse J3011G21DNL

16.4.2 Software Configuration

To use the Ethernet Controller, it must be enabled by setting the EPHY0 and EMAC0 bits in the **RCGC2** register (see page 228). The following steps can then be used to configure the Ethernet Controller for basic operation.

1. Program the **MACDIV** register to obtain a 2.5 MHz clock (or less) on the internal MII. Assuming a 20-MHz system clock, the **MACDIV** value should be 0x03 or greater.

2. Program the **MACIA0** and **MACIA1** register for address filtering.

3. Program the **MACTCTL** register for Auto CRC generation, padding, and full-duplex operation using a value of 0x16.

4. Program the **MACRCTL** register to flush the receive FIFO and reject frames with bad FCS using a value of 0x18.

5. Enable both the Transmitter and Receive by setting the LSB in both the **MACTCTL** and **MACRCTL** registers.

6. To transmit a frame, write the frame into the TX FIFO using the **Ethernet MAC Data (MACDATA)** register. Then set the NEWTX bit in the **Ethernet Mac Transmission Request (MACTR)** register to initiate the transmit process. When the NEWTX bit has been cleared, the TX FIFO is available for the next transmit frame.

7. To receive a frame, wait for the NPR field in the **Ethernet MAC Number of Packets (MACNP)** register to be non-zero. Then begin reading the frame from the RX FIFO by using the **MACDATA** register. To ensure that the entire packet is received, either use the DriverLib EthernetPacketGet() API or compare the number of bytes received to the Length field from the frame to determine when the packet has been completely read.

16.5 Ethernet Register Map

Table 16-4 on page 605 lists the Ethernet MAC registers. All addresses given are relative to the Ethernet MAC base address of 0x4004.8000. Note that the Ethernet module clock must be enabled before the registers can be programmed (see page 228). There must be a delay of 3 system clocks after the Ethernet module clock is enabled before any Ethernet module registers are accessed.

The *IEEE 802.3* standard specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers and are detailed in Section 22.2.4 of the *IEEE 802.3 specification*. Table 16-4 on page 605 also lists these MII Management registers. *All addresses given are absolute and are written directly to the REGADR field of the **Ethernet MAC Management Control (MACMCTL)** register.* The format of registers 0 to 15 are defined by the IEEE specification and are common to all PHY layer implementations. The only variance allowed is for features that may or may not be supported by a specific PHY implementation. Registers 16 to 31 are vendor-specific registers, used to support features that are specific to a vendor's PHY implementation. Vendor-specific registers not listed are reserved.

Table 16-4. Ethernet Register Map

Offset	Name	Type	Reset	Description	See page
Ethernet MAC					
0x000	MACRIS/MACIACK	R/W1C	0x0000.0000	Ethernet MAC Raw Interrupt Status/Acknowledge	607
0x004	MACIM	R/W	0x0000.007F	Ethernet MAC Interrupt Mask	610
0x008	MACRCTL	R/W	0x0000.0008	Ethernet MAC Receive Control	611
0x00C	MACTCTL	R/W	0x0000.0000	Ethernet MAC Transmit Control	612
0x010	MACDATA	R/W	0x0000.0000	Ethernet MAC Data	613
0x014	MACIA0	R/W	0x0000.0000	Ethernet MAC Individual Address 0	615
0x018	MACIA1	R/W	0x0000.0000	Ethernet MAC Individual Address 1	616
0x01C	MACTHR	R/W	0x0000.003F	Ethernet MAC Threshold	617
0x020	MACMCTL	R/W	0x0000.0000	Ethernet MAC Management Control	619

Table 16-4. Ethernet Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x024	MACMDV	R/W	0x0000.0080	Ethernet MAC Management Divider	620
0x02C	MACMTXD	R/W	0x0000.0000	Ethernet MAC Management Transmit Data	621
0x030	MACMRXD	R/W	0x0000.0000	Ethernet MAC Management Receive Data	622
0x034	MACNP	RO	0x0000.0000	Ethernet MAC Number of Packets	623
0x038	MACTR	R/W	0x0000.0000	Ethernet MAC Transmission Request	624
0x03C	MACTS	R/W	0x0000.0000	Ethernet MAC Timer Support	625
MII Management					
-	MR0	R/W	0x3100	Ethernet PHY Management Register 0 – Control	626
-	MR1	RO	0x7849	Ethernet PHY Management Register 1 – Status	628
-	MR2	RO	0x000E	Ethernet PHY Management Register 2 – PHY Identifier 1	630
-	MR3	RO	0x7237	Ethernet PHY Management Register 3 – PHY Identifier 2	631
-	MR4	R/W	0x01E1	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement	632
-	MR5	RO	0x0000	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability	634
-	MR6	RO	0x0000	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion	635
-	MR16	R/W	0x0140	Ethernet PHY Management Register 16 – Vendor-Specific	636
-	MR17	R/W	0x0000	Ethernet PHY Management Register 17 – Interrupt Control/Status	638
-	MR18	RO	0x0000	Ethernet PHY Management Register 18 – Diagnostic	640
-	MR19	R/W	0x4000	Ethernet PHY Management Register 19 – Transceiver Control	641
-	MR23	R/W	0x0010	Ethernet PHY Management Register 23 – LED Configuration	642
-	MR24	R/W	0x00C0	Ethernet PHY Management Register 24 –MDI/MDIX Control	643

16.6 Ethernet MAC Register Descriptions

The remainder of this section lists and describes the Ethernet MAC registers, in numerical order by address offset. Also see “MII Management Register Descriptions” on page 625.

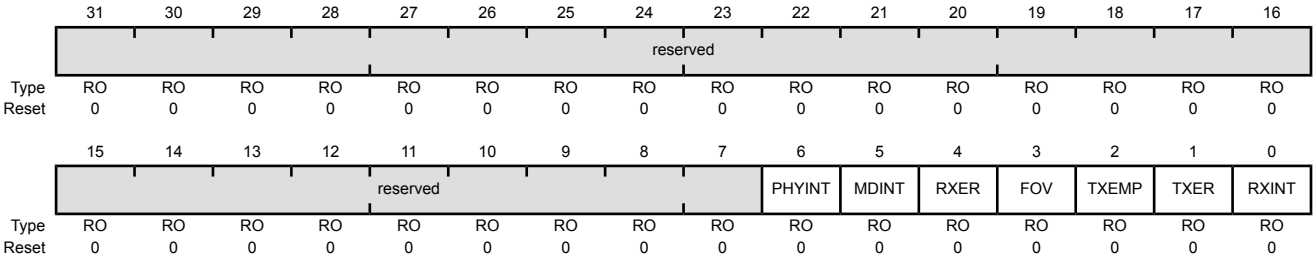
Register 1: Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK), offset 0x000

The **MACRIS/MACIACK** register is the interrupt status and acknowledge register. On a read, this register gives the current status value of the corresponding interrupt prior to masking. On a write, setting any bit clears the corresponding interrupt status bit.

Reads

Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK)

Base 0x4004.8000
Offset 0x000
Type RO, reset 0x0000.0000



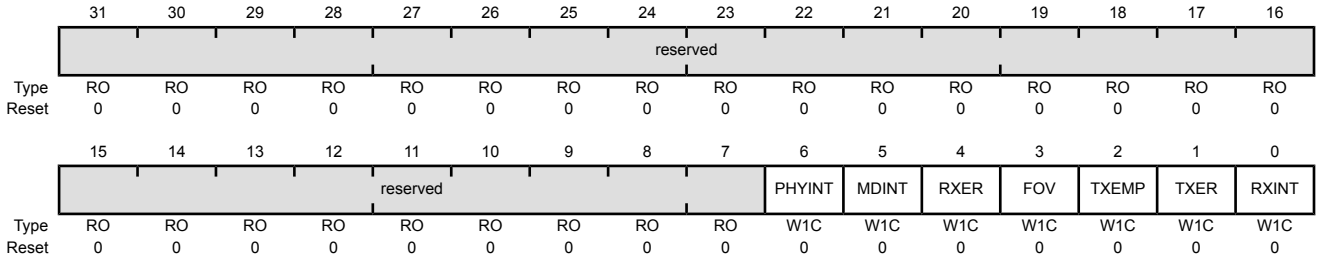
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINT	RO	0	PHY Interrupt When set, indicates that an enabled interrupt in the PHY layer has occurred. **MR17** in the PHY must be read to determine the specific PHY event that triggered this interrupt.
5	MDINT	RO	0	MII Transaction Complete When set, indicates that a transaction (read or write) on the MII interface has completed successfully.
4	RXER	RO	0	Receive Error This bit indicates that an error was encountered on the receiver. The possible errors that can cause this interrupt bit to be set are: ■ A receive error occurs during the reception of a frame (100 Mb/s only). ■ The frame is not an integer number of bytes (dribble bits) due to an alignment error. ■ The CRC of the frame does not pass the FCS check. ■ The length/type field is inconsistent with the frame data size when interpreted as a length field.
3	FOV	RO	0	FIFO Overrun When set, indicates that an overrun was encountered on the receive FIFO.

Bit/Field	Name	Type	Reset	Description
2	TXEMP	RO	0	Transmit FIFO Empty When set, indicates that the packet was transmitted and that the TX FIFO is empty.
1	TXER	RO	0	Transmit Error When set, indicates that an error was encountered on the transmitter. The possible errors that can cause this interrupt bit to be set are: ■ The data length field stored in the TX FIFO exceeds 2032 decimal (buffer length - 16 bytes of header data). The frame is not sent when this error occurs. ■ The retransmission attempts during the backoff process have exceeded the maximum limit of 16 decimal.
0	RXINT	RO	0	Packet Received When set, indicates that at least one packet has been received and is stored in the receiver FIFO.

Writes

Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK)

Base 0x4004.8000
Offset 0x000
Type WO, 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	PHYINT	W1C	0	Clear PHY Interrupt Setting this bit clears the PHYINT interrupt in the **MACRIS** register.
5	MDINT	W1C	0	Clear MII Transaction Complete Setting this bit clears the MDINT interrupt in the **MACRIS** register.
4	RXER	W1C	0	Clear Receive Error Setting this bit clears the RXER interrupt in the **MACRIS** register.
3	FOV	W1C	0	Clear FIFO Overrun Setting this bit clears the FOV interrupt in the **MACRIS** register.

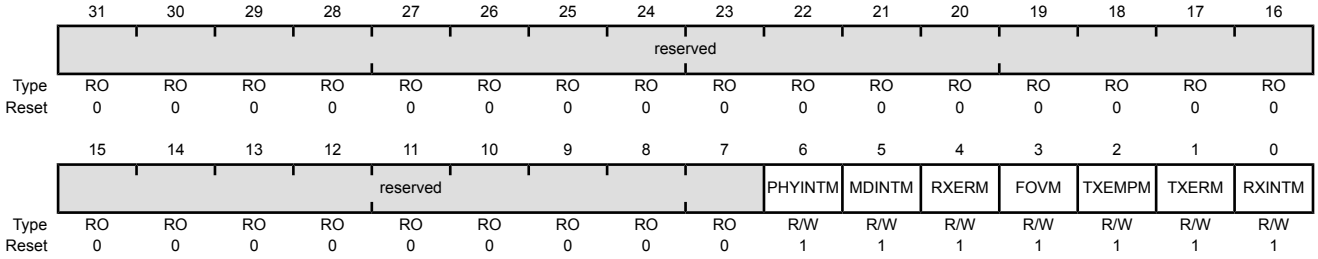
Bit/Field	Name	Type	Reset	Description
2	TXEMP	W1C	0	Clear Transmit FIFO Empty Setting this bit clears the TXEMP interrupt in the **MACRIS** register.
1	TXER	W1C	0	Clear Transmit Error Setting this bit clears the TXER interrupt in the **MACRIS** register and resets the TX FIFO write pointer.
0	RXINT	W1C	0	Clear Packet Received Setting this bit clears the RXINT interrupt in the **MACRIS** register.

Register 2: Ethernet MAC Interrupt Mask (MACIM), offset 0x004

This register allows software to enable/disable Ethernet MAC interrupts. Clearing a bit disables the interrupt, while setting the bit enables it.

Ethernet MAC Interrupt Mask (MACIM)

Base 0x4004.8000
Offset 0x004
Type R/W, reset 0x0000.007F



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	PHYINTM	R/W	1	Mask PHY Interrupt Clearing this bit masks the PHYINT bit in the **MACRIS** register from being set.
5	MDINTM	R/W	1	Mask MII Transaction Complete Clearing this bit masks the MDINT bit in the **MACRIS** register from being set.
4	RXERM	R/W	1	Mask Receive Error Clearing this bit masks the RXER bit in the **MACRIS** register from being set.
3	FOVM	R/W	1	Mask FIFO Overrun Clearing this bit masks the FOV bit in the **MACRIS** register from being set.
2	TXEMPM	R/W	1	Mask Transmit FIFO Empty Clearing this bit masks the TXEMP bit in the **MACRIS** register from being set.
1	TXERM	R/W	1	Mask Transmit Error Clearing this bit masks the TXER bit in the **MACRIS** register from being set.
0	RXINTM	R/W	1	Mask Packet Received Clearing this bit masks the RXINT bit in the **MACRIS** register from being set.

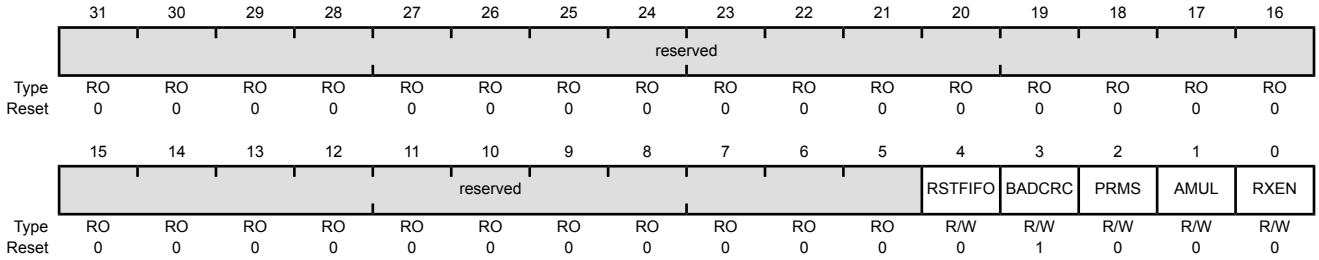
Register 3: Ethernet MAC Receive Control (MACRCTL), offset 0x008

This register configures the receiver and controls the types of frames that are received.

It is important to note that when the receiver is enabled, all valid frames with a broadcast address of FF-FF-FF-FF-FF-FF in the Destination Address field are received and stored in the RX FIFO, even if the AMUL bit is not set.

Ethernet MAC Receive Control (MACRCTL)

Base 0x4004.8000
Offset 0x008
Type R/W, reset 0x0000.0008



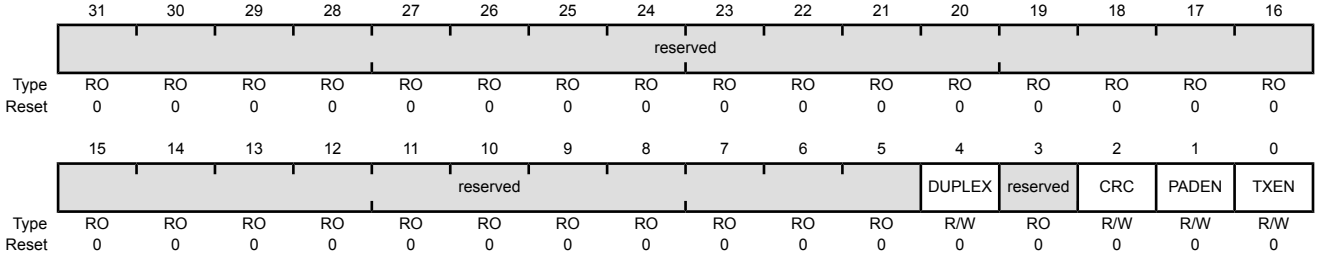
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	RSTFIFO	R/W	0	Clear Receive FIFO When set, this bit clears the receive FIFO. This should be done when software initialization is performed. It is recommended that the receiver be disabled (RXEN = 0), before a reset is initiated (RSTFIFO = 1). This sequence flushes and resets the RX FIFO. This bit is automatically cleared when read.
3	BADCRC	R/W	1	Enable Reject Bad CRC When set, the BADCRC bit enables the rejection of frames with an incorrectly calculated CRC. If a bad CRC is encountered, the RXER bit in the **MACRIS** register is set and the receiver FIFO is reset.
2	PRMS	R/W	0	Enable Promiscuous Mode When set, the PRMS bit enables Promiscuous mode, which accepts all valid frames, regardless of the specified Destination Address.
1	AMUL	R/W	0	Enable Multicast Frames When set, the AMUL bit enables the reception of multicast frames.
0	RXEN	R/W	0	Enable Receiver When set the RXEN bit enables the Ethernet receiver. When this bit is clear, the receiver is disabled and all frames are ignored.

Register 4: Ethernet MAC Transmit Control (MACTCTL), offset 0x00C

This register configures the transmitter and controls the frames that are transmitted.

Ethernet MAC Transmit Control (MACTCTL)

Base 0x4004.8000
Offset 0x00C
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	DUPLEX	R/W	0	Enable Duplex Mode When set, this bit enables Duplex mode, allowing simultaneous transmission and reception.
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	CRC	R/W	0	Enable CRC Generation When set this bit enables the automatic generation of the CRC and its placement at the end of the packet. If this bit is clear, the frames placed in the TX FIFO are sent exactly as they are written into the FIFO. Note that this bit should generally be set.
1	PADEN	R/W	0	Enable Packet Padding When set, this bit enables the automatic padding of packets that do not meet the minimum frame size. Note that this bit should generally be set.
0	TXEN	R/W	0	Enable Transmitter When set, this bit enables the transmitter. When this bit is clear, the transmitter is disabled.

Register 5: Ethernet MAC Data (MACDATA), offset 0x010

Important: This register is read-sensitive. See the register description for details.

This register enables software to access the TX and RX FIFOs.

Reads from this register return the data stored in the RX FIFO from the location indicated by the read pointer. The read pointer is then auto incremented to the next RX FIFO location. Reading from the RX FIFO when a frame has not been received or is in the process of being received will return indeterminate data and not increment the read pointer.

Writes to this register store the data in the TX FIFO at the location indicated by the write pointer. The write pointer is the auto incremented to the next TX FIFO location. Writing more data into the TX FIFO than indicated in the length field will result in the data being lost. Writing less data into the TX FIFO than indicated in the length field will result in indeterminate data being appended to the end of the frame to achieve the indicated length. Attempting to write the next frame into the TX FIFO before transmission of the first has completed will result in the data being lost.

There is no mechanism for randomly accessing bytes in either the RX or TX FIFOs. Data must be read from the RX FIFO sequentially and stored in a buffer for further processing. Once a read has been performed, the data in the FIFO cannot be re-read. Data must be written to the TX FIFO sequentially. If an error is made in placing the frame into the TX FIFO, the write pointer can be reset to the start of the TX FIFO by writing the TXER bit of the **MACIACK** register and then the data re-written.

Reads

Ethernet MAC Data (MACDATA)

Base 0x4004.8000
Offset 0x010
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RXDATA															
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

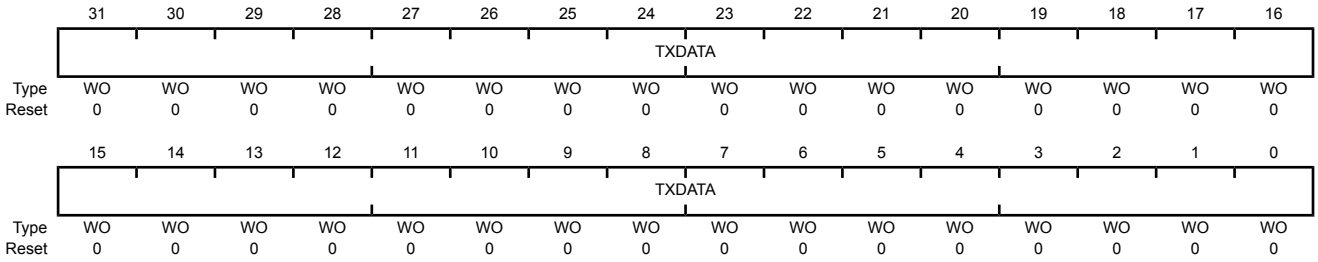
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RXDATA															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	RXDATA	RO	0x0000.0000	Receive FIFO Data The RXDATA bits represent the next word of data stored in the RX FIFO.

Writes

Ethernet MAC Data (MACDATA)

Base 0x4004.8000
Offset 0x010
Type WO, reset 0x0000.0000



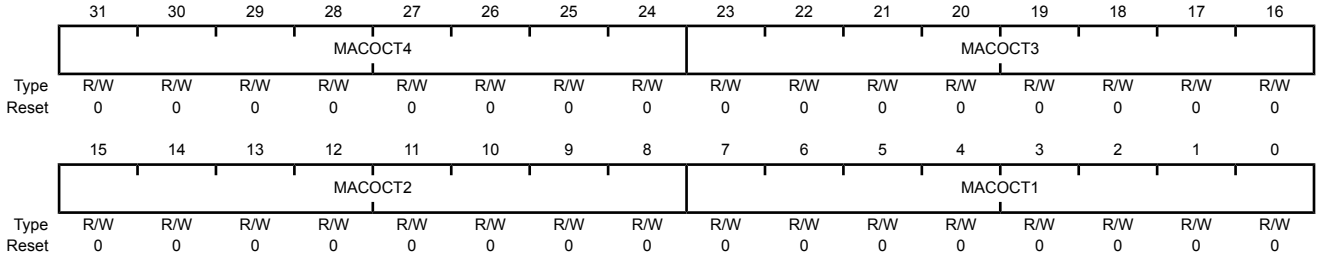
Bit/Field	Name	Type	Reset	Description
31:0	TXDATA	WO	0x0000.0000	Transmit FIFO Data The TXDATA bits represent the next word of data to place in the TX FIFO for transmission.

Register 6: Ethernet MAC Individual Address 0 (MACIA0), offset 0x014

This register enables software to program the first four bytes of the hardware MAC address of the Network Interface Card (NIC). (The last two bytes are in **MACIA1**). The 6-byte Individual Address is compared against the incoming Destination Address fields to determine whether the frame should be received.

Ethernet MAC Individual Address 0 (MACIA0)

Base 0x4004.8000
Offset 0x014
Type R/W, reset 0x0000.0000



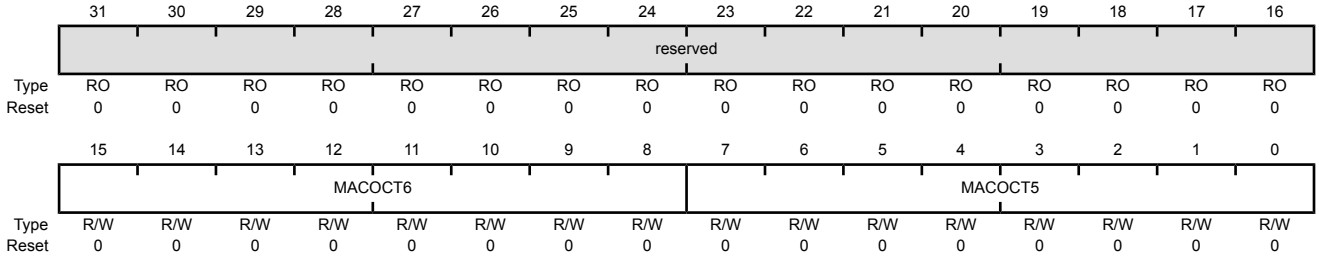
Bit/Field	Name	Type	Reset	Description
31:24	MACOCT4	R/W	0x00	MAC Address Octet 4 The MACOCT4 bits represent the fourth octet of the MAC address used to uniquely identify the Ethernet Controller.
23:16	MACOCT3	R/W	0x00	MAC Address Octet 3 The MACOCT3 bits represent the third octet of the MAC address used to uniquely identify the Ethernet Controller.
15:8	MACOCT2	R/W	0x00	MAC Address Octet 2 The MACOCT2 bits represent the second octet of the MAC address used to uniquely identify the Ethernet Controller.
7:0	MACOCT1	R/W	0x00	MAC Address Octet 1 The MACOCT1 bits represent the first octet of the MAC address used to uniquely identify the Ethernet Controller.

Register 7: Ethernet MAC Individual Address 1 (MACIA1), offset 0x018

This register enables software to program the last two bytes of the hardware MAC address of the Network Interface Card (NIC). (The first four bytes are in **MACIA0**). The 6-byte IAR is compared against the incoming Destination Address fields to determine whether the frame should be received.

Ethernet MAC Individual Address 1 (MACIA1)

Base 0x4004.8000
Offset 0x018
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	MACOCT6	R/W	0x00	MAC Address Octet 6 The MACOCT6 bits represent the sixth octet of the MAC address used to uniquely identify each Ethernet Controller.
7:0	MACOCT5	R/W	0x00	MAC Address Octet 5 The MACOCT5 bits represent the fifth octet of the MAC address used to uniquely identify the Ethernet Controller.

Register 8: Ethernet MAC Threshold (MACTHR), offset 0x01C

In order to increase the transmission rate, it is possible to program the Ethernet Controller to begin transmission of the next frame prior to the completion of the transmission of the current frame. Note: Extreme care must be used when implementing this function. Software must be able to guarantee that the complete frame is able to be stored in the transmission FIFO prior to the completion of the transmission frame.

This register enables software to set the threshold level at which the transmission of the frame begins. If the THRESH bits are set to 0x3F, which is the reset value, the early transmission feature is disabled, and transmission does not start until the NEWTX bit is set in the **MACTR** register.

Writing the THRESH bits to any value besides 0x3F enables the early transmission feature. Once the byte count of data in the TX FIFO reaches the value derived from the THRESH bits as shown below, transmission of the frame begins. When THRESH is set to all 0s, transmission of the frame begins after 4 bytes (a single write) are stored in the TX FIFO. Each increment of the THRESH bit field waits for an additional 32 bytes of data (eight writes) to be stored in the TX FIFO. Therefore, a value of 0x01 causes the transmitter to wait for 36 bytes of data to be written while a value of 0x02 makes the wait equal to 68 bytes of written data. In general, early transmission starts when:

```
Number of Bytes >= 4 (THRESH x 8 + 1)
```

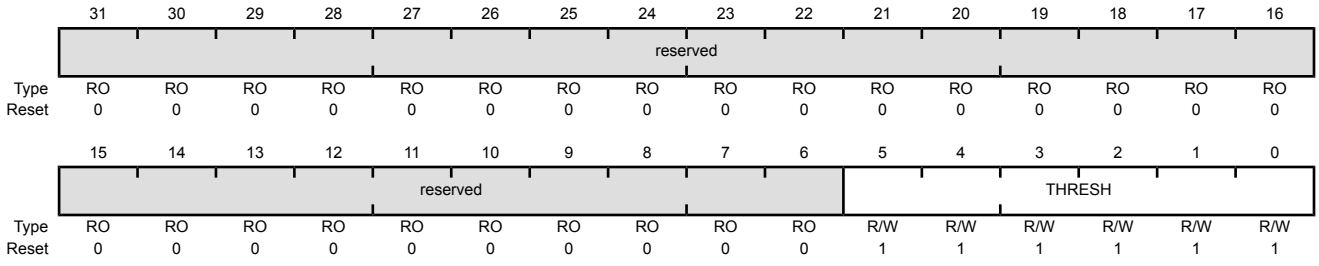
Reaching the threshold level has the same effect as setting the NEWTX bit in the **MACTR** register. Transmission of the frame begins and then the number of bytes indicated by the Data Length field is transmitted. Because under-run checking is not performed, if any event, such as an interrupt, delays the filling of the FIFO, the tail pointer may reach and pass the write pointer in the TX FIFO. In this event, indeterminate values are transmitted rather than the end of the frame. Therefore, sufficient bus bandwidth for writing to the TX FIFO must be guaranteed by the software.

If a frame smaller than the threshold level must be sent, the NEWTX bit in the **MACTR** register must be set with an explicit write. This initiates the transmission of the frame even though the threshold limit has not been reached.

If the threshold level is set too small, it is possible for the transmitter to underrun. If this occurs, the transmit frame is aborted, and a transmit error occurs. Note that in this case, the TXER bit in the **MACRIS** is not set meaning that the CPU receives no indication that a transmit error happened.

Ethernet MAC Threshold (MACTHR)

Base 0x4004.8000
Offset 0x01C
Type R/W, reset 0x0000.003F



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.

Bit/Field	Name	Type	Reset	Description
5:0	THRESH	R/W	0x3F	Threshold Value The THRESH bits represent the early transmit threshold. Once the amount of data in the TX FIFO exceeds the value represented by the above equation, transmission of the packet begins.

Register 9: Ethernet MAC Management Control (MACMCTL), offset 0x020

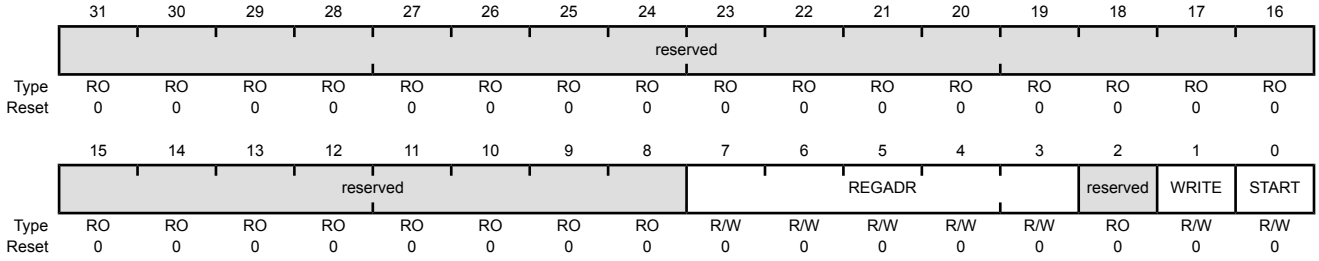
This register enables software to control the transfer of data to and from the MII Management registers in the Ethernet PHY layer. The address, name, type, reset configuration, and functional description of each of these registers can be found in Table 16-4 on page 605 and in "MII Management Register Descriptions" on page 625.

In order to initiate a *read* transaction from the MII Management registers, the WRITE bit must be cleared during the same cycle that the START bit is set.

In order to initiate a *write* transaction to the MII Management registers, the WRITE bit must be set during the same cycle that the START bit is set.

Ethernet MAC Management Control (MACMCTL)

Base 0x4004.8000
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.
7:3	REGADR	R/W	0x0	MII Register Address The REGADR bit field represents the MII Management register address for the next MII management interface transaction. Refer to Table 16-4 on page 605 for the PHY register offsets. Note that any address that is not valid in the register map should not be written to and any data read should be ignored.
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	WRITE	R/W	0	MII Register Transaction Type The WRITE bit represents the operation of the next MII management interface transaction. If WRITE is set, the next operation is a write; if WRITE is clear, the next transaction is a read.
0	START	R/W	0	MII Register Transaction Enable The START bit represents the initiation of the next MII management interface transaction. When this bit is set, the MII register located at REGADR is read (WRITE=0) or written (WRITE=1).

Register 10: Ethernet MAC Management Divider (MACMDV), offset 0x024

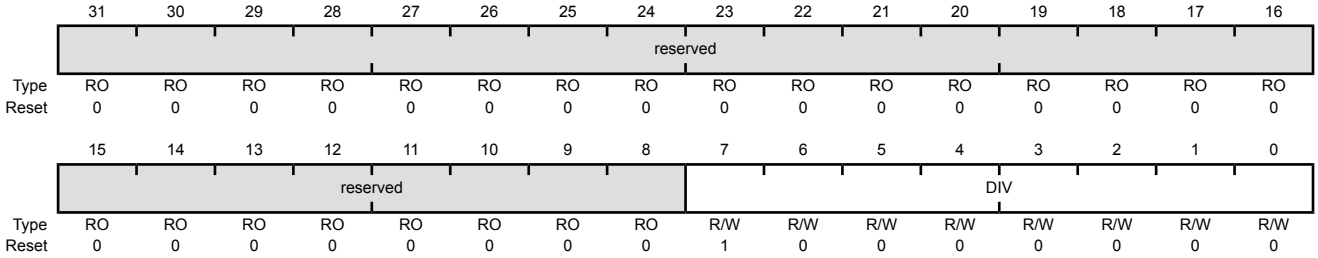
This register enables software to set the clock divider for the Management Data Clock (MDC). This clock is used to synchronize read and write transactions between the system and the MII Management registers. The frequency of the MDC clock can be calculated from the following formula:

$$F_{mdc} = \frac{F_{ipclk}}{2 \times (MACDVR + 1)}$$

The clock divider must be written with a value that ensures that the MDC clock does not exceed a frequency of 2.5 MHz.

Ethernet MAC Management Divider (MACMDV)

Base 0x4004.8000
Offset 0x024
Type R/W, reset 0x0000.0080



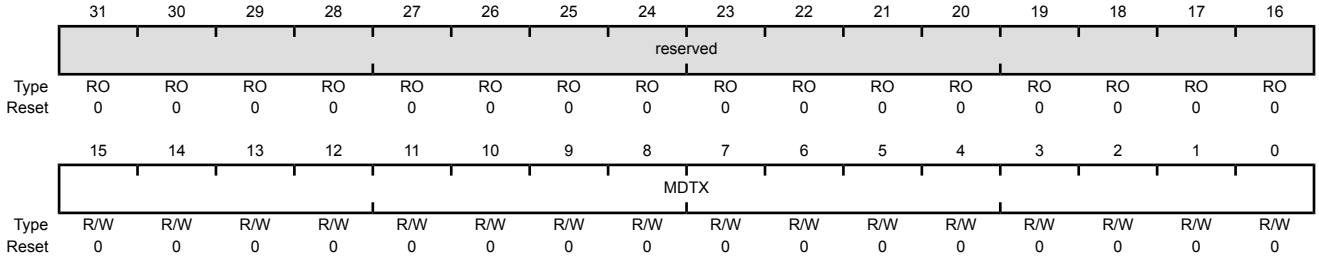
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIV	R/W	0x80	Clock Divider The DIV bits are used to set the clock divider for the MDC clock used to transmit data between the MAC and PHY layers over the serial MII interface.

Register 11: Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C

This register holds the next value to be written to the MII Management registers.

Ethernet MAC Management Transmit Data (MACMTXD)

Base 0x4004.8000
Offset 0x02C
Type R/W, reset 0x0000.0000



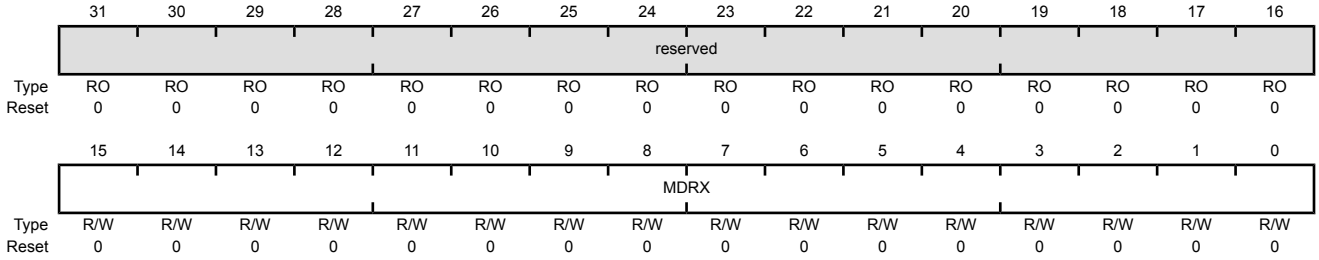
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MDTX	R/W	0x0000	MII Register Transmit Data The MDTX bits represent the data that will be written in the next MII management transaction.

Register 12: Ethernet MAC Management Receive Data (MACMRXD), offset 0x030

This register holds the last value read from the MII Management registers.

Ethernet MAC Management Receive Data (MACMRXD)

Base 0x4004.8000
Offset 0x030
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	MDRX	R/W	0x0000	MII Register Receive Data The MDRX bits represent the data that was read in the previous MII management transaction.

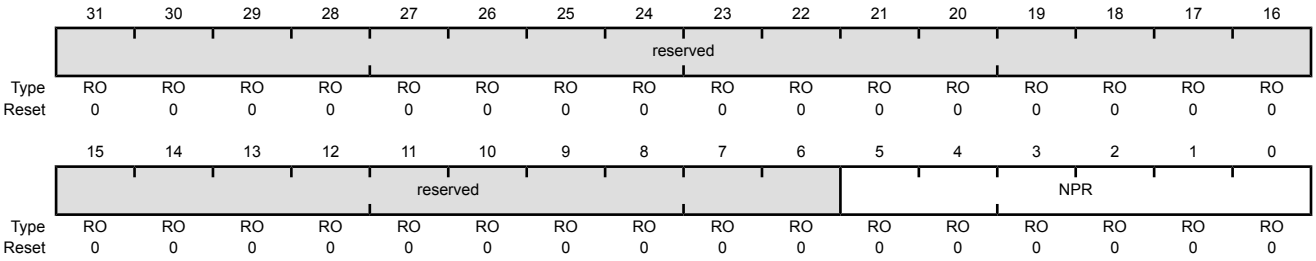
Register 13: Ethernet MAC Number of Packets (MACNP), offset 0x034

This register holds the number of frames that are currently in the RX FIFO. When NPR is 0, there are no frames in the RX FIFO, and the RXINT bit is clear. When NPR is any other value, at least one frame is in the RX FIFO, and the RXINT bit in the **MACRIS** register is set.

Note: The FCS bytes are not included in the NPR value. As a result, the NPR value could be zero before the FCS bytes are read from the FIFO. In addition, a new packet could be received before the NPR value reaches zero. To ensure that the entire packet is received, either use the DriverLib EthernetPacketGet() API or compare the number of bytes received to the Length field from the frame to determine when the packet has been completely read.

Ethernet MAC Number of Packets (MACNP)

Base 0x4004.8000
Offset 0x034
Type RO, reset 0x0000.0000



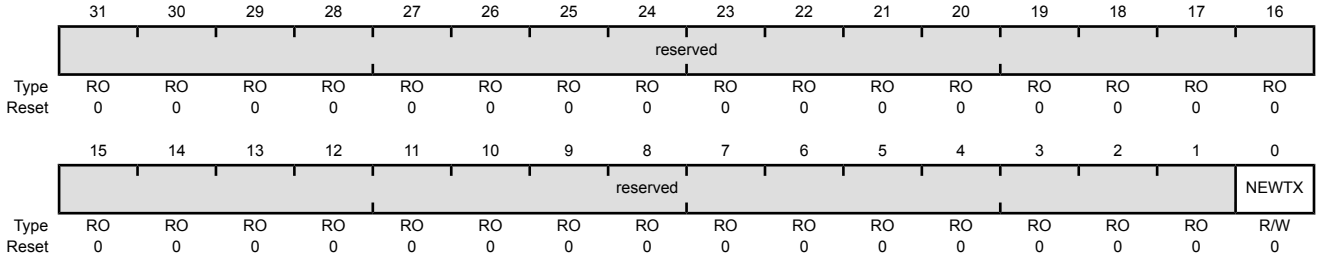
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	NPR	RO	0x00	Number of Packets in Receive FIFO The NPR bits represent the number of packets stored in the RX FIFO. While the NPR field is greater than 0, the RXINT interrupt in the **MACRIS** register is set.

Register 14: Ethernet MAC Transmission Request (MACTR), offset 0x038

This register enables software to initiate the transmission of the frame currently located in the TX FIFO. Once the frame has been transmitted from the TX FIFO or a transmission error has been encountered, the NEWTX bit is automatically cleared.

Ethernet MAC Transmission Request (MACTR)

Base 0x4004.8000
Offset 0x038
Type R/W, reset 0x0000.0000



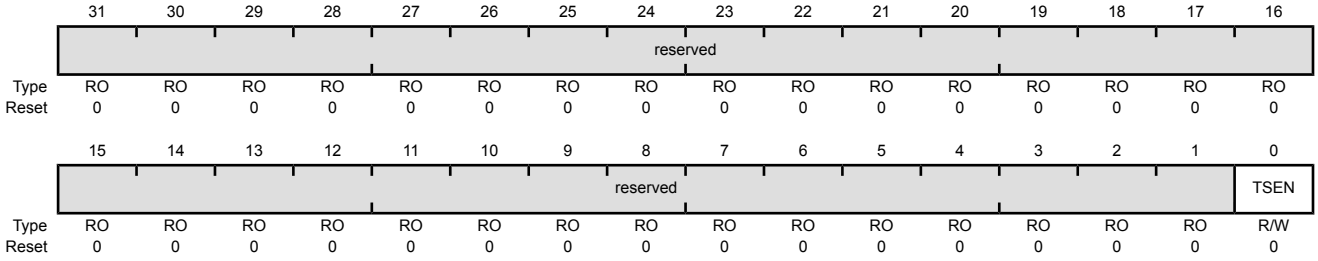
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	NEWTX	R/W	0	New Transmission When set, the NEWTX bit initiates an Ethernet transmission once the packet has been placed in the TX FIFO. This bit is cleared once the transmission has been completed. If early transmission is being used (see the **MACTHR** register), this bit does not need to be set.

Register 15: Ethernet MAC Timer Support (MACTS), offset 0x03C

This register enables software to enable highly precise timing on the transmission and reception of frames. To enable this function, set the TSEN bit.

Ethernet MAC Timer Support (MACTS)

Base 0x4004.8000
Offset 0x03C
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	TSEN	R/W	0	Time Stamp Enable When set, the TSEN bit multiplexes the TX and RX interrupts to the CCP inputs of General-Purpose Timer 3.

16.7 MII Management Register Descriptions

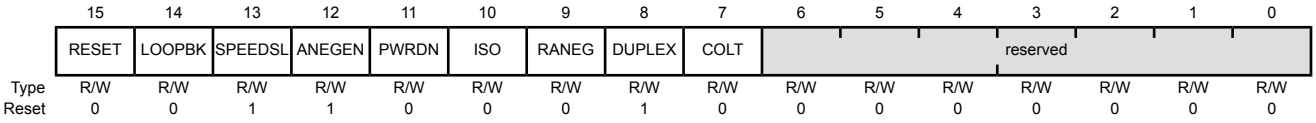
The *IEEE 802.3 standard* specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers. All addresses given are absolute. Addresses not listed are reserved; these addresses should not be written to and any data read should be ignored. Also see “Ethernet MAC Register Descriptions” on page 606.

Register 16: Ethernet PHY Management Register 0 – Control (MR0), address 0x00

This register enables software to configure the operation of the PHY layer. The default settings of these registers are designed to initialize the Ethernet Controller to a normal operational mode without configuration.

Ethernet PHY Management Register 0 – Control (MR0)

Base 0x4004.8000
Address 0x00
Type R/W, reset 0x3100



Bit/Field	Name	Type	Reset	Description
15	RESET	R/W	0	Reset Registers When set, this bit resets the PHY layer registers to their default state and reinitializes internal state machines. Once the reset operation has completed, this bit is cleared by hardware.
14	LOOPBK	R/W	0	Loopback Mode When set, this bit enables the Loopback mode of operation. The receiver ignores external inputs and receives the data that is transmitted by the transmitter.
13	SPEEDSL	R/W	1	Speed Select Value Description 1 Enables the 100 Mb/s mode of operation (100BASE-TX). 0 Enables the 10 Mb/s mode of operation (10BASE-T).
12	ANEGEN	R/W	1	Auto-Negotiation Enable When set, this bit enables the auto-negotiation process.
11	PWRDN	R/W	0	Power Down When set, this bit places the PHY layer into a low-power consuming state. All data on the data inputs is ignored.
10	ISO	R/W	0	Isolate When set, this bit isolates the transmit and receive data paths and ignores all data being transmitted and received.
9	RANEG	R/W	0	Restart Auto-Negotiation When set, this bit restarts the auto-negotiation process. Once the restart has initiated, this bit is cleared by hardware.

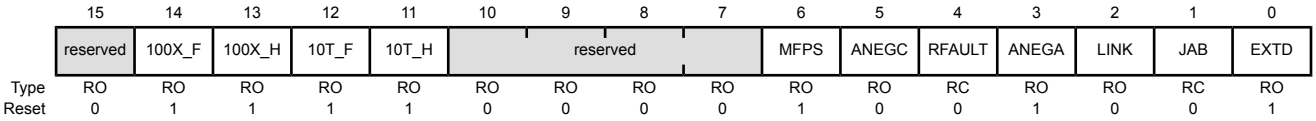
Bit/Field	Name	Type	Reset	Description
8	DUPLEX	R/W	1	Set Duplex Mode Value Description 1 Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the auto-negotiation process. 0 Enables the Half-Duplex mode of operation.
7	COLT	R/W	0	Collision Test When set, this bit enables the Collision Test mode of operation. The COLT bit is set after the initiation of a transmission and is cleared once the transmission is halted.
6:0	reserved	R/W	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. These bits should always be written as zero.

Register 17: Ethernet PHY Management Register 1 – Status (MR1), address 0x01

This register enables software to determine the capabilities of the PHY layer and perform its initialization and operation appropriately.

Ethernet PHY Management Register 1 – Status (MR1)

Base 0x4004.8000
Address 0x01
Type RO, reset 0x7849



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	100X_F	RO	1	100BASE-TX Full-Duplex Mode When set, this bit indicates that the Ethernet Controller is capable of supporting 100BASE-TX Full-Duplex mode.
13	100X_H	RO	1	100BASE-TX Half-Duplex Mode When set, this bit indicates that the Ethernet Controller is capable of supporting 100BASE-TX Half-Duplex mode.
12	10T_F	RO	1	10BASE-T Full-Duplex Mode When set, this bit indicates that the Ethernet Controller is capable of 10BASE-T Full-Duplex mode.
11	10T_H	RO	1	10BASE-T Half-Duplex Mode When set, this bit indicates that the Ethernet Controller is capable of supporting 10BASE-T Half-Duplex mode.
10:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	MFPS	RO	1	Management Frames with Preamble Suppressed When set, this bit indicates that the Management Interface is capable of receiving management frames with the preamble suppressed.
5	ANEGC	RO	0	Auto-Negotiation Complete When set, this bit indicates that the auto-negotiation process has been completed and that the extended registers defined by the auto-negotiation protocol are valid.
4	RFAULT	RC	0	Remote Fault When set, this bit indicates that a remote fault condition has been detected. This bit remains set until it is read, even if the condition no longer exists.
3	ANEGA	RO	1	Auto-Negotiation When set, this bit indicates that the Ethernet Controller has the ability to perform auto-negotiation.

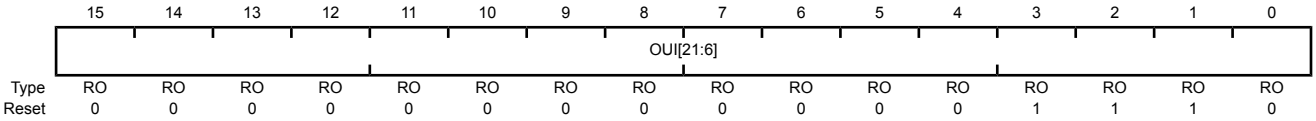
Bit/Field	Name	Type	Reset	Description
2	LINK	RO	0	Link Made When set, this bit indicates that a valid link has been established by the Ethernet Controller.
1	JAB	RC	0	Jabber Condition When set, this bit indicates that a jabber condition has been detected by the Ethernet Controller. This bit remains set until it is read, even if the jabber condition no longer exists.
0	EXTD	RO	1	Extended Capabilities When set, this bit indicates that the Ethernet Controller provides an extended set of capabilities that can be accessed through the extended register set.

Register 18: Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02

This register, along with **MR3**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2)

Base 0x4004.8000
Address 0x02
Type RO, reset 0x000E



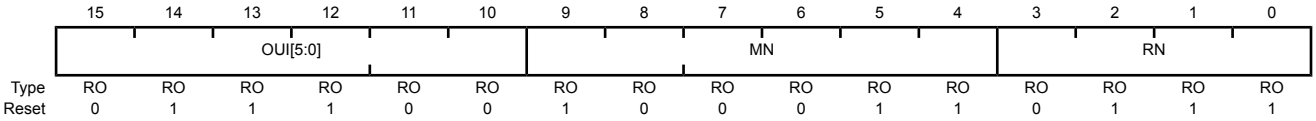
Bit/Field	Name	Type	Reset	Description
15:0	OUI[21:6]	RO	0x000E	Organizationally Unique Identifier[21:6] This field, along with the `OUI[5:0]` field in **MR3**, makes up the Organizationally Unique Identifier indicating the PHY manufacturer.

Register 19: Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03

This register, along with **MR2**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3)

Base 0x4004.8000
Address 0x03
Type RO, reset 0x7237



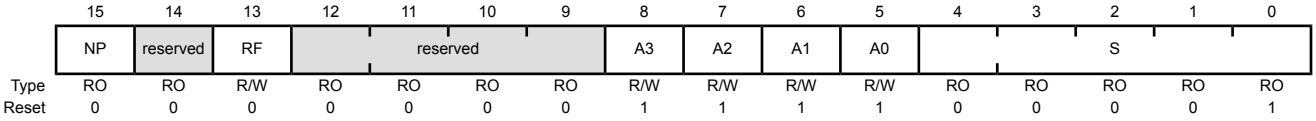
Bit/Field	Name	Type	Reset	Description
15:10	OUI[5:0]	RO	0x1C	Organizationally Unique Identifier[5:0] This field, along with the `OUI[21:6]` field in **MR2**, makes up the Organizationally Unique Identifier indicating the PHY manufacturer.
9:4	MN	RO	0x23	Model Number The `MN` field represents the Model Number of the PHY.
3:0	RN	RO	0x7	Revision Number The `RN` field represents the Revision Number of the PHY implementation.

Register 20: Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04

This register provides the advertised abilities of the Ethernet Controller used during auto-negotiation. Bits 8:5 represent the Technology Ability Field bits. This field can be overwritten by software to auto-negotiate to an alternate common technology. Writing to this register has no effect until auto-negotiation is re-initiated by setting the RANEG bit in the **MR0** register.

Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4)

Base 0x4004.8000
Address 0x04
Type R/W, reset 0x01E1



Bit/Field	Name	Type	Reset	Description
15	NP	RO	0	Next Page When set, this bit indicates the Ethernet Controller is capable of Next Page exchanges to provide more detailed information on the PHY layer's capabilities.
14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	RF	R/W	0	Remote Fault When set, this bit indicates to the link partner that a Remote Fault condition has been encountered.
12:9	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	A3	R/W	1	Technology Ability Field[3] When set, this bit indicates that the Ethernet Controller supports the 100Base-TX full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register.
7	A2	R/W	1	Technology Ability Field[2] When set, this bit indicates that the Ethernet Controller supports the 100Base-TX half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register.
6	A1	R/W	1	Technology Ability Field[1] When set, this bit indicates that the Ethernet Controller supports the 10BASE-T full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register..

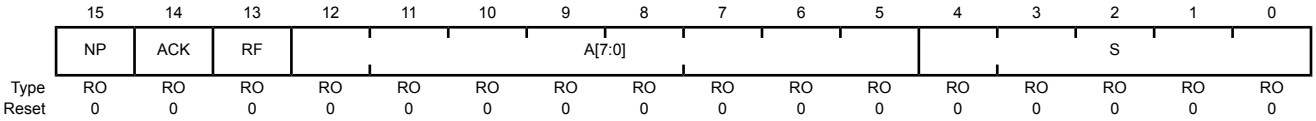
Bit/Field	Name	Type	Reset	Description
5	A0	R/W	1	Technology Ability Field[0] When set, this bit indicates that the Ethernet Controller supports the 10BASE-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the **MR0** register..
4:0	S	RO	0x1	Selector Field The S field encodes 32 possible messages for communicating between Ethernet Controllers. This field is hard-coded to 0x01, indicating that the Stellaris Ethernet Controller is *IEEE 802.3* compliant.

Register 21: Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05

This register provides the advertised abilities of the link partner's Ethernet Controller that are received and stored during auto-negotiation.

Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5)

Base 0x4004.8000
Address 0x05
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15	NP	RO	0	Next Page When set, this bit indicates that the link partner's Ethernet Controller is capable of Next page exchanges to provide more detailed information on the Ethernet Controller's capabilities.
14	ACK	RO	0	Acknowledge When set, this bit indicates that the Ethernet Controller has successfully received the link partner's advertised abilities during auto-negotiation.
13	RF	RO	0	Remote Fault Used as a standard transport mechanism for transmitting simple fault information from the link partner.
12:5	A[7:0]	RO	0x00	Technology Ability Field The `A[7:0]` field encodes individual technologies that are supported by the Ethernet Controller. See the **MR4** register for definitions. Note that bits 12:9 describe functions that are not implemented on the Stellaris Ethernet Controller. Refer to the IEEE 802.3 standard for definitions.
4:0	S	RO	0x00	Selector Field The S field encodes possible messages for communicating between Ethernet Controllers.

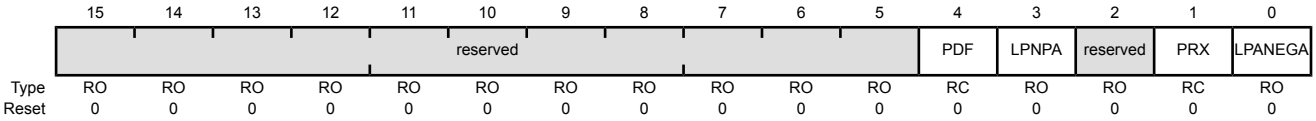
Value	Description
0x00	Reserved
0x01	IEEE Std 802.3
0x02	IEEE Std 802.9 ISLAN-16T
0x03	IEEE Std 802.5
0x04	IEEE Std 1394
0x05–0x1F	Reserved

Register 22: Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06

This register enables software to determine the auto-negotiation and next page capabilities of the Ethernet Controller and the link partner after auto-negotiation.

Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6)

Base 0x4004.8000
Address 0x06
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:5	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	PDF	RC	0	Parallel Detection Fault When set, this bit indicates that more than one technology has been detected at link up. This bit is cleared when read.
3	LPNPA	RO	0	Link Partner is Next Page Able When set, this bit indicates that the link partner is enabled to support next page.
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	PRX	RC	0	New Page Received When set, this bit indicates that a new page has been received from the link partner and stored. This bit remains set until the register is read.
0	LPANEGA	RO	0	Link Partner is Auto-Negotiation Able When set, this bit indicates that the link partner is enabled to support auto-negotiation.

Register 23: Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10

This register enables software to configure the operation of vendor-specific modes of the Ethernet Controller.

Ethernet PHY Management Register 16 – Vendor-Specific (MR16)

Base 0x4004.8000
Address 0x10
Type R/W, reset 0x0140

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPTR	INPOL	reserved	TXHIM	SQEI	NL10	reserved				APOL	RVSPOL	reserved		PCSBP	RXCC
Type R/W	R/W0	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset 0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15	RPTR	R/W	0	Repeater Mode When set, this bit enables the repeater mode of operation. In this mode, full-duplex is not allowed and the Carrier Sense signal only responds to receive activity.
14	INPOL	R/W0	0	Interrupt Polarity Value Description 1 Sets the polarity of the PHY interrupt to be active High. 0 Sets the polarity of the PHY interrupt to active Low. **Important:** Because the Media Access Controller expects active Low interrupts from the PHY, this bit must always be written with a 0 to ensure proper operation.
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	TXHIM	R/W	0	Transmit High Impedance Mode When set, this bit enables the transmitter High Impedance mode. In this mode, the TXOP and TXON transmitter pins are put into a high impedance state. The RXIP and RXIN pins remain fully functional.
11	SQEI	R/W	0	SQE Inhibit Testing When set, this bit prohibits 10BASE-T SQE testing. When clear, the SQE testing is performed by generating a collision pulse following the completion of the transmission of a frame.
10	NL10	R/W	0	Natural Loopback Mode When set, this bit enables the 10BASE-T Natural Loopback mode. In this mode, the transmission data received by the Ethernet Controller is looped back onto the receive data path when 10BASE-T mode is enabled.
9:6	reserved	RO	0x5	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
5	APOL	R/W	0	Auto-Polarity Disable When set, this bit disables the Ethernet Controller's auto-polarity function. If this bit is clear, the Ethernet Controller automatically inverts the received signal due to a wrong polarity connection during auto-negotiation when in 10BASE-T mode.
4	RVSPOL	R/W	0	Receive Data Polarity This bit indicates whether the receive data pulses are being inverted. If the APOL bit is 0, then the RVSPOL bit is read-only and indicates whether the auto-polarity circuitry is reversing the polarity. In this case, if RVSPOL is set, it indicates that the receive data is inverted; if RVSPOL is clear, it indicates that the receive data is not inverted. If the APOL bit is 1, then the RVSPOL bit is writable and software can force the receive data to be inverted. Setting RVSPOL to 1 forces the receive data to be inverted; clearing RVSPOL does not invert the receive data.
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	PCSBP	R/W	0	PCS Bypass When set, this bit enables the bypass of the PCS and scrambling/descrambling functions in 100BASE-TX mode. This mode is only valid when auto-negotiation is disabled and 100BASE-TX mode is enabled.
0	RXCC	R/W	0	Receive Clock Control When set, this bit enables the Receive Clock Control power saving mode if the Ethernet Controller is configured in 100BASE-TX mode. This mode shuts down the receive clock when no data is being received to save power. This mode should not be used when PCSBP is enabled and is automatically disabled when the LOOPBK bit in the **MR0** register is set.

Register 24: Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17), address 0x11

This register provides the means for controlling and observing the events which trigger a PHY layer interrupt in the **MACRIS** register. This register can also be used in a polling mode via the Media Independent Interface as a means to observe key events within the PHY layer via one register address. Bits 0 through 7 are status bits which are each set based on an event. These bits are cleared after the register is read. Bits 8 through 15 of this register, when set, enable the corresponding bit in the lower byte to signal a PHY layer interrupt in the **MACRIS** register.

Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17)

Base 0x4004.8000
Address 0x11
Type R/W, reset 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JABBER_IE	RXER_IE	PRX_IE	PDF_IE	LPACK_IE	LSCHG_IE	RFAULT_IE	ANEGCOMP_IE	JABBER_INT	RXER_INT	PRX_INT	PDF_INT	LPACK_INT	LSCHG_INT	RFAULT_INT	ANEGCOMP_INT
Type R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RC	RC	RC	RC	RC	RC	RC	RC
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15	JABBER_IE	R/W	0	Jabber Interrupt Enable When set, this bit enables system interrupts when a Jabber condition is detected by the Ethernet Controller.
14	RXER_IE	R/W	0	Receive Error Interrupt Enable When set, this bit enables system interrupts when a receive error is detected by the Ethernet Controller.
13	PRX_IE	R/W	0	Page Received Interrupt Enable When set, this bit enables system interrupts when a new page is received by the Ethernet Controller.
12	PDF_IE	R/W	0	Parallel Detection Fault Interrupt Enable When set, this bit enables system interrupts when a Parallel Detection Fault is detected by the Ethernet Controller.
11	LPACK_IE	R/W	0	LP Acknowledge Interrupt Enable When set, this bit enables system interrupts when FLP bursts are received with the ACK bit in the **MR5** register during auto-negotiation.
10	LSCHG_IE	R/W	0	Link Status Change Interrupt Enable When set, this bit enables system interrupts when the link status changes from OK to FAIL.
9	RFAULT_IE	R/W	0	Remote Fault Interrupt Enable When set, this bit enables system interrupts when a remote fault condition is signaled by the link partner.
8	ANEGCOMP_IE	R/W	0	Auto-Negotiation Complete Interrupt Enable When set, this bit enables system interrupts when the auto-negotiation sequence has completed successfully.

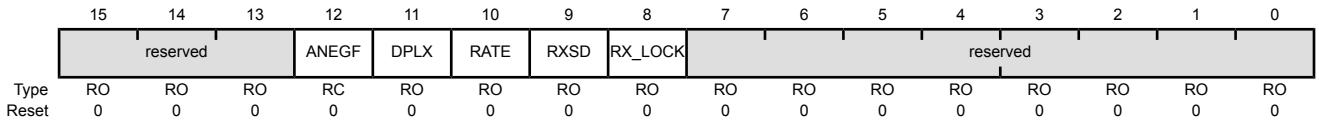
Bit/Field	Name	Type	Reset	Description
7	JABBER_INT	RC	0	Jabber Event Interrupt When set, this bit indicates that a Jabber event has been detected by the 10BASE-T circuitry.
6	RXER_INT	RC	0	Receive Error Interrupt When set, this bit indicates that a receive error has been detected by the Ethernet Controller.
5	PRX_INT	RC	0	Page Receive Interrupt When set, this bit indicates that a new page has been received from the link partner during auto-negotiation.
4	PDF_INT	RC	0	Parallel Detection Fault Interrupt When set, this bit indicates that a parallel detection fault has been detected by the Ethernet Controller during the auto-negotiation process.
3	LPACK_INT	RC	0	LP Acknowledge Interrupt When set, this bit indicates that an FLP burst has been received with the ACK bit set in the **MR5** register during auto-negotiation.
2	LSCHG_INT	RC	0	Link Status Change Interrupt When set, this bit indicates that the link status has changed from OK to FAIL.
1	RFAULT_INT	RC	0	Remote Fault Interrupt When set, this bit indicates that a remote fault condition has been signaled by the link partner.
0	ANEGCOMP_INT	RC	0	Auto-Negotiation Complete Interrupt When set, this bit indicates that the auto-negotiation sequence has completed successfully.

Register 25: Ethernet PHY Management Register 18 – Diagnostic (MR18), address 0x12

This register enables software to diagnose the results of the previous auto-negotiation.

Ethernet PHY Management Register 18 – Diagnostic (MR18)

Base 0x4004.8000
Address 0x12
Type RO, reset 0x0000



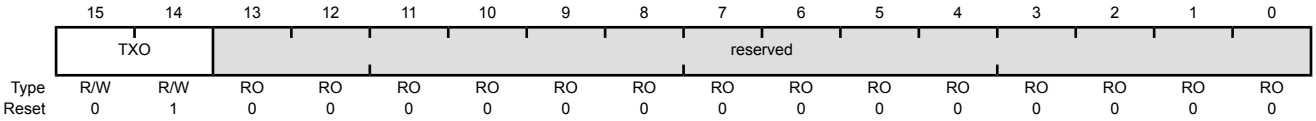
Bit/Field	Name	Type	Reset	Description
15:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	ANEGF	RC	0	Auto-Negotiation Failure When set, this bit indicates that no common technology was found during auto-negotiation and auto-negotiation has failed. This bit remains set until read.
11	DPLX	RO	0	Duplex Mode When set, this bit indicates that Full-Duplex was the highest common denominator found during the auto-negotiation process. Otherwise, Half-Duplex was the highest common denominator found.
10	RATE	RO	0	Rate When set, this bit indicates that 100BASE-TX was the highest common denominator found during the auto-negotiation process. Otherwise, 10BASE-T was the highest common denominator found.
9	RXSD	RO	0	Receive Detection When set, this bit indicates that receive signal detection has occurred (in 100BASE-TX mode) or that Manchester-encoded data has been detected (in 10BASE-T mode).
8	RX_LOCK	RO	0	Receive PLL Lock When set, this bit indicates that the Receive PLL has locked onto the receive signal for the selected speed of operation (10BASE-T or 100BASE-TX).
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 26: Ethernet PHY Management Register 19 – Transceiver Control (MR19), address 0x13

This register enables software to set the gain of the transmit output to compensate for transformer loss.

Ethernet PHY Management Register 19 – Transceiver Control (MR19)

Base 0x4004.8000
Address 0x13
Type R/W, reset 0x4000



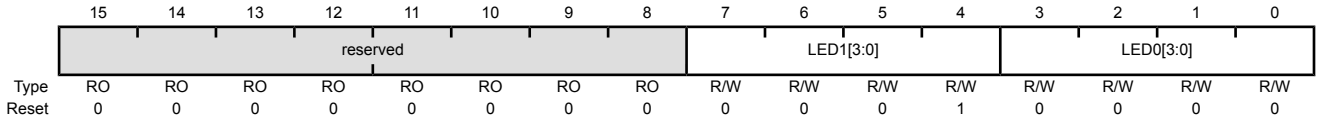
Bit/Field	Name	Type	Reset	Description
15:14	TXO	R/W	0x1	Transmit Amplitude Selection The TX0 field sets the transmit output amplitude to account for transmit transformer insertion loss. Value Description 0x0 Gain set for 0.0dB of insertion loss 0x1 Gain set for 0.4dB of insertion loss 0x2 Gain set for 0.8dB of insertion loss 0x3 Gain set for 1.2dB of insertion loss
13:0	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.

Register 27: Ethernet PHY Management Register 23 – LED Configuration (MR23), address 0x17

This register enables software to select the source that causes the LED1 and LED0 signals to toggle.

Ethernet PHY Management Register 23 – LED Configuration (MR23)

Base 0x4004.8000
Address 0x17
Type R/W, reset 0x0010



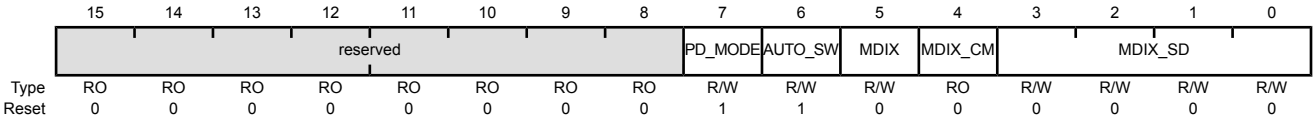
Bit/Field	Name	Type	Reset	Description
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:4	LED1[3:0]	R/W	0x1	LED1 Source The LED1 field selects the source that toggles the LED1 signal. Value Description 0x0 Link OK 0x1 RX or TX Activity (Default LED1) 0x2 Reserved 0x3 Reserved 0x4 Reserved 0x5 100BASE-TX mode 0x6 10BASE-T mode 0x7 Full-Duplex 0x8 Link OK & Blink=RX or TX Activity
3:0	LED0[3:0]	R/W	0x0	LED0 Source The LED0 field selects the source that toggles the LED0 signal. Value Description 0x0 Link OK (Default LED0) 0x1 RX or TX Activity 0x2 Reserved 0x3 Reserved 0x4 Reserved 0x5 100BASE-TX mode 0x6 10BASE-T mode 0x7 Full-Duplex 0x8 Link OK & Blink=RX or TX Activity

Register 28: Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24), address 0x18

This register enables software to control the behavior of the MDI/MDIX mux and its switching capabilities.

Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24)

Base 0x4004.8000
Address 0x18
Type R/W, reset 0x00C0



Bit/Field	Name	Type	Reset	Description
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	PD_MODE	R/W	1	Parallel Detection Mode When set, enables the Parallel Detection mode and allows auto-switching to work when auto-negotiation is not enabled.
6	AUTO_SW	R/W	1	Auto-Switching Enable When set, enables Auto-Switching of the MDI/MDIX mux.
5	MDIX	R/W	0	Auto-Switching Configuration When set, indicates that the MDI/MDIX mux is in the crossover (MDIX) configuration. When 0, it indicates that the mux is in the pass-through (MDI) configuration. When the AUTO_SW bit is 1, the MDIX bit is read-only. When the AUTO_SW bit is 0, the MDIX bit is read/write and can be configured manually.
4	MDIX_CM	RO	0	Auto-Switching Complete When set, indicates that the auto-switching sequence has completed. If 0, it indicates that the sequence has not completed or that auto-switching is disabled.
3:0	MDIX_SD	R/W	0x0	Auto-Switching Seed This field provides the initial seed for the switching algorithm. This seed directly affects the number of attempts [5,4] respectively to write bits [3:0]. A 0 sets the seed to 0x5.

17 Analog Comparators

An analog comparator is a peripheral that compares two analog voltages, and provides a logical output that signals the comparison result.

Note: Not all comparators have the option to drive an output pin.

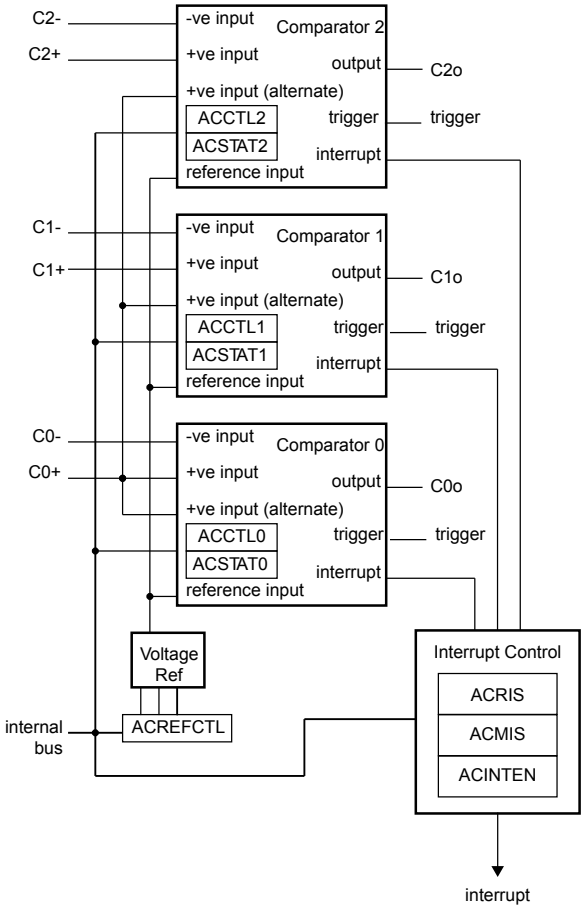
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts or triggers to the ADC to cause it to start capturing a sample sequence. The interrupt generation and ADC triggering logic is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

The Stellaris® Analog Comparators module has the following features:

- Three independent integrated analog comparators
- Configurable for output to drive an output pin, generate an interrupt, or initiate an ADC sample sequence
- Compare external pin input to external pin input or to internal programmable voltage reference
- Compare a test voltage against any one of these voltages
  - An individual external reference voltage
  - A shared single external reference voltage
  - A shared internal reference voltage

17.1 Block Diagram

Figure 17-1. Analog Comparator Module Block Diagram



17.2 Signal Description

Table 17-1 on page 645 and Table 17-2 on page 646 list the external signals of the Analog Comparators and describe the function of each. The Analog Comparator output signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 308) should be set to choose the Analog Comparator function. The positive and negative input signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 286.

Table 17-1. Analog Comparators Signals (100LQFP)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
C0+	90	I	Analog	Analog comparator 0 positive input.
C0-	92	I	Analog	Analog comparator 0 negative input.
C0o	90	O	TTL	Analog comparator 0 output.
C1+	24	I	Analog	Analog comparator 1 positive input.

Table 17-1. Analog Comparators Signals (100LQFP) *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
C1-	91	I	Analog	Analog comparator 1 negative input.
C1o	24	O	TTL	Analog comparator 1 output.
C2+	23	I	Analog	Analog comparator 2 positive input.
C2-	22	I	Analog	Analog comparator 2 negative input.
C2o	23	O	TTL	Analog comparator 2 output.

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

Table 17-2. Analog Comparators Signals (108BGA)

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
C0+	A7	I	Analog	Analog comparator 0 positive input.
C0-	A6	I	Analog	Analog comparator 0 negative input.
C0o	A7	O	TTL	Analog comparator 0 output.
C1+	M1	I	Analog	Analog comparator 1 positive input.
C1-	B7	I	Analog	Analog comparator 1 negative input.
C1o	M1	O	TTL	Analog comparator 1 output.
C2+	M2	I	Analog	Analog comparator 2 positive input.
C2-	L2	I	Analog	Analog comparator 2 negative input.
C2o	M2	O	TTL	Analog comparator 2 output.

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

17.3 Functional Description

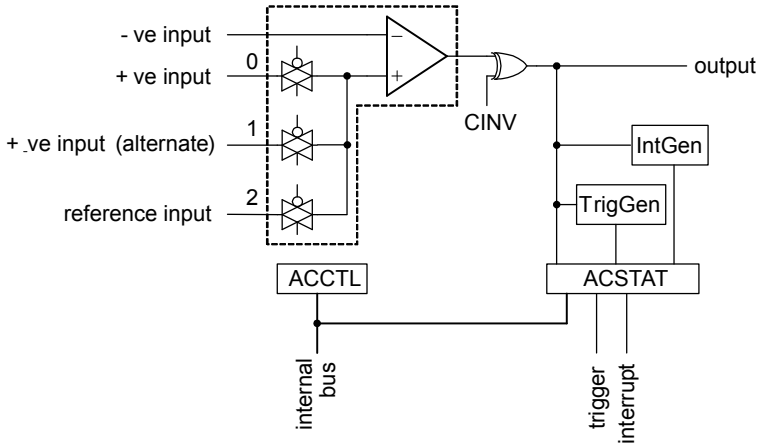
Important: It is recommended that the Digital-Input enable (the GPIODEN bit in the GPIO module) for the analog input pin be disabled to prevent excessive current draw from the I/O pads.

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

```
VIN- < VIN+, VOUT = 1
VIN- > VIN+, VOUT = 0
```

As shown in Figure 17-2 on page 647, the input source for VIN- is an external input. In addition to an external input, input sources for VIN+ can be the +ve input of comparator 0 or an internal reference.

Figure 17-2. Structure of Comparator Unit



A comparator is configured through two status/control registers (**ACCTL** and **ACSTAT**). The internal reference is configured through one control register (**ACREFCTL**). Interrupt status and control is configured through three registers (**ACMIS**, **ACRIS**, and **ACINTEN**).

Typically, the comparator output is used internally to generate controller interrupts. It may also be used to drive an external pin or generate an analog-to-digital converter (ADC) trigger.

Important: The ASRCP bits in the **ACCTLn** register must be set before using the analog comparators.

17.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 17-3 on page 647. This is controlled by a single configuration register (**ACREFCTL**). Table 17-3 on page 647 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally.

Figure 17-3. Comparator Internal Reference Structure

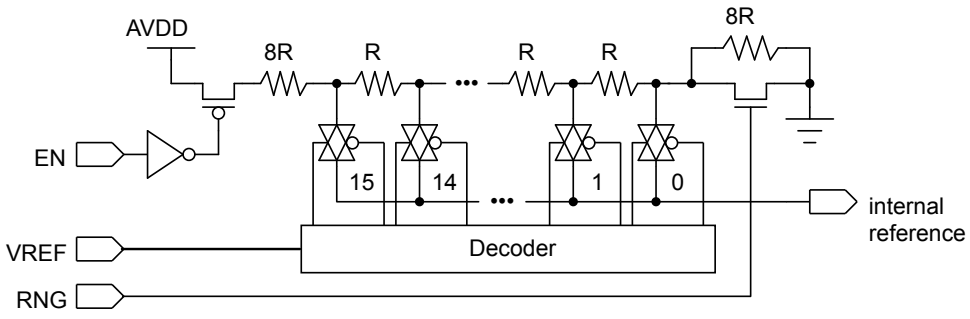


Table 17-3. Internal Reference Voltage and ACREFCTL Field Values

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN=0	RNG=X	0 V (GND) for any value of VREF; however, it is recommended that RNG=1 and VREF=0 for the least noisy ground reference.

Table 17-3. Internal Reference Voltage and ACREFCTL Field Values *(continued)*

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	**RNG Bit Value**	
EN=1	RNG=0	Total resistance in ladder is 31 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{31}$ $V_{REF} = 0.85 + 0.106 \times VREF$ The range of internal reference in this mode is 0.85-2.448 V.
	RNG=1	Total resistance in ladder is 23 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{VREF}{23}$ $V_{REF} = 0.143 \times VREF$ The range of internal reference for this mode is 0-2.152 V.

17.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator 0 clock by writing a value of 0x0010.0000 to the **RCGC1** register in the System Control module.
2. In the GPIO module, enable the GPIO port/pin associated with C0- as a GPIO input.
3. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
4. Configure comparator 0 to use the internal voltage reference and to *not* invert the output by writing the **ACCTL0** register with the value of 0x0000.040C.
5. Delay for some time.
6. Read the comparator output value by reading the **ACSTAT0** register's OVAL value.

Change the level of the signal input on C0- to see the OVAL value change.

17.5 Register Map

Table 17-4 on page 649 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000.

Note that the analog comparator module clock must be enabled before the registers can be programmed (see page 219). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

Table 17-4. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	650
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	651
0x008	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	652
0x010	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	653
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	654
0x024	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	655
0x040	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	654
0x044	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	655
0x060	ACSTAT2	RO	0x0000.0000	Analog Comparator Status 2	654
0x064	ACCTL2	R/W	0x0000.0000	Analog Comparator Control 2	655

17.6 Register Descriptions

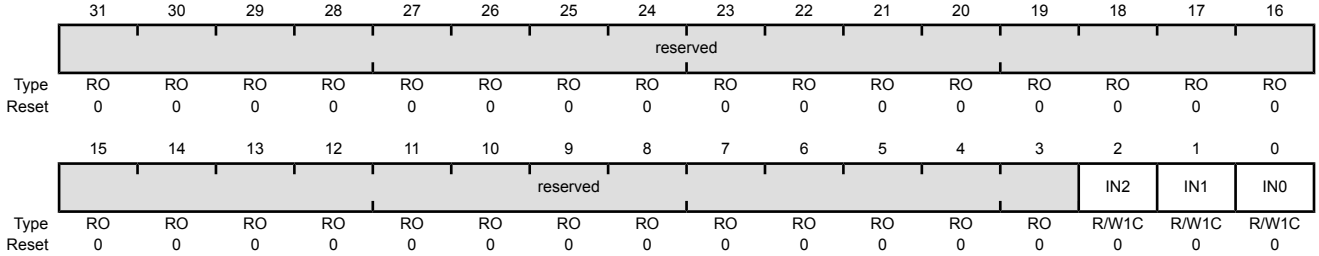
The remainder of this section lists and describes the Analog Comparator registers, in numerical order by address offset.

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparator.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x000
Type R/W1C, reset 0x0000.0000



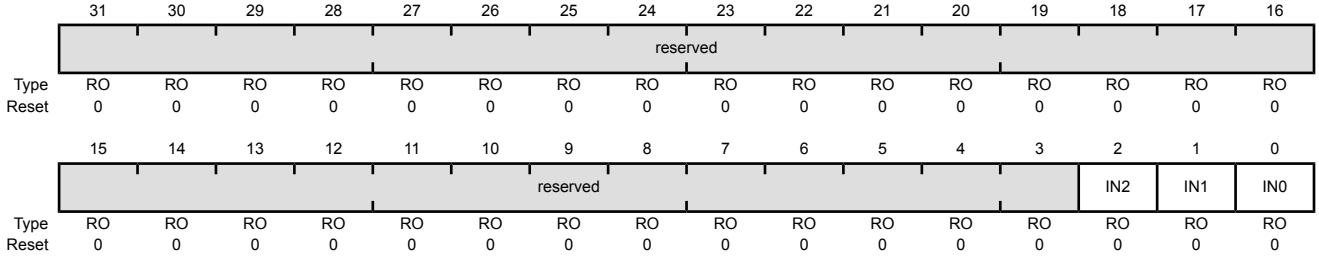
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W1C	0	Comparator 2 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparator.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x004
Type RO, reset 0x0000.0000



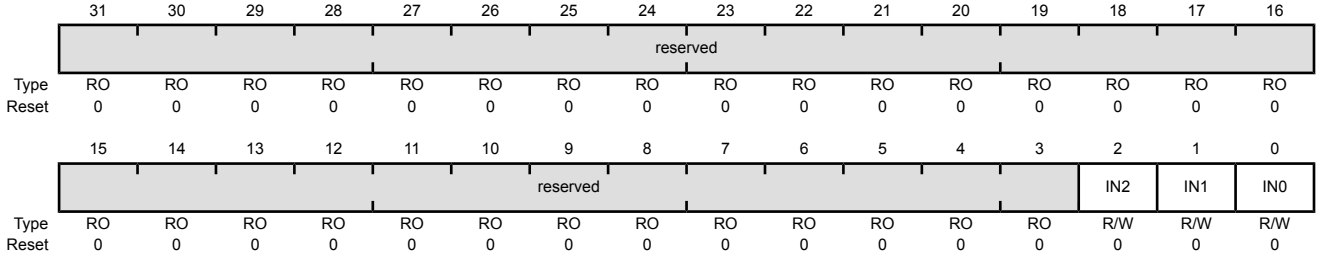
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	RO	0	Comparator 2 Interrupt Status When set, indicates that an interrupt has been generated by comparator 2.
1	IN1	RO	0	Comparator 1 Interrupt Status When set, indicates that an interrupt has been generated by comparator 1.
0	IN0	RO	0	Comparator 0 Interrupt Status When set, indicates that an interrupt has been generated by comparator 0.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparator.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
Offset 0x008
Type R/W, reset 0x0000.0000



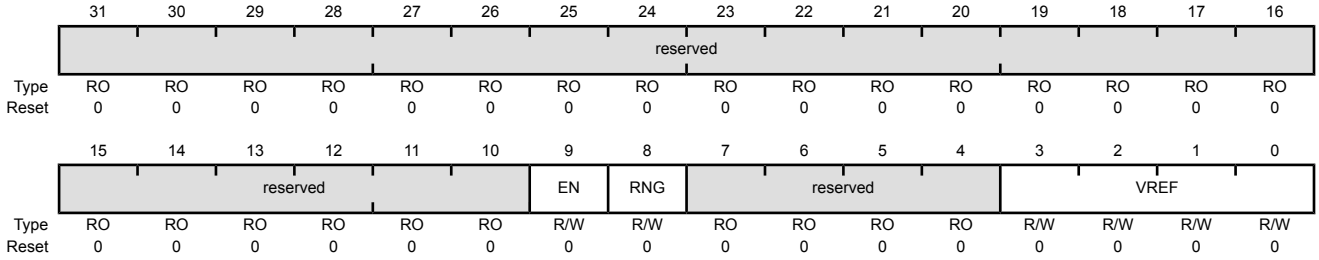
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W	0	Comparator 2 Interrupt Enable When set, enables the controller interrupt from the comparator 2 output
1	IN1	R/W	0	Comparator 1 Interrupt Enable When set, enables the controller interrupt from the comparator 1 output.
0	IN0	R/W	0	Comparator 0 Interrupt Enable When set, enables the controller interrupt from the comparator 0 output.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	EN	R/W	0	Resistor Ladder Enable The EN bit specifies whether the resistor ladder is powered on. If 0, the resistor ladder is unpowered. If 1, the resistor ladder is connected to the analog V_{DD}. This bit is reset to 0 so that the internal reference consumes the least amount of power if not used and programmed.
8	RNG	R/W	0	Resistor Ladder Range The RNG bit specifies the range of the resistor ladder. If 0, the resistor ladder has a total resistance of 31 R. If 1, the resistor ladder has a total resistance of 23 R.
7:4	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.
3:0	VREF	R/W	0x00	Resistor Ladder Voltage Ref The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 17-3 on page 647 for some output reference voltage examples.

Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020

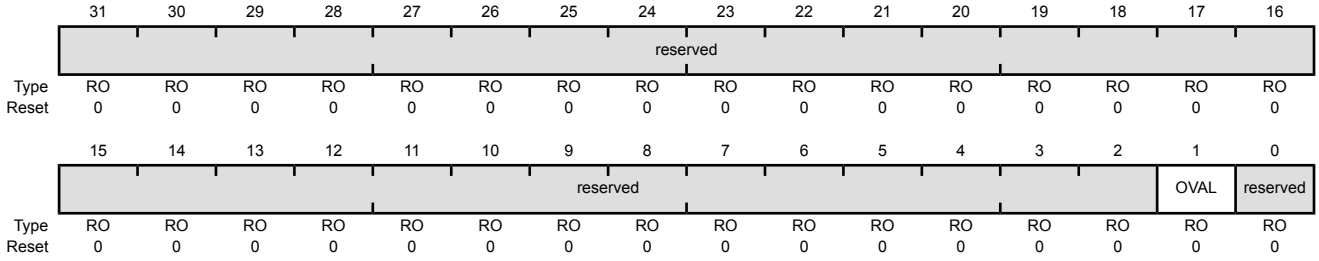
Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040

Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x060

These registers specify the current output value of the comparator.

Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000
Offset 0x020
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	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	OVAL	RO	0	Comparator Output Value The 0VAL bit specifies the current output value of the comparator.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x024

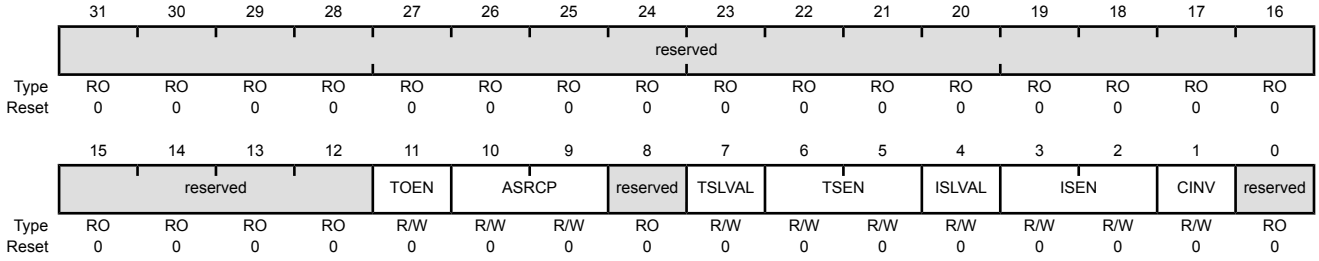
Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044

Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000
Offset 0x024
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	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.
11	TOEN	R/W	0	Trigger Output Enable The TOEN bit enables the ADC event transmission to the ADC. If 0, the event is suppressed and not sent to the ADC. If 1, the event is transmitted to the ADC.
10:9	ASRCP	R/W	0x00	Analog Source Positive The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows: Value Function 0x0 Pin value 0x1 Pin value of C0+ 0x2 Internal voltage reference 0x3 Reserved
8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TSLVAL	R/W	0	Trigger Sense Level Value The TSLVAL bit specifies the sense value of the input that generates an ADC event if in Level Sense mode. If 0, an ADC event is generated if the comparator output is Low. Otherwise, an ADC event is generated if the comparator output is High.

Bit/Field	Name	Type	Reset	Description
6:5	TSEN	R/W	0x0	Trigger Sense The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows: Value Function 0x0 Level sense, see TSLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
4	ISLVAL	R/W	0	Interrupt Sense Level Value The ISLVAL bit specifies the sense value of the input that generates an interrupt if in Level Sense mode. If 0, an interrupt is generated if the comparator output is Low. Otherwise, an interrupt is generated if the comparator output is High.
3:2	ISEN	R/W	0x0	Interrupt Sense The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows: Value Function 0x0 Level sense, see ISLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
1	CINV	R/W	0	Comparator Output Invert The CINV bit conditionally inverts the output of the comparator. If 0, the output of the comparator is unchanged. If 1, the output of the comparator is inverted prior to being processed by hardware.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

18 Pin Diagram

The LM3S8938 microcontroller pin diagrams are shown below.

Figure 18-1. 100-Pin LQFP Package Pin Diagram

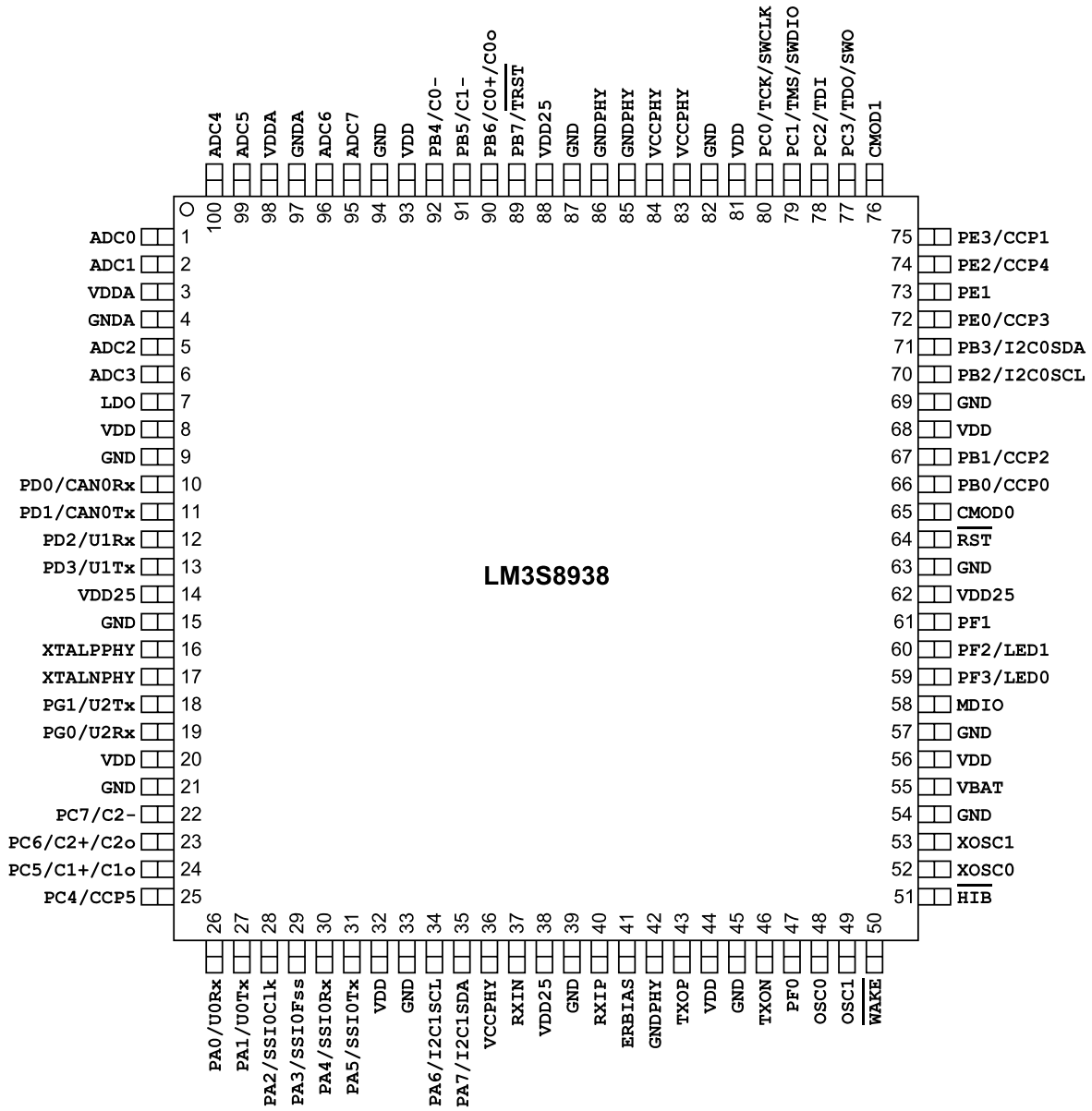
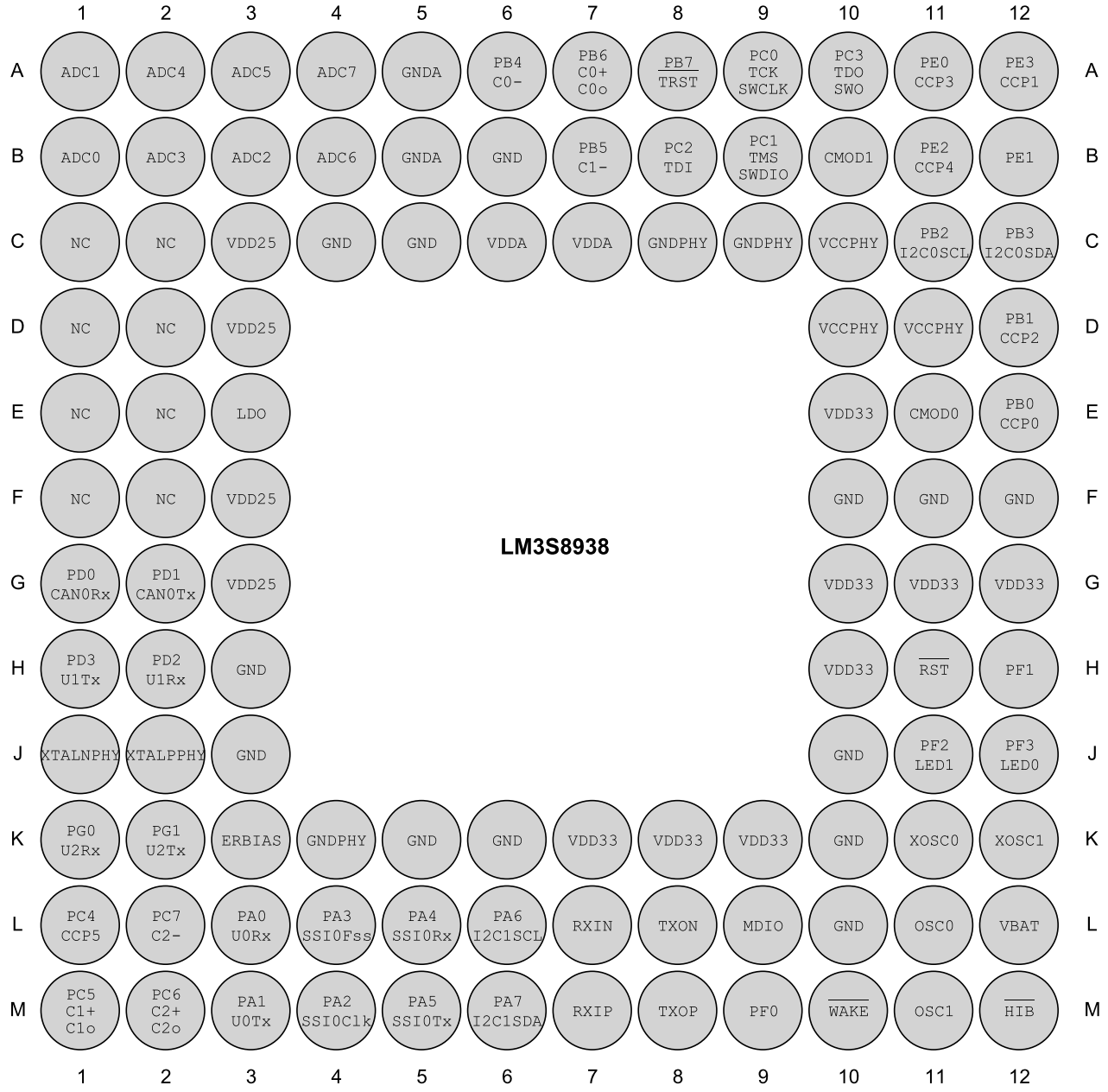


Figure 18-2. 108-Ball BGA Package Pin Diagram (Top View)



19 Signal Tables

Important: All multiplexed pins are GPIOs by default, with the exception of the five JTAG pins (PB7 and PC[3:0]) which default to the JTAG functionality.

The following tables list the signals available for each pin. Functionality is enabled by software with the **GPIOAFSEL** register. All digital inputs are Schmitt triggered.

- Signals by Pin Number
- Signals by Signal Name
- Signals by Function, Except for GPIO
- GPIO Pins and Alternate Functions
- Connections for Unused Signals

19.1 100-Pin LQFP Package Pin Tables

19.1.1 Signals by Pin Number

Table 19-1. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
1	ADC0	I	Analog	Analog-to-digital converter input 0.
2	ADC1	I	Analog	Analog-to-digital converter input 1.
3	VDDA	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in "Recommended DC Operating Conditions" on page 689, regardless of system implementation.
4	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
5	ADC2	I	Analog	Analog-to-digital converter input 2.
6	ADC3	I	Analog	Analog-to-digital converter input 3.
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. The LDO pin must also be connected to the VDD25 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.
	CAN0Rx	I	TTL	CAN module 0 receive.
11	PD1	I/O	TTL	GPIO port D bit 1.
	CAN0Tx	O	TTL	CAN module 0 transmit.
12	PD2	I/O	TTL	GPIO port D bit 2.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.

Table 19-1. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
13	PD3	I/O	TTL	GPIO port D bit 3.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
14	VDD25	-	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	XTALPPHY	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.
17	XTALNPHY	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
18	PG1	I/O	TTL	GPIO port G bit 1.
	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.
	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.
	C2-	I	Analog	Analog comparator 2 negative input.
23	PC6	I/O	TTL	GPIO port C bit 6.
	C2+	I	Analog	Analog comparator 2 positive input.
	C2o	O	TTL	Analog comparator 2 output.
24	PC5	I/O	TTL	GPIO port C bit 5.
	C1+	I	Analog	Analog comparator 1 positive input.
	C1o	O	TTL	Analog comparator 1 output.
25	PC4	I/O	TTL	GPIO port C bit 4.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
26	PA0	I/O	TTL	GPIO port A bit 0.
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
27	PA1	I/O	TTL	GPIO port A bit 1.
	U0Tx	O	TTL	UART module 0 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 signal.
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.

Table 19-1. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
33	GND	-	Power	Ground reference for logic and I/O pins.
34	PA6	I/O	TTL	GPIO port A bit 6.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
35	PA7	I/O	TTL	GPIO port A bit 7.
	I2C1SDA	I/O	OD	I^2C module 1 data.
36	VCCPHY	-	Power	VCC of the Ethernet PHY.
37	RXIN	I	Analog	RXIN of the Ethernet PHY.
38	VDD25	-	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	RXIP	I	Analog	RXIP of the Ethernet PHY.
41	ERBIAS	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
42	GNDPHY	-	Power	GND of the Ethernet PHY.
43	TXOP	O	Analog	TXOP of the Ethernet PHY.
44	VDD	-	Power	Positive supply for I/O and some logic.
45	GND	-	Power	Ground reference for logic and I/O pins.
46	TXON	O	Analog	TXON of the Ethernet PHY.
47	PF0	I/O	TTL	GPIO port F bit 0.
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 open-drain output with internal pull-up 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 crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
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	MDIO	I/O	TTL	MDIO of the Ethernet PHY.
59	PF3	I/O	TTL	GPIO port F bit 3.
	LED0	O	TTL	Ethernet LED 0.
60	PF2	I/O	TTL	GPIO port F bit 2.
	LED1	O	TTL	Ethernet LED 1.
61	PF1	I/O	TTL	GPIO port F bit 1.
62	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.

Table 19-1. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
63	GND	-	Power	Ground reference for logic and I/O pins.
64	RST	I	TTL	System reset input.
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.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
67	PB1	I/O	TTL	GPIO port B bit 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
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.
	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.
73	PE1	I/O	TTL	GPIO port E bit 1.
74	PE2	I/O	TTL	GPIO port E bit 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
75	PE3	I/O	TTL	GPIO port E bit 3.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
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.
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/O	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	VCCPHY	-	Power	VCC of the Ethernet PHY.
84	VCCPHY	-	Power	VCC of the Ethernet PHY.
85	GNDPHY	-	Power	GND of the Ethernet PHY.
86	GNDPHY	-	Power	GND of the Ethernet PHY.
87	GND	-	Power	Ground reference for logic and I/O pins.

Table 19-1. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
88	VDD25	-	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.
	$\overline{\text{TRST}}$	I	TTL	JTAG $\overline{\text{TRST}}$.
90	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
	C0o	O	TTL	Analog comparator 0 output.
91	PB5	I/O	TTL	GPIO port B bit 5.
	C1-	I	Analog	Analog comparator 1 negative input.
92	PB4	I/O	TTL	GPIO port B bit 4.
	C0-	I	Analog	Analog comparator 0 negative input.
93	VDD	-	Power	Positive supply for I/O and some logic.
94	GND	-	Power	Ground reference for logic and I/O pins.
95	ADC7	I	Analog	Analog-to-digital converter input 7.
96	ADC6	I	Analog	Analog-to-digital converter input 6.
97	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
98	VDDA	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in “Recommended DC Operating Conditions” on page 689, regardless of system implementation.
99	ADC5	I	Analog	Analog-to-digital converter input 5.
100	ADC4	I	Analog	Analog-to-digital converter input 4.

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

19.1.2 Signals by Signal Name

Table 19-2. Signals by Signal Name

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC0	1	I	Analog	Analog-to-digital converter input 0.
ADC1	2	I	Analog	Analog-to-digital converter input 1.
ADC2	5	I	Analog	Analog-to-digital converter input 2.
ADC3	6	I	Analog	Analog-to-digital converter input 3.
ADC4	100	I	Analog	Analog-to-digital converter input 4.
ADC5	99	I	Analog	Analog-to-digital converter input 5.
ADC6	96	I	Analog	Analog-to-digital converter input 6.
ADC7	95	I	Analog	Analog-to-digital converter input 7.
C0+	90	I	Analog	Analog comparator 0 positive input.
C0-	92	I	Analog	Analog comparator 0 negative input.
C0o	90	O	TTL	Analog comparator 0 output.

Table 19-2. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
C1+	24	I	Analog	Analog comparator 1 positive input.
C1-	91	I	Analog	Analog comparator 1 negative input.
C1o	24	O	TTL	Analog comparator 1 output.
C2+	23	I	Analog	Analog comparator 2 positive input.
C2-	22	I	Analog	Analog comparator 2 negative input.
C2o	23	O	TTL	Analog comparator 2 output.
CAN0Rx	10	I	TTL	CAN module 0 receive.
CAN0Tx	11	O	TTL	CAN module 0 transmit.
CCP0	66	I/O	TTL	Capture/Compare/PWM 0.
CCP1	75	I/O	TTL	Capture/Compare/PWM 1.
CCP2	67	I/O	TTL	Capture/Compare/PWM 2.
CCP3	72	I/O	TTL	Capture/Compare/PWM 3.
CCP4	74	I/O	TTL	Capture/Compare/PWM 4.
CCP5	25	I/O	TTL	Capture/Compare/PWM 5.
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.
ERBIAS	41	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
GND	9 15 21 33 39 45 54 57 63 69 82 87 94	-	Power	Ground reference for logic and I/O pins.
GNDA	4 97	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
GNDPHY	42 85 86	-	Power	GND of the Ethernet PHY.
HIB	51	O	OD	An open-drain output with internal pull-up that indicates the processor is in Hibernate mode.
I2C0SCL	70	I/O	OD	I^2C module 0 clock.
I2C0SDA	71	I/O	OD	I^2C module 0 data.
I2C1SCL	34	I/O	OD	I^2C module 1 clock.
I2C1SDA	35	I/O	OD	I^2C module 1 data.

Table 19-2. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
LED0	59	O	TTL	Ethernet LED 0.
LED1	60	O	TTL	Ethernet LED 1.
MDIO	58	I/O	TTL	MDIO of the Ethernet PHY.
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.
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.
PB1	67	I/O	TTL	GPIO port B bit 1.
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.
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.

Table 19-2. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
PE3	75	I/O	TTL	GPIO port E bit 3.
PF0	47	I/O	TTL	GPIO port F bit 0.
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.
PG0	19	I/O	TTL	GPIO port G bit 0.
PG1	18	I/O	TTL	GPIO port G bit 1.
$\overline{\text{RST}}$	64	I	TTL	System reset input.
RXIN	37	I	Analog	RXIN of the Ethernet PHY.
RXIP	40	I	Analog	RXIP of the Ethernet PHY.
SSI0Clk	28	I/O	TTL	SSI module 0 clock.
SSI0Fss	29	I/O	TTL	SSI module 0 frame signal.
SSI0Rx	30	I	TTL	SSI module 0 receive.
SSI0Tx	31	O	TTL	SSI module 0 transmit.
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/O	TTL	JTAG TMS and SWDIO.
$\overline{\text{TRST}}$	89	I	TTL	JTAG $\overline{\text{TRST}}$.
TXON	46	O	Analog	TXON of the Ethernet PHY.
TXOP	43	O	Analog	TXOP of the Ethernet PHY.
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.
U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	19	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	18	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
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.
VCCPHY	36 83 84	-	Power	VCC of the Ethernet PHY.

Table 19-2. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
VDD	8 20 32 44 56 68 81 93	-	Power	Positive supply for I/O and some logic.
VDD25	14 38 62 88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDDA	3 98	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in "Recommended DC Operating Conditions" on page 689, regardless of system implementation.
$\overline{\text{WAKE}}$	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 crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
XOSC1	53	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
XTALNPHY	17	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
XTALPPHY	16	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

19.1.3 Signals by Function, Except for GPIO

Table 19-3. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC	ADC0	1	I	Analog	Analog-to-digital converter input 0.
	ADC1	2	I	Analog	Analog-to-digital converter input 1.
	ADC2	5	I	Analog	Analog-to-digital converter input 2.
	ADC3	6	I	Analog	Analog-to-digital converter input 3.
	ADC4	100	I	Analog	Analog-to-digital converter input 4.
	ADC5	99	I	Analog	Analog-to-digital converter input 5.
	ADC6	96	I	Analog	Analog-to-digital converter input 6.
	ADC7	95	I	Analog	Analog-to-digital converter input 7.

Table 19-3. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Analog Comparators	C0+	90	I	Analog	Analog comparator 0 positive input.
	C0-	92	I	Analog	Analog comparator 0 negative input.
	C0o	90	O	TTL	Analog comparator 0 output.
	C1+	24	I	Analog	Analog comparator 1 positive input.
	C1-	91	I	Analog	Analog comparator 1 negative input.
	C1o	24	O	TTL	Analog comparator 1 output.
	C2+	23	I	Analog	Analog comparator 2 positive input.
	C2-	22	I	Analog	Analog comparator 2 negative input.
	C2o	23	O	TTL	Analog comparator 2 output.
Controller Area Network	CAN0Rx	10	I	TTL	CAN module 0 receive.
	CAN0Tx	11	O	TTL	CAN module 0 transmit.
Ethernet	ERBIAS	41	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
	GNDPHY	42 85 86	-	Power	GND of the Ethernet PHY.
	LED0	59	O	TTL	Ethernet LED 0.
	LED1	60	O	TTL	Ethernet LED 1.
	MDIO	58	I/O	TTL	MDIO of the Ethernet PHY.
	RXIN	37	I	Analog	RXIN of the Ethernet PHY.
	RXIP	40	I	Analog	RXIP of the Ethernet PHY.
	TXON	46	O	Analog	TXON of the Ethernet PHY.
	TXOP	43	O	Analog	TXOP of the Ethernet PHY.
	VCCPHY	36 83 84	-	Power	VCC of the Ethernet PHY.
	XTALNPHY	17	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
	XTALPPHY	16	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.
General-Purpose Timers	CCP0	66	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	75	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	67	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	72	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	74	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	25	I/O	TTL	Capture/Compare/PWM 5.

Table 19-3. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Hibernate	HIB	51	O	OD	An open-drain output with internal pull-up 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 crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
	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	34	I/O	OD	I^2C module 1 clock.
	I2C1SDA	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/O	TTL	JTAG TMS and SWDIO.
	TRST	89	I	TTL	JTAG TRST.

Table 19-3. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Power	GND	9 15 21 33 39 45 54 57 63 69 82 87 94	-	Power	Ground reference for logic and I/O pins.
	GNDA	4 97	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	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. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
	VDD	8 20 32 44 56 68 81 93	-	Power	Positive supply for I/O and some logic.
	VDD25	14 38 62 88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDDA	3 98	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in “Recommended DC Operating Conditions” on page 689, regardless of system implementation.
SSI	SSI0Clk	28	I/O	TTL	SSI module 0 clock.
	SSI0Fss	29	I/O	TTL	SSI module 0 frame signal.
	SSI0Rx	30	I	TTL	SSI module 0 receive.
	SSI0Tx	31	O	TTL	SSI module 0 transmit.

Table 19-3. 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.
	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.
	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.
	U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	19	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	18	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.

19.1.4 GPIO Pins and Alternate Functions

Table 19-4. GPIO Pins and Alternate Functions

IO	Pin Number	Multiplexed Function	Multiplexed Function
PA0	26	U0Rx	
PA1	27	U0Tx	
PA2	28	SSI0Clk	
PA3	29	SSI0Fss	
PA4	30	SSI0Rx	
PA5	31	SSI0Tx	
PA6	34	I2C1SCL	
PA7	35	I2C1SDA	
PB0	66	CCP0	
PB1	67	CCP2	
PB2	70	I2C0SCL	
PB3	71	I2C0SDA	
PB4	92	C0-	
PB5	91	C1-	
PB6	90	C0+	C0o
PB7	89	TRST	
PC0	80	TCK	SWCLK
PC1	79	TMS	SWDIO

Table 19-4. GPIO Pins and Alternate Functions *(continued)*

IO	Pin Number	Multiplexed Function	Multiplexed Function
PC2	78	TDI	
PC3	77	TDO	SWO
PC4	25	CCP5	
PC5	24	C1+	C1o
PC6	23	C2+	C2o
PC7	22	C2-	
PD0	10	CAN0Rx	
PD1	11	CAN0Tx	
PD2	12	U1Rx	
PD3	13	U1Tx	
PE0	72	CCP3	
PE1	73		
PE2	74	CCP4	
PE3	75	CCP1	
PF0	47		
PF1	61		
PF2	60	LED1	
PF3	59	LED0	
PG0	19	U2Rx	
PG1	18	U2Tx	

19.2 108-Ball BGA Package Pin Tables

19.2.1 Signals by Pin Number

Table 19-5. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
A1	ADC1	I	Analog	Analog-to-digital converter input 1.
A2	ADC4	I	Analog	Analog-to-digital converter input 4.
A3	ADC5	I	Analog	Analog-to-digital converter input 5.
A4	ADC7	I	Analog	Analog-to-digital converter input 7.
A5	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
A6	PB4	I/O	TTL	GPIO port B bit 4.
	C0-	I	Analog	Analog comparator 0 negative input.
A7	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
	C0o	O	TTL	Analog comparator 0 output.
A8	PB7	I/O	TTL	GPIO port B bit 7.
	$\overline{\text{TRST}}$	I	TTL	JTAG $\overline{\text{TRST}}$.

Table 19-5. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
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.
A12	PE3	I/O	TTL	GPIO port E bit 3.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
B1	ADC0	I	Analog	Analog-to-digital converter input 0.
B2	ADC3	I	Analog	Analog-to-digital converter input 3.
B3	ADC2	I	Analog	Analog-to-digital converter input 2.
B4	ADC6	I	Analog	Analog-to-digital converter input 6.
B5	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
B6	GND	-	Power	Ground reference for logic and I/O pins.
B7	PB5	I/O	TTL	GPIO port B bit 5.
	C1-	I	Analog	Analog comparator 1 negative input.
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/O	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.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
B12	PE1	I/O	TTL	GPIO port E bit 1.
C1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
C2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
C3	VDD25	-	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 for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in “Recommended DC Operating Conditions” on page 689, regardless of system implementation.

Table 19-5. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
C7	VDDA	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in "Recommended DC Operating Conditions" on page 689, regardless of system implementation.
C8	GNDPHY	-	Power	GND of the Ethernet PHY.
C9	GNDPHY	-	Power	GND of the Ethernet PHY.
C10	VCCPHY	-	Power	VCC of the Ethernet PHY.
C11	PB2	I/O	TTL	GPIO port B bit 2.
	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	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
D10	VCCPHY	-	Power	VCC of the Ethernet PHY.
D11	VCCPHY	-	Power	VCC of the Ethernet PHY.
D12	PB1	I/O	TTL	GPIO port B bit 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
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. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
E10	VDD33	-	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.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
F1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
F2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
F3	VDD25	-	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.
G1	PD0	I/O	TTL	GPIO port D bit 0.
	CAN0Rx	I	TTL	CAN module 0 receive.
G2	PD1	I/O	TTL	GPIO port D bit 1.
	CAN0Tx	O	TTL	CAN module 0 transmit.
G3	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.

Table 19-5. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
G10	VDD33	-	Power	Positive supply for I/O and some logic.
G11	VDD33	-	Power	Positive supply for I/O and some logic.
G12	VDD33	-	Power	Positive supply for I/O and some logic.
H1	PD3	I/O	TTL	GPIO port D bit 3.
	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.
	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	VDD33	-	Power	Positive supply for I/O and some logic.
H11	RST	I	TTL	System reset input.
H12	PF1	I/O	TTL	GPIO port F bit 1.
J1	XTALNPHY	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
J2	XTALPPHY	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.
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.
	LED1	O	TTL	Ethernet LED 1.
J12	PF3	I/O	TTL	GPIO port F bit 3.
	LED0	O	TTL	Ethernet LED 0.
K1	PG0	I/O	TTL	GPIO port G bit 0.
	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.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
K3	ERBIAS	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
K4	GNDPHY	-	Power	GND of the Ethernet PHY.
K5	GND	-	Power	Ground reference for logic and I/O pins.
K6	GND	-	Power	Ground reference for logic and I/O pins.
K7	VDD33	-	Power	Positive supply for I/O and some logic.
K8	VDD33	-	Power	Positive supply for I/O and some logic.
K9	VDD33	-	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 crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
K12	XOSC1	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.

Table 19-5. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
L1	PC4	I/O	TTL	GPIO port C bit 4.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
L2	PC7	I/O	TTL	GPIO port C bit 7.
	C2-	I	Analog	Analog comparator 2 negative input.
L3	PA0	I/O	TTL	GPIO port A bit 0.
	U0Rx	I	TTL	UART module 0 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 signal.
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.
	I2C1SCL	I/O	OD	I^2C module 1 clock.
L7	RXIN	I	Analog	RXIN of the Ethernet PHY.
L8	TXON	O	Analog	TXON of the Ethernet PHY.
L9	MDIO	I/O	TTL	MDIO of the Ethernet PHY.
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.
	C1+	I	Analog	Analog comparator 1 positive input.
	C1o	O	TTL	Analog comparator 1 output.
M2	PC6	I/O	TTL	GPIO port C bit 6.
	C2+	I	Analog	Analog comparator 2 positive input.
	C2o	O	TTL	Analog comparator 2 output.
M3	PA1	I/O	TTL	GPIO port A bit 1.
	U0Tx	O	TTL	UART module 0 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.
	I2C1SDA	I/O	OD	I^2C module 1 data.
M7	RXIP	I	Analog	RXIP of the Ethernet PHY.
M8	TXOP	O	Analog	TXOP of the Ethernet PHY.
M9	PF0	I/O	TTL	GPIO port F bit 0.
M10	$\overline{\text{WAKE}}$	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.

Table 19-5. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
M12	HIB	O	OD	An open-drain output with internal pull-up that indicates the processor is in Hibernate mode.

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

19.2.2 Signals by Signal Name

Table 19-6. Signals by Signal Name

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC0	B1	I	Analog	Analog-to-digital converter input 0.
ADC1	A1	I	Analog	Analog-to-digital converter input 1.
ADC2	B3	I	Analog	Analog-to-digital converter input 2.
ADC3	B2	I	Analog	Analog-to-digital converter input 3.
ADC4	A2	I	Analog	Analog-to-digital converter input 4.
ADC5	A3	I	Analog	Analog-to-digital converter input 5.
ADC6	B4	I	Analog	Analog-to-digital converter input 6.
ADC7	A4	I	Analog	Analog-to-digital converter input 7.
C0+	A7	I	Analog	Analog comparator 0 positive input.
C0-	A6	I	Analog	Analog comparator 0 negative input.
C0o	A7	O	TTL	Analog comparator 0 output.
C1+	M1	I	Analog	Analog comparator 1 positive input.
C1-	B7	I	Analog	Analog comparator 1 negative input.
C1o	M1	O	TTL	Analog comparator 1 output.
C2+	M2	I	Analog	Analog comparator 2 positive input.
C2-	L2	I	Analog	Analog comparator 2 negative input.
C2o	M2	O	TTL	Analog comparator 2 output.
CAN0Rx	G1	I	TTL	CAN module 0 receive.
CAN0Tx	G2	O	TTL	CAN module 0 transmit.
CCP0	E12	I/O	TTL	Capture/Compare/PWM 0.
CCP1	A12	I/O	TTL	Capture/Compare/PWM 1.
CCP2	D12	I/O	TTL	Capture/Compare/PWM 2.
CCP3	A11	I/O	TTL	Capture/Compare/PWM 3.
CCP4	B11	I/O	TTL	Capture/Compare/PWM 4.
CCP5	L1	I/O	TTL	Capture/Compare/PWM 5.
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.
ERBIAS	K3	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.

Table 19-6. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
GND	B6 C4 C5 F10 F11 F12 H3 J3 J10 K5 K6 K10 L10	-	Power	Ground reference for logic and I/O pins.
GNDA	A5 B5	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
GNDPHY	C8 C9 K4	-	Power	GND of the Ethernet PHY.
HIB	M12	O	OD	An open-drain output with internal pull-up that indicates the processor is in Hibernate mode.
I2C0SCL	C11	I/O	OD	I^2C module 0 clock.
I2C0SDA	C12	I/O	OD	I^2C module 0 data.
I2C1SCL	L6	I/O	OD	I^2C module 1 clock.
I2C1SDA	M6	I/O	OD	I^2C module 1 data.
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. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
LED0	J12	O	TTL	Ethernet LED 0.
LED1	J11	O	TTL	Ethernet LED 1.
MDIO	L9	I/O	TTL	MDIO of the Ethernet PHY.
NC	C1 C2 D1 D2 E1 E2 F1 F2	-	-	No connect. Leave the pin electrically unconnected/isolated.
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.
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 19-6. Signals by Signal Name *(continued)*

Pin Name	Pin Number	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.
PB1	D12	I/O	TTL	GPIO port B bit 1.
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.
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.
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.
PG0	K1	I/O	TTL	GPIO port G bit 0.
PG1	K2	I/O	TTL	GPIO port G bit 1.
$\overline{\text{RST}}$	H11	I	TTL	System reset input.
RXIN	L7	I	Analog	RXIN of the Ethernet PHY.
RXIP	M7	I	Analog	RXIP of the Ethernet PHY.
SSI0Clk	M4	I/O	TTL	SSI module 0 clock.
SSI0Fss	L4	I/O	TTL	SSI module 0 frame signal.
SSI0Rx	L5	I	TTL	SSI module 0 receive.
SSI0Tx	M5	O	TTL	SSI module 0 transmit.
SWCLK	A9	I	TTL	JTAG/SWD CLK.

Table 19-6. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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/O	TTL	JTAG TMS and SWDIO.
TRST	A8	I	TTL	JTAG TRST.
TXON	L8	O	Analog	TXON of the Ethernet PHY.
TXOP	M8	O	Analog	TXOP of the Ethernet PHY.
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.
U1Rx	H2	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	H1	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	K1	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	K2	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
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.
VCCPHY	C10 D10 D11	-	Power	VCC of the Ethernet PHY.
VDD25	C3 D3 F3 G3	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD33	E10 G10 G11 G12 H10 K7 K8 K9	-	Power	Positive supply for I/O and some logic.
VDDA	C6 C7	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in “Recommended DC Operating Conditions” on page 689, regardless of system implementation.
$\overline{\text{WAKE}}$	M10	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.

Table 19-6. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
XOSC0	K11	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
XOSC1	K12	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
XTALNPHY	J1	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
XTALPPHY	J2	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.

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

19.2.3 Signals by Function, Except for GPIO

Table 19-7. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC	ADC0	B1	I	Analog	Analog-to-digital converter input 0.
	ADC1	A1	I	Analog	Analog-to-digital converter input 1.
	ADC2	B3	I	Analog	Analog-to-digital converter input 2.
	ADC3	B2	I	Analog	Analog-to-digital converter input 3.
	ADC4	A2	I	Analog	Analog-to-digital converter input 4.
	ADC5	A3	I	Analog	Analog-to-digital converter input 5.
	ADC6	B4	I	Analog	Analog-to-digital converter input 6.
	ADC7	A4	I	Analog	Analog-to-digital converter input 7.
Analog Comparators	C0+	A7	I	Analog	Analog comparator 0 positive input.
	C0-	A6	I	Analog	Analog comparator 0 negative input.
	C0o	A7	O	TTL	Analog comparator 0 output.
	C1+	M1	I	Analog	Analog comparator 1 positive input.
	C1-	B7	I	Analog	Analog comparator 1 negative input.
	C1o	M1	O	TTL	Analog comparator 1 output.
	C2+	M2	I	Analog	Analog comparator 2 positive input.
	C2-	L2	I	Analog	Analog comparator 2 negative input.
	C2o	M2	O	TTL	Analog comparator 2 output.
Controller Area Network	CAN0Rx	G1	I	TTL	CAN module 0 receive.
	CAN0Tx	G2	O	TTL	CAN module 0 transmit.

Table 19-7. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Ethernet	ERBIAS	K3	I	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
	GNDPHY	C8 C9 K4	-	Power	GND of the Ethernet PHY.
	LED0	J12	O	TTL	Ethernet LED 0.
	LED1	J11	O	TTL	Ethernet LED 1.
	MDIO	L9	I/O	TTL	MDIO of the Ethernet PHY.
	RXIN	L7	I	Analog	RXIN of the Ethernet PHY.
	RXIP	M7	I	Analog	RXIP of the Ethernet PHY.
	TXON	L8	O	Analog	TXON of the Ethernet PHY.
	TXOP	M8	O	Analog	TXOP of the Ethernet PHY.
	VCCPHY	C10 D10 D11	-	Power	VCC of the Ethernet PHY.
	XTALNPHY	J1	O	TTL	Ethernet PHY XTALN 25-MHz oscillator crystal output. Connect this pin to ground when using a single-ended 25-MHz clock input connected to the XTALPPHY pin.
	XTALPPHY	J2	I	TTL	Ethernet PHY XTALP 25-MHz oscillator crystal input or external clock reference input.
General-Purpose Timers	CCP0	E12	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	A12	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	D12	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	A11	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	B11	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	L1	I/O	TTL	Capture/Compare/PWM 5.
Hibernate	HIB	M12	O	OD	An open-drain output with internal pull-up 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 crystal or a 32.768-kHz oscillator for the Hibernation module RTC.
	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	L6	I/O	OD	I^2C module 1 clock.
	I2C1SDA	M6	I/O	OD	I^2C module 1 data.

Table 19-7. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
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/O	TTL	JTAG TMS and SWDIO.
	TRST	A8	I	TTL	JTAG TRST.
Power	GND	B6 C4 C5 F10 F11 F12 H3 J3 J10 K5 K6 K10 L10	-	Power	Ground reference for logic and I/O pins.
	GNDA	A5 B5	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	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. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
	VDD25	C3 D3 F3 G3	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD33	E10 G10 G11 G12 H10 K7 K8 K9	-	Power	Positive supply for I/O and some logic.
	VDDA	C6 C7	-	Power	The positive supply for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be supplied with a voltage that meets the specification in “Recommended DC Operating Conditions” on page 689, regardless of system implementation.

Table 19-7. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
SSI	SSI0Clk	M4	I/O	TTL	SSI module 0 clock.
	SSI0Fss	L4	I/O	TTL	SSI module 0 frame signal.
	SSI0Rx	L5	I	TTL	SSI module 0 receive.
	SSI0Tx	M5	O	TTL	SSI module 0 transmit.
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.
	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.
	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.
	U1Rx	H2	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	H1	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	K1	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	K2	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.

19.2.4 GPIO Pins and Alternate Functions

Table 19-8. GPIO Pins and Alternate Functions

IO	Pin Number	Multiplexed Function	Multiplexed Function
PA0	L3	U0Rx	
PA1	M3	U0Tx	
PA2	M4	SSI0Clk	
PA3	L4	SSI0Fss	
PA4	L5	SSI0Rx	
PA5	M5	SSI0Tx	
PA6	L6	I2C1SCL	
PA7	M6	I2C1SDA	
PB0	E12	CCP0	
PB1	D12	CCP2	
PB2	C11	I2C0SCL	
PB3	C12	I2C0SDA	
PB4	A6	C0-	
PB5	B7	C1-	

Table 19-8. GPIO Pins and Alternate Functions *(continued)*

IO	Pin Number	Multiplexed Function	Multiplexed Function
PB6	A7	C0+	C0o
PB7	A8	TRST	
PC0	A9	TCK	SWCLK
PC1	B9	TMS	SWDIO
PC2	B8	TDI	
PC3	A10	TDO	SWO
PC4	L1	CCP5	
PC5	M1	C1+	C1o
PC6	M2	C2+	C2o
PC7	L2	C2-	
PD0	G1	CAN0Rx	
PD1	G2	CAN0Tx	
PD2	H2	U1Rx	
PD3	H1	U1Tx	
PE0	A11	CCP3	
PE1	B12		
PE2	B11	CCP4	
PE3	A12	CCP1	
PF0	M9		
PF1	H12		
PF2	J11	LED1	
PF3	J12	LED0	
PG0	K1	U2Rx	
PG1	K2	U2Tx	

19.3 Connections for Unused Signals

Table 19-9 on page 685 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 19-9. Connections for Unused Signals (100-pin LQFP)

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
ADC	ADC0	1	NC	GNDA
	ADC1	2		
	ADC2	5		
	ADC3	6		
	ADC4	100		
	ADC5	99		
	ADC6	96		
	ADC7	95		

Table 19-9. Connections for Unused Signals (100-pin LQFP) *(continued)*

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
Ethernet	ERBIAS	41	Connect to GND through 12.4-kΩ resistor.	Connect to GND through 12.4-kΩ resistor.
	GNDPHY	42 85 86	GND	GND
	MDIO[a]	58	NC	NC
	RXIN	37	NC	GND
	RXIP	40	NC	GND
	TXON	46	NC	GND
	TXOP	43	NC	GND
	VCCPHY	36 83 84	VDD	VDD
	XTALNPHY[a]	17	NC	NC
	XTALPPHY[a]	16	NC	GND
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 173	Connect through a capacitor to GND as close to pin as possible

a. Note that the Ethernet PHY is powered up by default. The PHY cannot be powered down unless a clock source is provided and the MDIO pin is pulled up through a 10-kΩ resistor.

Table 19-10 on page 687 show how to handle signals for functions that are not used in a particular system implementation for devices that are in a 108-pin 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 19-10. Connections for Unused Signals, 108-pin BGA

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
ADC	ADC0	B1	NC	GNDA
	ADC1	A1		
	ADC2	B3		
	ADC3	B2		
	ADC4	A2		
	ADC5	A3		
	ADC6	B4		
	ADC7	A4		
Ethernet	ERBIAS	K3	Connect to GND through 12.4-kΩ resistor.	Connect to GND through 12.4-kΩ resistor.
	GNDPHY	C8	GND	GND
		C9		
		K4		
	MDIO[a]	L9	NC	NC
	RXIN	L7	NC	GND
	RXIP	M7	NC	GND
	TXON	L8	NC	GND
	TXOP	M8	NC	GND
	VCCPHY	C10	VDD	VDD
		D10		
		D11		
	XTALNPHY[a]	J1	NC	NC
	XTALPPHY[a]	J2	NC	GND
GPIO	All unused GPIOs	-	NC	GND
Hibernate	$\overline{\text{HIB}}$	M12	NC	NC
	VBAT	L12	NC	GND
	$\overline{\text{WAKE}}$	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
	$\overline{\text{RST}}$	H11	Pull up as shown in Figure 5-1 on page 173	Connect through a capacitor to GND as close to pin as possible

a. Note that the Ethernet PHY is powered up by default. The PHY cannot be powered down unless a clock source is provided and the MDIO pin is pulled up through a 10-kΩ resistor.

20 Operating Characteristics

Table 20-1. Temperature Characteristics

Characteristic	Symbol	Value	Unit
Industrial operating temperature range	T_A	-40 to +85	°C
Extended operating temperature range	T_A	-40 to +105	°C
Unpowered storage temperature range	T_S	-65 to +150	°C

Table 20-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient)[a]	Θ_{JA}	32	°C/W
Junction temperature[b]	T_J	$T_A + (P \bullet \Theta_{JA})$	°C

a. Junction to ambient thermal resistance θ_{JA} numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

Table 20-3. ESD Absolute Maximum Ratings[a]

Parameter Name	Min	Nom	Max	Unit
V_{ESDHBM}	-	-	2.0	kV
V_{ESDCDM}	-	-	1.0	kV
V_{ESDMM}	-	-	100	V

a. All Stellaris parts are ESD tested following the JEDEC standard.

21 Electrical Characteristics

21.1 DC Characteristics

21.1.1 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 21-1. Maximum Ratings

Characteristic[a]	Symbol	Value		Unit
		Min	Max	
I/O supply voltage (V_{DD})	V_{DD}	0	4	V
Core supply voltage (V_{DD25})	V_{DD25}	0	3	V
Analog supply voltage (V_{DDA})	V_{DDA}	0	4	V
Battery supply voltage (V_{BAT})	V_{BAT}	0	4	V
Ethernet PHY supply voltage (V_{CCPHY})	V_{CCPHY}	0	4	V
Input voltage	V_{IN}	-0.3	5.5	V
Input voltage for a GPIO configured as an analog input		-0.3	V_{DD} + 0.3	V
Maximum current per output pins	I	-	25	mA
Maximum input voltage on a non-power pin when the microcontroller is unpowered	V_{NON}	-	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 (for example, either GND or V_{DD}).

21.1.2 Recommended DC 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 21-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	I/O supply voltage	3.0	3.3	3.6	V
V_{DD25}	Core supply voltage	2.25	2.5	2.75	V
V_{DDA}	Analog supply voltage	3.0	3.3	3.6	V
V_{BAT}	Battery supply voltage	2.3	3.0	3.6	V
V_{CCPHY}	Ethernet PHY supply voltage	3.0	3.3	3.6	V

Table 21-2. Recommended DC Operating Conditions *(continued)*

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{IH}	High-level input voltage	2.0	-	5.0	V
V_{IL}	Low-level input voltage	-0.3	-	1.3	V
V_{OH}[a]	High-level output voltage	2.4	-	-	V
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

a. V_{OL} and V_{OH} shift to 1.2 V when using high-current GPIOs.

21.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 21-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{LDOOUT}	Programmable internal (logic) power supply output value	2.25	2.5	2.75	V
	Output voltage accuracy	-	2%	-	%
t_{PON}	Power-on time	-	-	100	µs
t_{ON}	Time on	-	-	200	µs
t_{OFF}	Time off	-	-	100	µs
V_{STEP}	Step programming incremental voltage	-	50	-	mV
C_{LDO}	External filter capacitor size for internal power supply	1.0	-	3.0	µF

21.1.4 GPIO Module Characteristics

Table 21-4. GPIO Module DC Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{GPIOPU}	GPIO internal pull-up resistor	50	-	110	kΩ
R_{GPIOPD}	GPIO internal pull-down resistor	55	-	180	kΩ
I_{LKG}	GPIO input leakage current[a]	-	-	2	µA

a. The leakage current is measured with GND or V_{DD} 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.

21.1.5 Power Specifications

The power measurements specified in the tables that follow are run on the core processor using SRAM with the following specifications (except as noted):

- V_{DD} = 3.3 V

- V_{DD25} = 2.50 V
- V_{BAT} = 3.0 V
- V_{DDA} = 3.3 V
- V_{DDPHY} = 3.3 V
- Temperature = 25°C
- Clock Source (MOSC) =3.579545 MHz Crystal Oscillator
- Main oscillator (MOSC) = enabled
- Internal oscillator (IOSC) = disabled

Table 21-5. Detailed Power Specifications

Parameter	Parameter Name	Conditions	3.3 V V_{DD}, V_{DDA}, V_{DDPHY}		2.5 V V_{DD25}		3.0 V V_{BAT}		Unit
			Nom	Max	Nom	Max	Nom	Max	
I_{DD_RUN}	Run mode 1 (Flash loop)	V_{DD25} = 2.50 V Code= while(1){} executed out of Flash Peripherals = All ON System Clock = 50 MHz (with PLL)	48	pending[a]	108	pending[a]	0	pending[a]	mA
	Run mode 2 (Flash loop)	V_{DD25} = 2.50 V Code= while(1){} executed out of Flash Peripherals = All OFF System Clock = 50 MHz (with PLL)	5	pending[a]	52	pending[a]	0	pending[a]	mA
	Run mode 1 (SRAM loop)	V_{DD25} = 2.50 V Code= while(1){} executed in SRAM Peripherals = All ON System Clock = 50 MHz (with PLL)	48	pending[a]	100	pending[a]	0	pending[a]	mA
	Run mode 2 (SRAM loop)	V_{DD25} = 2.50 V Code= while(1){} executed in SRAM Peripherals = All OFF System Clock = 50 MHz (with PLL)	5	pending[a]	45	pending[a]	0	pending[a]	mA
I_{DD_SLEEP}	Sleep mode	V_{DD25} = 2.50 V Peripherals = All OFF System Clock = 50 MHz (with PLL)	5	pending[a]	16	pending[a]	0	pending[a]	mA
$I_{DD_DEEPSLEEP}$	Deep-Sleep mode	LDO = 2.25 V Peripherals = All OFF System Clock = IOSC30KHZ/64	4.6	pending[a]	0.21	pending[a]	0	pending[a]	mA

Table 21-5. Detailed Power Specifications *(continued)*

Parameter	Parameter Name	Conditions	3.3 V V_{DD}, V_{DDA}, V_{DDPHY}		2.5 V V_{DD25}		3.0 V V_{BAT}		Unit
			Nom	Max	Nom	Max	Nom	Max	
$I_{DD_HIBERNATE}$	Hibernate mode	V_{BAT} = 3.0 V V_{DD} = 0 V V_{DD25} = 0 V V_{DDA} = 0 V V_{DDPHY} = 0 V Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz	0	0	0	0	16	pending[a]	µA

a. Pending characterization completion.

21.1.6 Flash Memory Characteristics

Table 21-6. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{CYC}	Number of guaranteed program/erase cycles before failure[a]	10,000	100,000	-	cycles
T_{RET}	Data retention at average operating temperature of 85˚C (industrial) or 105˚C (extended)	10	-	-	years
T_{PROG}	Word program time	20	-	-	µs
T_{ERASE}	Page erase time	20	-	-	ms
T_{ME}	Mass erase time	-	-	250	ms

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

21.1.7 Hibernation

Table 21-7. Hibernation Module DC Characteristics

Parameter	Parameter Name	Value	Unit
V_{LOWBAT}	Low battery detect voltage	2.35	V
R_{WAKEPU}	$\overline{WAKE}$ internal pull-up resistor	200	kΩ

21.1.8 Ethernet Controller

Table 21-8. Ethernet Controller DC Characteristics

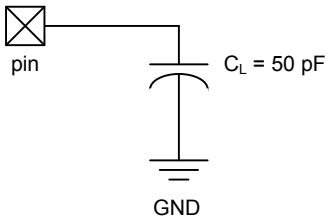
Parameter	Parameter Name	Value	Unit
R_{EBIAS}	Value of the pull-down resistor on the ERBIAS pin	12.4K ± 1 %	Ω

21.2 AC Characteristics

21.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements. Timing measurements are for 4-mA drive strength.

Figure 21-1. Load Conditions



21.2.2 Clocks

Table 21-9. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{ref_crystal}$	Crystal reference[a]	3.579545	-	8.192	MHz
f_{ref_ext}	External clock reference[a]	3.579545	-	8.192	MHz
f_{pll}	PLL frequency[b]	-	400	-	MHz
T_{READY}	PLL lock time	-	-	0.5	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.

Table 21-10 on page 693 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

Table 21-10. Actual PLL Frequency

XTAL	Crystal Frequency (MHz)	PLL Frequency (MHz)	Error
0x4	3.5795	400.904	0.0023%
0x5	3.6864	398.1312	0.0047%
0x6	4.0	400	-
0x7	4.096	401.408	0.0035%
0x8	4.9152	398.1312	0.0047%
0x9	5.0	400	-
0xA	5.12	399.36	0.0016%
0xB	6.0	400	-
0xC	6.144	399.36	0.0016%
0xD	7.3728	398.1312	0.0047%
0xE	8.0	400	0.0047%
0xF	8.192	398.6773333	0.0033%

Table 21-11. Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{IOSC}	Internal 12 MHz oscillator frequency	8.4	12	15.6	MHz
$f_{IOSC30KHZ}$	Internal 30 KHz oscillator frequency	15	30	45	KHz
f_{XOSC}	Hibernation module oscillator frequency	-	4.194304	-	MHz
f_{XOSC_XTAL}	Crystal reference for hibernation oscillator	-	4.194304	-	MHz

Table 21-11. Clock Characteristics *(continued)*

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{XOSC_EXT}	External clock reference for hibernation module	-	32.768	-	KHz
f_{MOSC}	Main oscillator frequency	1	-	8.192	MHz
t_{MOSC_per}	Main oscillator period	125	-	1000	ns
$f_{ref_crystal_bypass}$	Crystal reference using the main oscillator (PLL in BYPASS mode)[a]	1	-	8.192	MHz
$f_{ref_ext_bypass}$	External clock reference (PLL in BYPASS mode)[a]	0	-	50	MHz
f_{system_clock}	System clock	0	-	50	MHz

a. The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Table 21-12. Crystal Characteristics

Parameter Name	Value				Units
Frequency	8	6	4	3.5	MHz
Frequency tolerance[a]	±50	±50	±50	±50	ppm
Oscillation mode	Parallel	Parallel	Parallel	Parallel	-
Motional capacitance (typ)	27.8	37.0	55.6	63.5	pF
Motional inductance (typ)	14.3	19.1	28.6	32.7	mH
Equivalent series resistance (max)	120	160	200	220	Ω
Shunt capacitance (max)	10	10	10	10	pF
Load capacitance (typ)	16	16	16	16	pF
Drive level (typ)	100	100	100	100	µW

a. This tolerance provides a guard band for temperature stability and aging drift.

21.2.2.1 System Clock Specifications with ADC Operation

Table 21-13. System Clock Characteristics with ADC Operation

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{sysadc}	System clock frequency when the ADC module is operating (when PLL is bypassed)	16	-	-	MHz

21.2.3 JTAG and Boundary Scan

Table 21-14. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	f_{TCK}	TCK operational clock frequency	0	-	10	MHz
J2	t_{TCK}	TCK operational clock period	100	-	-	ns
J3	t_{TCK_LOW}	TCK clock Low time	-	$t_{TCK}/2$	-	ns
J4	t_{TCK_HIGH}	TCK clock High time	-	$t_{TCK}/2$	-	ns
J5	t_{TCK_R}	TCK rise time	0	-	10	ns
J6	t_{TCK_F}	TCK fall time	0	-	10	ns
J7	t_{TMS_SU}	TMS setup time to TCK rise	20	-	-	ns
J8	t_{TMS_HLD}	TMS hold time from TCK rise	20	-	-	ns

Table 21-14. JTAG Characteristics *(continued)*

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J9	t_{TDI_SU}	TDI setup time to TCK rise	25	-	-	ns
J10	t_{TDI_HLD}	TDI hold time from TCK rise	25	-	-	ns
J11 t_{TDO_ZDV}	TCK fall to Data Valid from High-Z	2-mA drive	-	23	35	ns
		4-mA drive		15	26	ns
		8-mA drive		14	25	ns
		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
		4-mA drive		14	25	ns
		8-mA drive		13	24	ns
		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
		4-mA drive		7	9	ns
		8-mA drive		6	8	ns
		8-mA drive with slew rate control		7	9	ns
J14	t_{TRST}	TRST assertion time	100	-	-	ns
J15	t_{TRST_SU}	TRST setup time to TCK rise	10	-	-	ns

Figure 21-2. JTAG Test Clock Input Timing

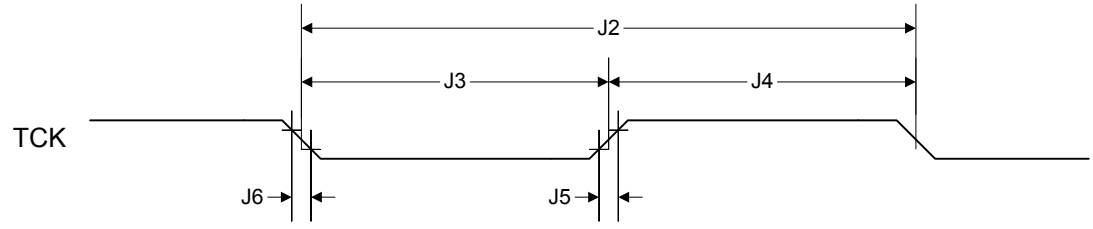


Figure 21-3. JTAG Test Access Port (TAP) Timing

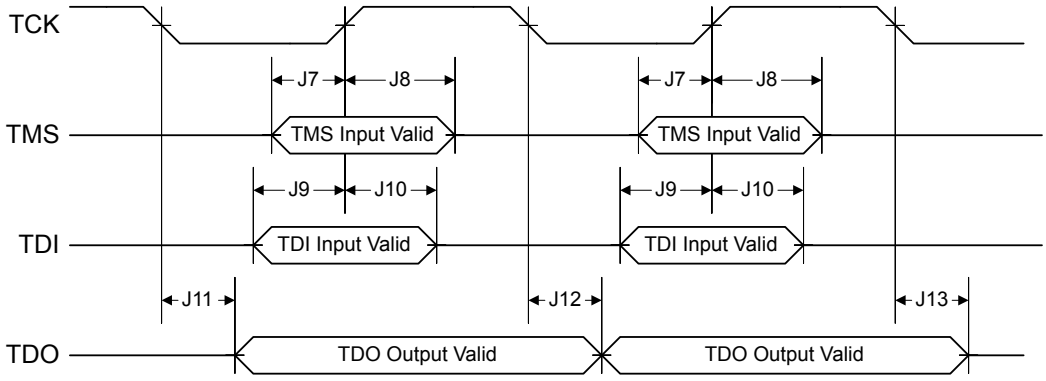
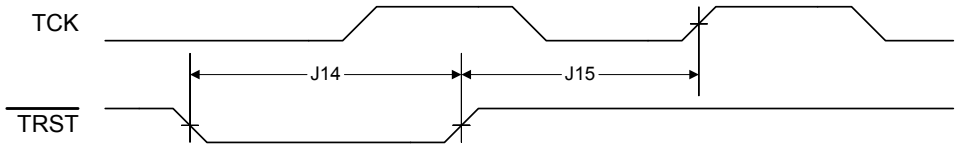


Figure 21-4. JTAG TRST Timing



21.2.4 Reset

Table 21-15. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V_{TH}	Reset threshold	-	2.0	-	V
R2	V_{BTH}	Brown-Out threshold	2.85	2.9	2.95	V
R3	T_{POR}	Power-On Reset timeout	-	10	-	ms
R4	T_{BOR}	Brown-Out timeout	-	500	-	µs
R5	T_{IRPOR}	Internal reset timeout after POR	6	-	11	ms
R6	T_{IRBOR}	Internal reset timeout after BOR[a]	0	-	1	µs
R7	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{RST}$ pin)	0	-	1	ms
R8	T_{IRSWR}	Internal reset timeout after software-initiated system reset [a]	2.5	-	20	µs
R9	T_{IRWDR}	Internal reset timeout after watchdog reset[a]	2.5	-	20	µs
R10	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0V-3.3V), power on reset	-	-	100	ms
		Supply voltage (V_{DD}) rise time (0V-3.3V), waking from hibernation	-	-	250	µs
R11	T_{MIN}	Minimum $\overline{RST}$ pulse width	2	-	-	µs

a. 20 * t_{MOSC_per}

Figure 21-5. External Reset Timing ($\overline{RST}$)

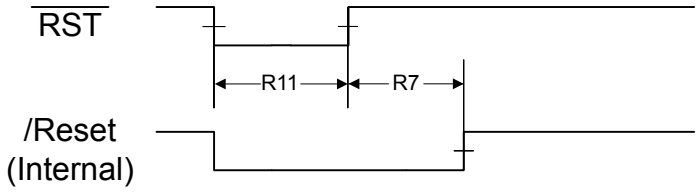


Figure 21-6. Power-On Reset Timing

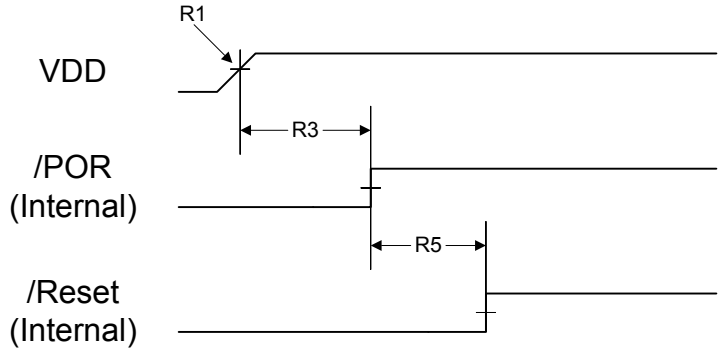


Figure 21-7. Brown-Out Reset Timing

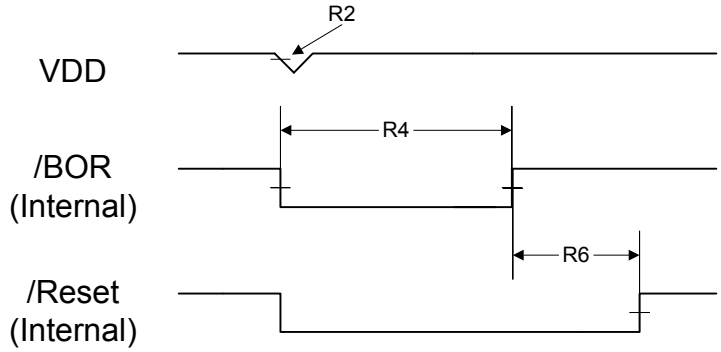


Figure 21-8. Software Reset Timing

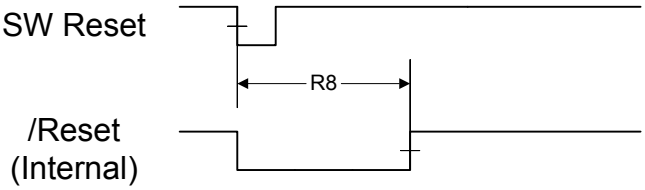
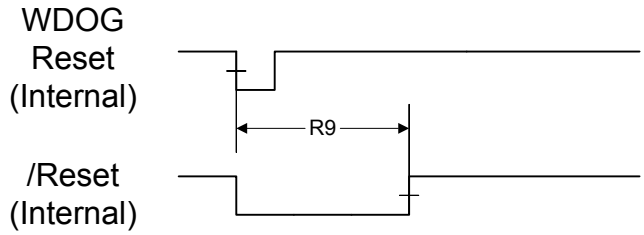


Figure 21-9. Watchdog Reset Timing



21.2.5 Sleep Modes

Table 21-16. 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

a. Values in this table assume the IOSC is the clock source during sleep or deep-sleep mode.

21.2.6 Hibernation Module

The Hibernation Module requires special system implementation considerations since it is intended to power-down all other sections of its host device. The system power-supply distribution and interfaces to the device must be driven to 0 V_{DC} or powered down with the same external voltage regulator controlled by $\overline{HIB}$.

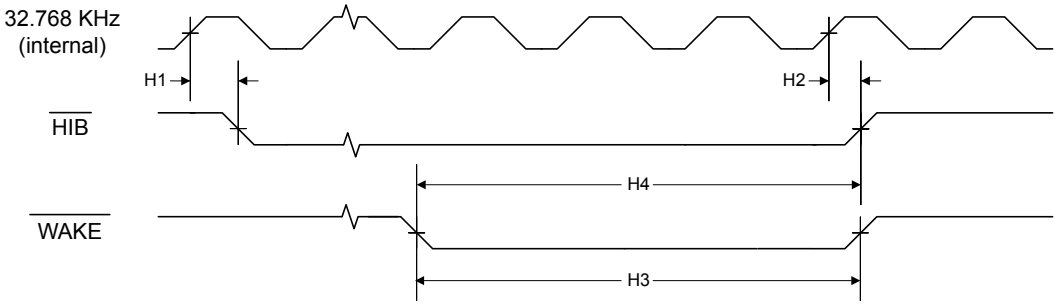
The external voltage regulators controlled by $\overline{HIB}$ must have a settling time of 250 µs or less.

Table 21-17. 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	-	200	-	µs
H2	t_{HIB_HIGH}	Internal 32.768 KHz clock reference rising edge to /HIB deasserted	-	30	-	µs
H3	t_{WAKE_ASSERT}	/WAKE assertion time	62	-	-	µs
H4	$t_{WAKETOHIB}$	/WAKE assert to /HIB desassert	62	-	124	µs
H5	t_{XOSC_SETTLE}	XOSC settling time[a]	20	-	-	ms
H6	$t_{HIB_REG_ACCESS}$	Access time to or from a non-volatile register in HIB module to complete	92	-	-	µs
H7	$t_{HIB_TO_VDD}$	$\overline{HIB}$ deassert to VDD and VDD25 at minimum operational level	-	-	250	µs

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).

Figure 21-10. Hibernation Module Timing



21.2.7 General-Purpose I/O (GPIO)

Note: All GPIOs are 5 V-tolerant.

Table 21-18. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
t_{GPIOR}	GPIO Rise Time (from 20% to 80% of V_{DD})	2-mA drive	-	17	26	ns
		4-mA drive		9	13	ns
		8-mA drive		6	9	ns
		8-mA drive with slew rate control		10	12	ns
t_{GPIOF}	GPIO Fall Time (from 80% to 20% of V_{DD})	2-mA drive	-	17	25	ns
		4-mA drive		8	12	ns
		8-mA drive		6	10	ns
		8-mA drive with slew rate control		11	13	ns

21.2.8 Analog-to-Digital Converter

Table 21-19. ADC Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{ADCIN}	Maximum single-ended, full-scale analog input voltage	-	-	3.0	V
	Minimum single-ended, full-scale analog input voltage	0.0	-	-	V
	Maximum differential, full-scale analog input voltage	-	-	1.5	V
	Minimum differential, full-scale analog input voltage	0.0	-	-	V
N	Resolution		10		bits
f_{ADC}	ADC internal clock frequency[b]	14	16	18	MHz
$t_{ADCCONV}$	Conversion time[c]		1		µs
$f_{ADCCONV}$	Conversion rate[c]		1000		k samples/s
t_{LT}	Latency from trigger to start of conversion	-	2	-	system clocks
I_L	ADC input leakage	-	-	±3.0	µA
R_{ADC}	ADC equivalent resistance	-	-	10	kΩ
C_{ADC}	ADC equivalent capacitance	0.9	1.0	1.1	pF
E_L	Integral nonlinearity error	-	-	±3	LSB
E_D	Differential nonlinearity error	-	-	±2	LSB
E_O	Offset error	-	-	+6[d]	LSB
E_G	Full-scale gain error	-	-	±3	LSB
E_{TS}	Temperature sensor accuracy	-	-	±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. The offset error listed above is the conversion result with 0 V applied to the ADC input.

Figure 21-11. ADC Input Equivalency Diagram

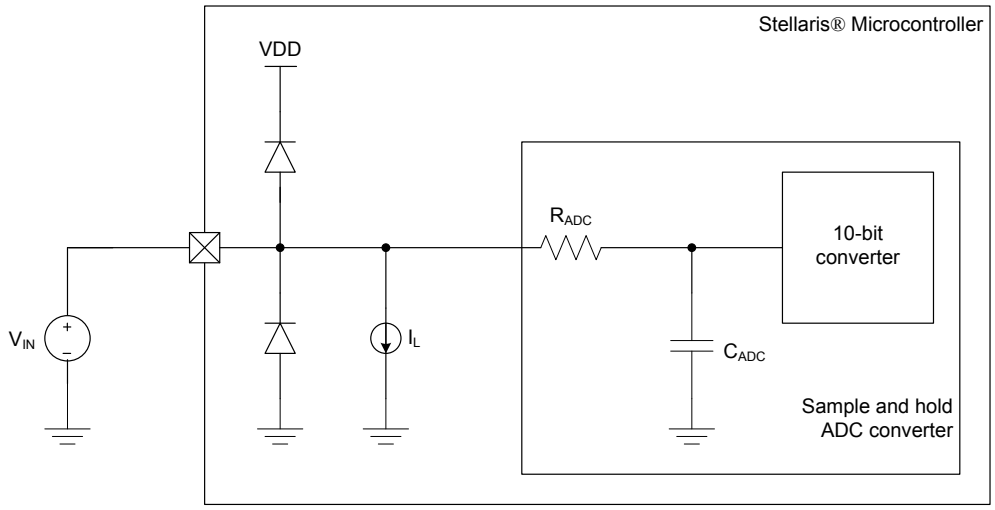


Table 21-20. ADC Module Internal Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{REFI}	Internal voltage reference for ADC	-	3.0	-	V
E_{IR}	Internal voltage reference error	-	-	±2.5	%

21.2.9 Synchronous Serial Interface (SSI)

Table 21-21. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	t_{clk_per}	SSIClk cycle time	2	-	65024	system clocks
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[a]	-	6	10	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
S9	t_{DSh}	Data from slave hold time	2	-	-	system clocks

a. Note that the delays shown are using 8-mA drive strength.

Figure 21-12. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

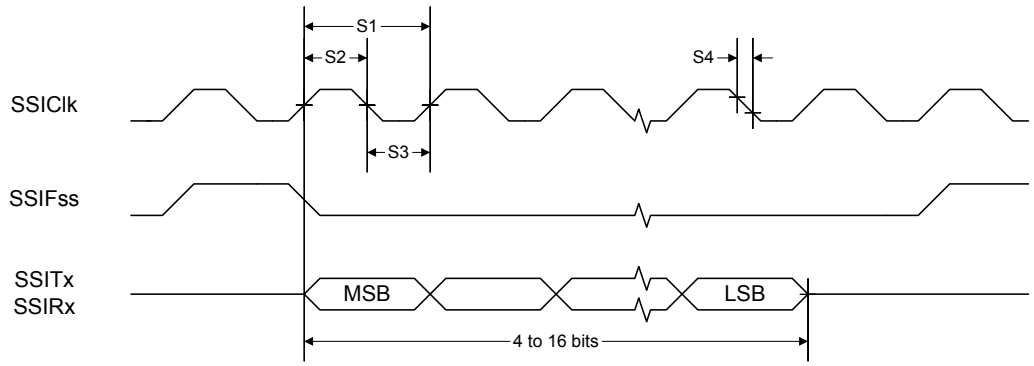


Figure 21-13. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

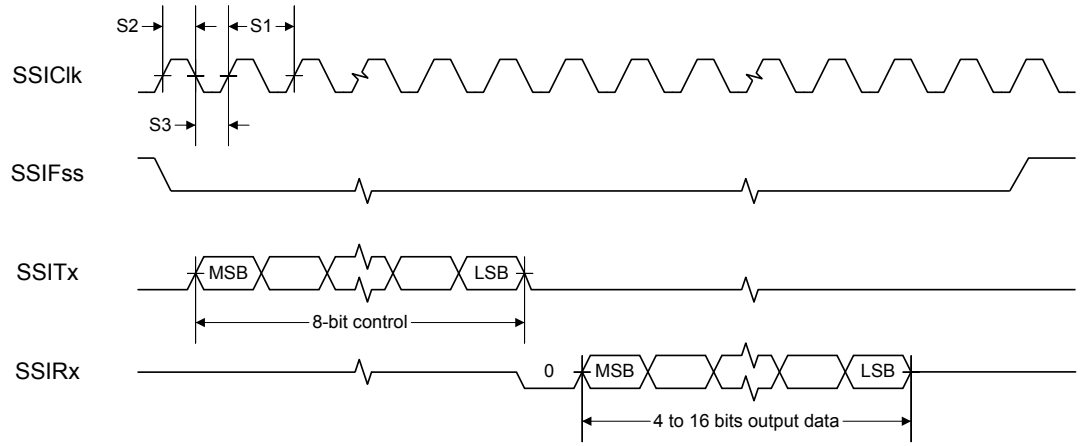
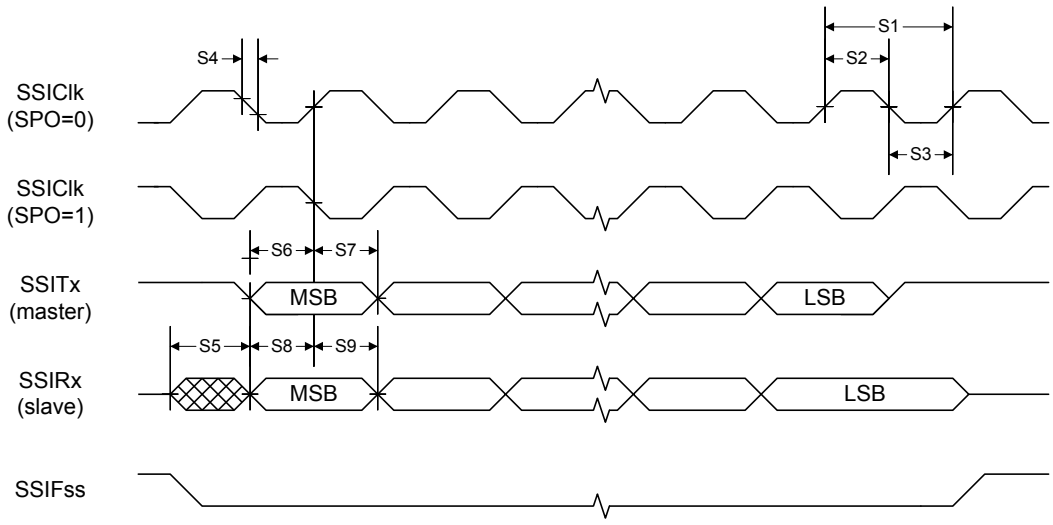


Figure 21-14. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



21.2.10 Inter-Integrated Circuit (I^2C) Interface

Table 21-22. I^2C Characteristics

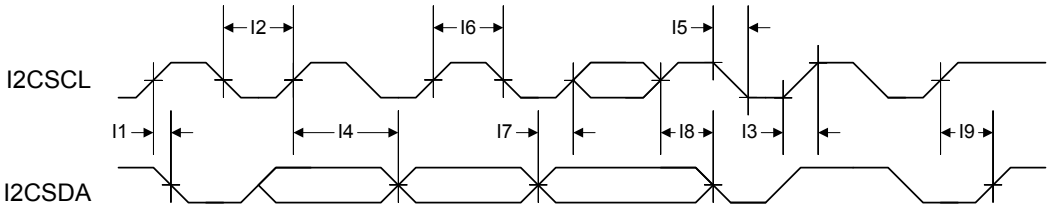
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1[a]	t_{SCH}	Start condition hold time	36	-	-	system clocks
I2[a]	t_{LP}	Clock Low period	36	-	-	system clocks
I3[b]	t_{SRT}	I2CSCL/I2CSDA rise time (V_{IL} =0.5 V to V_{IH} =2.4 V)	-	-	(see note b)	ns
I4[a]	t_{DH}	Data hold time	2	-	-	system clocks
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 21-15. I²C Timing



21.2.11 Ethernet Controller

Table 21-23. 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	-	-	-	ps
Jitter	-	-	1.4	ns

a. Measured at the line side of the transformer.

Table 21-24. 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 21-25. 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	-	20	-	kΩ
Jitter tolerance (pk-pk)	4	-	-	ns
Baseline wander tracking	-75	-	+75	%
Signal detect assertion time	-	-	1000	µs
Signal detect de-assertion time	-	-	4	µs

Table 21-26. 10BASE-T Transmitter Characteristics[a]

Parameter Name	Min	Nom	Max	Unit
Peak differential output signal	2.2	-	2.8	V
Harmonic content	27	-	-	dB
Link pulse width	-	100	-	ns

Table 21-26. 10BASE-T Transmitter Characteristics *(continued)*

Parameter Name	Min	Nom	Max	Unit
Start-of-idle pulse width	-	300 350	-	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 21-27. 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 21-28. 10BASE-T Receiver Characteristics

Parameter Name	Min	Nom	Max	Unit
Jitter tolerance (pk-pk)	30	-	-	ns
Input squelched threshold	500	600	700	mVppd
Differential input resistance	-	20	-	kΩ
Common-mode rejection	25	-	-	V

Table 21-29. 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. For the transmit line transformer with higher insertion losses, up to 1.2 dB of insertion loss can be compensated by selecting the appropriate setting in the Transmit Amplitude Selection (TX0) bits in the **MR19** register.

Table 21-30. Ethernet Reference Crystal[a]

Name	Value	Condition
Frequency	25.00000	MHz
Frequency tolerance[b]	±50	PPM

Table 21-30. Ethernet Reference Crystal *(continued)*

Name	Value	Condition
Oscillation mode	Parallel resonance, fundamental mode	
Parameters at 25° C ±2° C; Drive level = 0.5 mW		
Drive level (typ)	50-100	µW
Shunt capacitance (max)	10	pF
Motional capacitance (min)	10	fF
Series resistance (max)	60	Ω
Spurious response (max)	> 5 dB below main within 500 kHz	

a. If the internal crystal oscillator is used, select a crystal that meets these specifications.

b. This tolerance provides a guard band for temperature stability and aging drift.

Figure 21-16. External XTLP Oscillator Characteristics

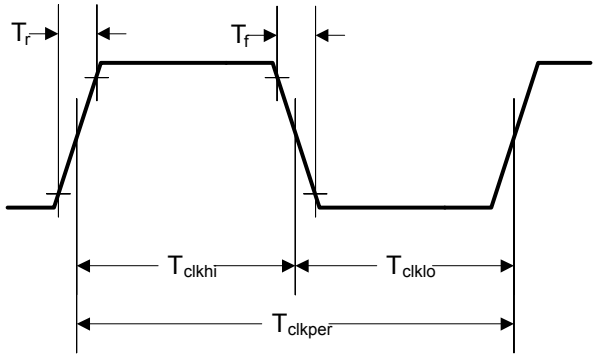


Table 21-31. External XTLP Oscillator Characteristics

Parameter Name	Symbol	Min	Nom	Max	Unit
XTLN Input Low Voltage	$XTLN_{ILV}$	-	-	0.8	-
XTLP Frequency[a]	$XTLP_f$	-	25.0	-	-
XTLP Period[b]	T_{clkper}	-	40	-	-
XTLP Duty Cycle	$XTLP_{DC}$	40 40	-	60 60	%
Rise/Fall Time	T_r , T_f	-	-	4.0	ns
Absolute Jitter	T_{JITTER}	-	-	0.1	ns

a. IEEE 802.3 frequency tolerance ±50 ppm.

b. IEEE 802.3 frequency tolerance ±50 ppm.

21.2.12 Analog Comparator

Table 21-32. Analog Comparator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{OS}	Input offset voltage	-	±10	±25	mV

Table 21-32. Analog Comparator Characteristics *(continued)*

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{CM}	Input common mode voltage range	0	-	V_{DD}-1.5	V
C_{MRR}	Common mode rejection ratio	50	-	-	dB
T_{RT}	Response time	-	-	1	µs
T_{MC}	Comparator mode change to Output Valid	-	-	10	µs

Table 21-33. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{HR}	Resolution high range	-	V_{DD}/31	-	LSB
R_{LR}	Resolution low range	-	V_{DD}/23	-	LSB
A_{HR}	Absolute accuracy high range	-	-	±1/2	LSB
A_{LR}	Absolute accuracy low range	-	-	±1/4	LSB

A Serial Flash Loader

A.1 Serial Flash Loader

The Stellaris® serial flash loader is a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface. The serial flash loader uses a simple packet interface to provide synchronous communication with the device. The flash loader runs off the crystal and does not enable the PLL, so its speed is determined by the crystal used. The two serial interfaces that can be used are the UART0 and SSI0 interfaces. For simplicity, both the data format and communication protocol are identical for both serial interfaces.

A.2 Interfaces

Once communication with the flash loader is established via one of the serial interfaces, that interface is used until the flash loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the flash loader via the UART are disabled until the device is reset.

A.2.1 UART

The Universal Asynchronous Receivers/Transmitters (UART) communication uses a fixed serial format of 8 bits of data, no parity, and 1 stop bit. The baud rate used for communication is automatically detected by the flash loader and can be any valid baud rate supported by the host and the device. The auto detection sequence requires that the baud rate should be no more than 1/32 the crystal frequency of the board that is running the serial flash loader. This is actually the same as the hardware limitation for the maximum baud rate for any UART on a Stellaris device which is calculated as follows:

```
Max Baud Rate = System Clock Frequency / 16
```

In order to determine the baud rate, the serial flash loader needs to determine the relationship between its own crystal frequency and the baud rate. This is enough information for the flash loader to configure its UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate that it wants to communicate with the device.

The method used to perform this automatic synchronization relies on the host sending the flash loader two bytes that are both 0x55. This generates a series of pulses to the flash loader that it can use to calculate the ratios needed to program the UART to match the host's baud rate. After the host sends the pattern, it attempts to read back one byte of data from the UART. The flash loader returns the value of 0xCC to indicate successful detection of the baud rate. If this byte is not received after at least twice the time required to transfer the two bytes, the host can resend another pattern of 0x55, 0x55, and wait for the 0xCC byte again until the flash loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the flash loader should be calculated as at least 2*(20(bits/sync)/baud rate (bits/sec)). For a baud rate of 115200, this time is 2*(20/115200) or 0.35 ms.

A.2.2 SSI

The Synchronous Serial Interface (SSI) port also uses a fixed serial format for communications, with the framing defined as Motorola format with SPH set to 1 and SPO set to 1. See "Frame Formats" on page 475 in the SSI chapter for more information on formats for this transfer protocol. Like the UART, this interface has hardware requirements that limit the maximum speed that the SSI clock can run. This allows the SSI clock to be at most 1/12 the crystal frequency of the board running

the flash loader. Since the host device is the master, the SSI on the flash loader device does not need to determine the clock as it is provided directly by the host.

A.3 Packet Handling

All communications, with the exception of the UART auto-baud, are done via defined packets that are acknowledged (ACK) or not acknowledged (NAK) by the devices. The packets use the same format for receiving and sending packets, including the method used to acknowledge successful or unsuccessful reception of a packet.

A.3.1 Packet Format

All packets sent and received from the device use the following byte-packed format.

```
struct
{
 unsigned char ucSize;
 unsigned char ucCheckSum;
 unsigned char Data[];
};
```

ucSize	The first byte received holds the total size of the transfer including the size and checksum bytes.
ucChecksum	This holds a simple checksum of the bytes in the data buffer only. The algorithm is Data[0]+Data[1]+…+ Data[`ucSize`-3].
Data	This is the raw data intended for the device, which is formatted in some form of command interface. There should be `ucSize`–2 bytes of data provided in this buffer to or from the device.

A.3.2 Sending Packets

The actual bytes of the packet can be sent individually or all at once; the only limitation is that commands that cause flash memory access should limit the download sizes to prevent losing bytes during flash programming. This limitation is discussed further in the section that describes the serial flash loader command, COMMAND_SEND_DATA (see “COMMAND_SEND_DATA (0x24)” on page 710).

Once the packet has been formatted correctly by the host, it should be sent out over the UART or SSI interface. Then the host should poll the UART or SSI interface for the first non-zero data returned from the device. The first non-zero byte will either be an ACK (0xCC) or a NAK (0x33) byte from the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This does not indicate that the actual contents of the command issued in the data portion of the packet were valid, just that the packet was received correctly.

A.3.3 Receiving Packets

The flash loader sends a packet of data in the same format that it receives a packet. The flash loader may transfer leading zero data before the first actual byte of data is sent out. The first non-zero byte is the size of the packet followed by a checksum byte, and finally followed by the data itself. There is no break in the data after the first non-zero byte is sent from the flash loader. Once the device communicating with the flash loader receives all the bytes, it must either ACK or NAK the packet to indicate that the transmission was successful. The appropriate response after sending a NAK to the flash loader is to resend the command that failed and request the data again. If needed, the host may send leading zeros before sending down the ACK/NAK signal to the flash loader, as the

flash loader only accepts the first non-zero data as a valid response. This zero padding is needed by the SSI interface in order to receive data to or from the flash loader.

A.4 Commands

The next section defines the list of commands that can be sent to the flash loader. The first byte of the data should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

A.4.1 COMMAND_PING (0X20)

This command simply accepts the command and sets the global status to success. The format of the packet is as follows:

```
Byte[0] = 0x03;
Byte[1] = checksum(Byte[2]);
Byte[2] = COMMAND_PING;
```

The ping command has 3 bytes and the value for COMMAND_PING is 0x20 and the checksum of one byte is that same byte, making Byte[1] also 0x20. Since the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the flash loader.

A.4.2 COMMAND_GET_STATUS (0x23)

This command returns the status of the last command that was issued. Typically, this command should be sent after every command to ensure that the previous command was successful or to properly respond to a failure. The command requires one byte in the data of the packet and should be followed by reading a packet with one byte of data that contains a status code. The last step is to ACK or NAK the received data so the flash loader knows that the data has been read.

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

A.4.3 COMMAND_DOWNLOAD (0x21)

This command is sent to the flash loader to indicate where to store data and how many bytes will be sent by the COMMAND_SEND_DATA commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command also triggers an erase of the full area to be programmed so this command takes longer than other commands. This results in a longer time to receive the ACK/NAK back from the board. This command should be followed by a COMMAND_GET_STATUS to ensure that the Program Address and Program size are valid for the device running the flash loader.

The format of the packet to send this command is a follows:

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_DOWNLOAD
Byte[3] = Program Address [31:24]
Byte[4] = Program Address [23:16]
Byte[5] = Program Address [15:8]
Byte[6] = Program Address [7:0]
Byte[7] = Program Size [31:24]
```

```
Byte[8] = Program Size [23:16]
Byte[9] = Program Size [15:8]
Byte[10] = Program Size [7:0]
```

A.4.4 COMMAND_SEND_DATA (0x24)

This command should only follow a COMMAND_DOWNLOAD command or another COMMAND_SEND_DATA command if more data is needed. Consecutive send data commands automatically increment address and continue programming from the previous location. The caller should limit transfers of data to a maximum 8 bytes of packet data to allow the flash to program successfully and not overflow input buffers of the serial interfaces. The command terminates programming once the number of bytes indicated by the COMMAND_DOWNLOAD command has been received. Each time this function is called it should be followed by a COMMAND_GET_STATUS to ensure that the data was successfully programmed into the flash. If the flash loader sends a NAK to this command, the flash loader does not increment the current address to allow retransmission of the previous data.

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_SEND_DATA
Byte[3] = Data[0]
Byte[4] = Data[1]
Byte[5] = Data[2]
Byte[6] = Data[3]
Byte[7] = Data[4]
Byte[8] = Data[5]
Byte[9] = Data[6]
Byte[10] = Data[7]
```

A.4.5 COMMAND_RUN (0x22)

This command is used to tell the flash loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first and the flash loader responds with an ACK signal back to the host device before actually executing the code at the given address. This allows the host to know that the command was received successfully and the code is now running.

```
Byte[0] = 7
Byte[1] = checksum(Bytes[2:6])
Byte[2] = COMMAND_RUN
Byte[3] = Execute Address[31:24]
Byte[4] = Execute Address[23:16]
Byte[5] = Execute Address[15:8]
Byte[6] = Execute Address[7:0]
```

A.4.6 COMMAND_RESET (0x25)

This command is used to tell the flash loader device to reset. This is useful when downloading a new image that overwrote the flash loader and wants to start from a full reset. Unlike the COMMAND_RUN command, this allows the initial stack pointer to be read by the hardware and set up for the new code. It can also be used to reset the flash loader if a critical error occurs and the host device wants to restart communication with the flash loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

The flash loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the flash loader. This allows the host to know that the command was received successfully and the part will be reset.

B 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 59)															
DATA															
DATA															
R1, type R/W, , reset - (see page 59)															
DATA															
DATA															
R2, type R/W, , reset - (see page 59)															
DATA															
DATA															
R3, type R/W, , reset - (see page 59)															
DATA															
DATA															
R4, type R/W, , reset - (see page 59)															
DATA															
DATA															
R5, type R/W, , reset - (see page 59)															
DATA															
DATA															
R6, type R/W, , reset - (see page 59)															
DATA															
DATA															
R7, type R/W, , reset - (see page 59)															
DATA															
DATA															
R8, type R/W, , reset - (see page 59)															
DATA															
DATA															
R9, type R/W, , reset - (see page 59)															
DATA															
DATA															
R10, type R/W, , reset - (see page 59)															
DATA															
DATA															
R11, type R/W, , reset - (see page 59)															
DATA															
DATA															
R12, type R/W, , reset - (see page 59)															
DATA															
DATA															
SP, type R/W, , reset - (see page 60)															
SP															
SP															
LR, type R/W, , reset 0xFFFF.FFFF (see page 61)															
LINK															
LINK															
PC, type R/W, , reset - (see page 62)															
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 63)															
N	Z	C	V	Q	ICI / IT		THUMB								
ICI / IT										ISRNUM					
PRIMASK, type R/W, , reset 0x0000.0000 (see page 67)															
															PRIMASK
FAULTMASK, type R/W, , reset 0x0000.0000 (see page 68)															
															FAULTMASK
BASEPRI, type R/W, , reset 0x0000.0000 (see page 69)															
								BASEPRI							
CONTROL, type R/W, , reset 0x0000.0000 (see page 70)															
														ASP	TMPL
Cortex-M3 Peripherals **System Timer (SysTick) Registers** Base 0xE000.E000															
STCTRL, type R/W, offset 0x010, reset 0x0000.0000															
															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 **Nested Vectored Interrupt Controller (NVIC) Registers** Base 0xE000.E000															
EN0, type R/W, offset 0x100, reset 0x0000.0000															
INT															
INT															
EN1, type R/W, offset 0x104, reset 0x0000.0000															
INT															
DIS0, type R/W, offset 0x180, reset 0x0000.0000															
INT															
INT															
DIS1, type R/W, offset 0x184, reset 0x0000.0000															
INT															
PEND0, type R/W, offset 0x200, reset 0x0000.0000															
INT															
INT															
PEND1, type R/W, offset 0x204, reset 0x0000.0000															
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											
ACTIVE0, type RO, offset 0x300, reset 0x0000.0000															
INT															
INT															
ACTIVE1, type RO, offset 0x304, reset 0x0000.0000															
				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							
SWTRIG, type WO, offset 0xF00, reset 0x0000.0000															
										INTID					

Cortex-M3 Peripherals

System Control Block (SCB) Registers

Base 0xE000.E000

CPUID, type RO, offset 0xD00, reset 0x411F.C231		
IMP	VAR	CON
PARTNO		REV

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
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 0x0000.0000															
						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															

Cortex-M3 Peripherals

Memory Protection Unit (MPU) Registers

Base 0xE000.E000

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
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		

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
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 Base 0x400F.E000															
DID0, type RO, offset 0x000, reset - (see page 186)															
	VER							CLASS							
MAJOR								MINOR							
PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD (see page 188)															
														BORIOR	
LDOPCTL, type R/W, offset 0x034, reset 0x0000.0000 (see page 189)															
										VADJ					
RIS, type RO, offset 0x050, reset 0x0000.0000 (see page 190)															
									PLLLRIS					BORRIS	
IMC, type R/W, offset 0x054, reset 0x0000.0000 (see page 191)															
									PLLLIM					BORIM	
MISC, type R/W1C, offset 0x058, reset 0x0000.0000 (see page 192)															
									PLLLMIS					BORMIS	
RESC, type R/W, offset 0x05C, reset - (see page 193)															
											SW	WDT	BOR	POR	EXT
RCC, type R/W, offset 0x060, reset 0x0780.3AD1 (see page 194)															
				ACG		SYSDIV			USESYSDIV						
		PWRDN		BYPASS		XTAL				OSCSRC				IOSCDIS	MOSCDIS
PLLCFG, type RO, offset 0x064, reset - (see page 198)															
		F									R				

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
RCC2, type R/W, offset 0x070, reset 0x0780.2810 (see page 199)															
USERCC2			SYSDIV2												
		PWRDN2		BYPASS2					OSCSRC2						
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000 (see page 201)															
			DSDIVORIDE												
									DSOSCSRC						
DID1, type RO, offset 0x004, reset - (see page 202)															
VER				FAM				PARTNO							
PINCOUNT								TEMP			PKG		ROHS	QUAL	
DC0, type RO, offset 0x008, reset 0x00FF.007F (see page 204)															
SRAMSZ															
FLASHSZ															
DC1, type RO, offset 0x010, reset 0x0101.33FF (see page 205)															
							CAN0								ADC
MINSYSDIV						MAXADCSPD		MPU	HIB	TEMPSNS	PLL	WDT	SWO	SWD	JTAG
DC2, type RO, offset 0x014, reset 0x070F.5017 (see page 207)															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0								SSI0		UART2	UART1	UART0
DC3, type RO, offset 0x018, reset 0xBFFF.7FC0 (see page 209)															
32KHZ		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS						
DC4, type RO, offset 0x01C, reset 0x5100.007F (see page 211)															
	EPHY0		EMAC0				E1588								
									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
RCGC0, type R/W, offset 0x100, reset 0x00000040 (see page 213)															
							CAN0								ADC
						MAXADCSPD			HIB			WDT			
SCGC0, type R/W, offset 0x110, reset 0x00000040 (see page 215)															
							CAN0								ADC
						MAXADCSPD			HIB			WDT			
DCGC0, type R/W, offset 0x120, reset 0x00000040 (see page 217)															
							CAN0								ADC
									HIB			WDT			
RCGC1, type R/W, offset 0x104, reset 0x00000000 (see page 219)															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0								SSI0		UART2	UART1	UART0
SCGC1, type R/W, offset 0x114, reset 0x00000000 (see page 222)															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0								SSI0		UART2	UART1	UART0
DCGC1, type R/W, offset 0x124, reset 0x00000000 (see page 225)															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0								SSI0		UART2	UART1	UART0
RCGC2, type R/W, offset 0x108, reset 0x00000000 (see page 228)															
	EPHY0		EMAC0												
									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SCGC2, type R/W, offset 0x118, reset 0x00000000 (see page 230)															
	EPHY0		EMAC0												
									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DCGC2, type R/W, offset 0x128, reset 0x00000000 (see page 232)															
	EPHY0		EMAC0												
									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
SRCR0, type R/W, offset 0x040, reset 0x00000000 (see page 234)															
							CAN0								ADC
									HIB			WDT			
SRCR1, type R/W, offset 0x044, reset 0x00000000 (see page 235)															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0								SSI0		UART2	UART1	UART0
SRCR2, type R/W, offset 0x048, reset 0x00000000 (see page 237)															
	EPHY0		EMAC0												
									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Hibernation Module Base 0x400F.C000															
HIBRTCC, type RO, offset 0x000, reset 0x0000.0000 (see page 247)															
RTCC															
RTCC															
HIBRTCM0, type R/W, offset 0x004, reset 0xFFFF.FFFF (see page 248)															
RTCM0															
RTCM0															
HIBRTCM1, type R/W, offset 0x008, reset 0xFFFF.FFFF (see page 249)															
RTCM1															
RTCM1															
HIBRTCLD, type R/W, offset 0x00C, reset 0xFFFF.FFFF (see page 250)															
RTCLD															
RTCLD															
HIBCTL, type R/W, offset 0x010, reset 0x8000.0000 (see page 251)															
								VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
HIBIM, type R/W, offset 0x014, reset 0x0000.0000 (see page 253)															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRIS, type RO, offset 0x018, reset 0x0000.0000 (see page 254)															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBMIS, type RO, offset 0x01C, reset 0x0000.0000 (see page 255)															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBIC, type R/W1C, offset 0x020, reset 0x0000.0000 (see page 256)															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRTCT, type R/W, offset 0x024, reset 0x0000.7FFF (see page 257)															
TRIM															
HIBDATA, type R/W, offset 0x030-0x12C, reset - (see page 258)															
RTD															
RTD															
Internal Memory **Flash Memory Control Registers (Flash Control Offset)** Base 0x400F.D000															
FMA, type R/W, offset 0x000, reset 0x0000.0000															
															OFFSET
OFFSET															

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
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
FCIM, type R/W, offset 0x010, reset 0x0000.0000															
														PMASK	AMASK
FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000															
														PMISC	AMISC

Internal Memory

Flash Memory Protection Registers (System Control Offset)

Base 0x400F.E000

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
USECRL, type R/W, offset 0x140, reset 0x31															
								USEC							
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															
USER_DBG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE															
NW	DATA														
DATA														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															
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															
FMPRE3, type R/W, offset 0x20C, reset 0xFFFF.FFFF															
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															

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
FMPPE3, type R/W, offset 0x40C, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
General-Purpose Input/Outputs (GPIOs) GPIO Port A base: 0x4000.4000 GPIO Port B base: 0x4000.5000 GPIO Port C base: 0x4000.6000 GPIO Port D base: 0x4000.7000 GPIO Port E base: 0x4002.4000 GPIO Port F base: 0x4002.5000 GPIO Port G base: 0x4002.6000															
GPIODATA, type R/W, offset 0x000, reset 0x0000.0000 (see page 299)															
DATA															
GPIODIR, type R/W, offset 0x400, reset 0x0000.0000 (see page 300)															
DIR															
GPIOIS, type R/W, offset 0x404, reset 0x0000.0000 (see page 301)															
IS															
GPIOIBE, type R/W, offset 0x408, reset 0x0000.0000 (see page 302)															
IBE															
GPIOIEV, type R/W, offset 0x40C, reset 0x0000.0000 (see page 303)															
IEV															
GPIOIM, type R/W, offset 0x410, reset 0x0000.0000 (see page 304)															
IME															
GPIORIS, type RO, offset 0x414, reset 0x0000.0000 (see page 305)															
RIS															
GPIOMIS, type RO, offset 0x418, reset 0x0000.0000 (see page 306)															
MIS															
GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000 (see page 307)															
IC															
GPIOAFSEL, type R/W, offset 0x420, reset - (see page 308)															
AFSEL															
GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF (see page 310)															
DRV2															
GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000 (see page 311)															
DRV4															
GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000 (see page 312)															
DRV8															
GPIOODR, type R/W, offset 0x50C, reset 0x0000.0000 (see page 313)															
ODE															

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
GPIOPUR, type R/W, offset 0x510, reset - (see page 314)															
								PUE							
GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000 (see page 315)															
								PDE							
GPIOSLR, type R/W, offset 0x518, reset 0x0000.0000 (see page 316)															
								SRL							
GPIODEN, type R/W, offset 0x51C, reset - (see page 317)															
								DEN							
GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001 (see page 318)															
LOCK															
LOCK															
GPIOCR, type -, offset 0x524, reset - (see page 319)															
								CR							
GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 321)															
								PID4							
GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 322)															
								PID5							
GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 323)															
								PID6							
GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 324)															
								PID7							
GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061 (see page 325)															
								PID0							
GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 (see page 326)															
								PID1							
GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 327)															
								PID2							
GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 328)															
								PID3							
GPIOPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 329)															
								CID0							
GPIOPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 330)															
								CID1							
GPIOPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 331)															
								CID2							

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
GPIOPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 332)															
								CID3							
General-Purpose Timers Timer0 base: 0x4003.0000 Timer1 base: 0x4003.1000 Timer2 base: 0x4003.2000 Timer3 base: 0x4003.3000															
GPTMCFG, type R/W, offset 0x000, reset 0x0000.0000 (see page 346)															
													GPTMCFG		
GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000 (see page 347)															
												TAAMS	TACMR	TAMR	
GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000 (see page 349)															
												TBAMS	TBCMR	TBMR	
GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000 (see page 351)															
	TBPWML	TBOTE		TBEVENT		TBSTALL	TBEN		TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000 (see page 354)															
					CBEIM	CBMIM	TBTOIM					RTCIM	CAEIM	CAMIM	TATOIM
GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000 (see page 356)															
					CBERIS	CBMRIS	TBTORIS					RTCRIS	CAERIS	CAMRIS	TATORIS
GPTMMIS, type RO, offset 0x020, reset 0x0000.0000 (see page 357)															
					CBEMIS	CBMMIS	TBTOMIS					RTCMIS	CAEMIS	CAMMIS	TATOMIS
GPTMICR, type W1C, offset 0x024, reset 0x0000.0000 (see page 358)															
					CBECINT	CBMCINT	TBTOCINT					RTCCINT	CAECINT	CAMCINT	TATOCINT
GPTMTAILR, type R/W, offset 0x028, reset 0xFFFF.FFFF (see page 360)															
								TAILRH							
								TAILRL							
GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF (see page 361)															
								TBILRL							
GPTMTAMATCHR, type R/W, offset 0x030, reset 0xFFFF.FFFF (see page 362)															
								TAMRH							
								TAMRL							
GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF (see page 363)															
								TBMRL							
GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000 (see page 364)															
								TAPSR							
GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000 (see page 365)															
								TBPSR							
GPTMTAPMR, type R/W, offset 0x040, reset 0x0000.0000 (see page 366)															
								TAPSMR							

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
GPTMTBPMR, type R/W, offset 0x044, reset 0x0000.0000 (see page 367)															
								TBPSMR							
GPTMTAR, type RO, offset 0x048, reset 0xFFFF.FFFF (see page 368)															
TARH															
TARL															
GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF (see page 369)															
TBRL															

Watchdog Timer

Base 0x4000.0000

WDTLOAD, type R/W, offset 0x000, reset 0xFFFF.FFFF (see page 374)															
WDTLoad															
WDTLoad															
WDTVALUE, type RO, offset 0x004, reset 0xFFFF.FFFF (see page 375)															
WDTValue															
WDTValue															
WDTCTL, type R/W, offset 0x008, reset 0x0000.0000 (see page 376)															
														RESEN	INTEN
WDTICR, type WO, offset 0x00C, reset - (see page 377)															
WDTIntClr															
WDTIntClr															
WDTRIS, type RO, offset 0x010, reset 0x0000.0000 (see page 378)															
															WDTRIS
WDTMIS, type RO, offset 0x014, reset 0x0000.0000 (see page 379)															
															WDTMIS
WDTTEST, type R/W, offset 0x418, reset 0x0000.0000 (see page 380)															
							STALL								
WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000 (see page 381)															
WDTLock															
WDTLock															
WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 382)															
								PID4							
WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 383)															
								PID5							
WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 384)															
								PID6							
WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 385)															
								PID7							
WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005 (see page 386)															
								PID0							
WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018 (see page 387)															
								PID1							

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
WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 388)															
								PID2							
WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 389)															
								PID3							
WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 390)															
								CID0							
WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 391)															
								CID1							
WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 392)															
								CID2							
WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 393)															
								CID3							
Analog-to-Digital Converter (ADC) Base 0x4003.8000															
ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000 (see page 404)															
												ASEN3	ASEN2	ASEN1	ASEN0
ADCRIS, type RO, offset 0x004, reset 0x0000.0000 (see page 405)															
												INR3	INR2	INR1	INR0
ADCIM, type R/W, offset 0x008, reset 0x0000.0000 (see page 406)															
												MASK3	MASK2	MASK1	MASK0
ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000 (see page 407)															
												IN3	IN2	IN1	IN0
ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000 (see page 408)															
												OV3	OV2	OV1	OV0
ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000 (see page 409)															
EM3				EM2				EM1				EM0			
ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000 (see page 412)															
												UV3	UV2	UV1	UV0
ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210 (see page 413)															
		SS3				SS2				SS1				SS0	
ADCPSSI, type WO, offset 0x028, reset - (see page 415)															
												SS3	SS2	SS1	SS0
ADCSAC, type R/W, offset 0x030, reset 0x0000.0000 (see page 416)															
														AVG	
ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000 (see page 417)															
	MUX7				MUX6				MUX5				MUX4		
	MUX3				MUX2				MUX1				MUX0		

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
ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000 (see page 419)															
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 422)															
						DATA									
ADCSSFIFO1, type RO, offset 0x068, reset - (see page 422)															
						DATA									
ADCSSFIFO2, type RO, offset 0x088, reset - (see page 422)															
						DATA									
ADCSSFIFO3, type RO, offset 0x0A8, reset - (see page 422)															
						DATA									
ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100 (see page 423)															
			FULL				EMPTY			HPTR				TPTR	
ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100 (see page 423)															
			FULL				EMPTY			HPTR				TPTR	
ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100 (see page 423)															
			FULL				EMPTY			HPTR				TPTR	
ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100 (see page 423)															
			FULL				EMPTY			HPTR				TPTR	
ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000 (see page 424)															
	MUX3				MUX2				MUX1				MUX0		
ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000 (see page 424)															
	MUX3				MUX2				MUX1				MUX0		
ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000 (see page 425)															
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 425)															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSMUX3, type R/W, offset 0x0A0, reset 0x0000.0000 (see page 427)															
														MUX0	
ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002 (see page 428)															
												TS0	IE0	END0	D0
ADCTMLB, type R/W, offset 0x100, reset 0x0000.0000 (see page 429)															
															LB

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
Universal Asynchronous Receivers/Transmitters (UARTs) UART0 base: 0x4000.C000 UART1 base: 0x4000.D000 UART2 base: 0x4000.E000															
UARTDR, type R/W, offset 0x000, reset 0x0000.0000 (see page 440)															
				OE	BE	PE	FE	DATA							
UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000 (Reads) (see page 442)															
												OE	BE	PE	FE
UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Writes) (see page 442)															
								DATA							
UARTFR, type RO, offset 0x018, reset 0x0000.0090 (see page 444)															
								TXFE	RXFF	TXFF	RXFE	BUSY			
UARTILPR, type R/W, offset 0x020, reset 0x0000.0000 (see page 446)															
								ILPDVSR							
UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000 (see page 447)															
DIVINT															
UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000 (see page 448)															
										DIVFRAC					
UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000 (see page 449)															
								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
UARTCTL, type R/W, offset 0x030, reset 0x0000.0300 (see page 451)															
						RXE	TXE	LBE					SIRLP	SIREN	UARTEN
UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012 (see page 453)															
										RXIFLSEL			TXIFLSEL		
UARTIM, type R/W, offset 0x038, reset 0x0000.0000 (see page 455)															
					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM				
UARTRIS, type RO, offset 0x03C, reset 0x0000.0000 (see page 457)															
					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS				
UARTMIS, type RO, offset 0x040, reset 0x0000.0000 (see page 458)															
					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS				
UARTICR, type W1C, offset 0x044, reset 0x0000.0000 (see page 459)															
					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC				
UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 461)															
								PID4							
UARTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 462)															
								PID5							

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
UARTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 463)															
								PID6							
UARTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 464)															
								PID7							
UARTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0011 (see page 465)															
								PID0							
UARTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 (see page 466)															
								PID1							
UARTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 467)															
								PID2							
UARTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 468)															
								PID3							
UARTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 469)															
								CID0							
UARTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 470)															
								CID1							
UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 471)															
								CID2							
UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 472)															
								CID3							
Synchronous Serial Interface (SSI) SSI0 base: 0x4000.8000															
SSICR0, type R/W, offset 0x000, reset 0x0000.0000 (see page 486)															
SCR								SPH	SPO	FRF		DSS			
SSICR1, type R/W, offset 0x004, reset 0x0000.0000 (see page 488)															
												SOD	MS	SSE	LBM
SSIDR, type R/W, offset 0x008, reset 0x0000.0000 (see page 490)															
DATA															
SSISR, type RO, offset 0x00C, reset 0x0000.0003 (see page 491)															
											BSY	RFF	RNE	TNF	TFE
SSICPSR, type R/W, offset 0x010, reset 0x0000.0000 (see page 493)															
								CPSDVSR							
SSIIM, type R/W, offset 0x014, reset 0x0000.0000 (see page 494)															
												TXIM	RXIM	RTIM	RORIM
SSIRIS, type RO, offset 0x018, reset 0x0000.0008 (see page 496)															
												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 497)															
												TXMIS	RXMIS	RTMIS	RORMIS
SSIICR, type W1C, offset 0x020, reset 0x0000.0000 (see page 498)															
														RTIC	RORIC
SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 499)															
								PID4							
SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 (see page 500)															
								PID5							
SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 (see page 501)															
								PID6							
SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 502)															
								PID7							
SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022 (see page 503)															
								PID0							
SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 (see page 504)															
								PID1							
SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 505)															
								PID2							
SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 506)															
								PID3							
SSIPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 507)															
								CID0							
SSIPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 508)															
								CID1							
SSIPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 509)															
								CID2							
SSIPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 510)															
								CID3							
Inter-Integrated Circuit (I^2C) Interface **I^2C Master** I2C 0 base: 0x4002.0000 I2C 1 base: 0x4002.1000															
I2CMSA, type R/W, offset 0x000, reset 0x0000.0000															
								SA							R/S
I2CMCS, type RO, offset 0x004, reset 0x0000.0000 (Reads)															
									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY

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 WO, offset 0x004, reset 0x0000.0000 (Writes)															
												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 (I²C) Interface **I²C Slave** I2C 0 base: 0x4002.0000 I2C 1 base: 0x4002.1000															
I2CSOAR, type R/W, offset 0x800, reset 0x0000.0000															
									OAR						
I2CSCSR, type RO, offset 0x804, reset 0x0000.0000 (Reads)															
													FBR	TREQ	RREQ
I2CSCSR, type WO, offset 0x804, reset 0x0000.0000 (Writes)															
															DA
I2CSDR, type R/W, offset 0x808, reset 0x0000.0000															
								DATA							
I2CSIMR, type R/W, offset 0x80C, reset 0x0000.0000															
															DATAIM
I2CSRIS, type RO, offset 0x810, reset 0x0000.0000															
															DATARIS
I2CSMIS, type RO, offset 0x814, reset 0x0000.0000															
															DATAMIS
I2CSICR, type WO, offset 0x818, reset 0x0000.0000															
															DATAIC

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
Controller Area Network (CAN) Module CAN0 base: 0x4004.0000															
CANCTL, type R/W, offset 0x000, reset 0x0000.0001 (see page 570)															
								TEST	CCE	DAR		EIE	SIE	IE	INIT
CANSTS, type R/W, offset 0x004, reset 0x0000.0000 (see page 572)															
								BOFF	EWARN	EPASS	RXOK	TXOK	LEC		
CANERR, type RO, offset 0x008, reset 0x0000.0000 (see page 574)															
RP	REC							TEC							
CANBIT, type R/W, offset 0x00C, reset 0x0000.2301 (see page 575)															
	TSEG2			TSEG1				SJW		BRP					
CANINT, type RO, offset 0x010, reset 0x0000.0000 (see page 576)															
INTID															
CANTST, type R/W, offset 0x014, reset 0x0000.0000 (see page 577)															
								RX	TX		LBACK	SILENT	BASIC		
CANBRPE, type R/W, offset 0x018, reset 0x0000.0000 (see page 579)															
												BRPE			
CANIF1CRQ, type R/W, offset 0x020, reset 0x0000.0001 (see page 580)															
BUSY										MNUM					
CANIF2CRQ, type R/W, offset 0x080, reset 0x0000.0001 (see page 580)															
BUSY										MNUM					
CANIF1CMSK, type R/W, offset 0x024, reset 0x0000.0000 (see page 581)															
								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
CANIF2CMSK, type R/W, offset 0x084, reset 0x0000.0000 (see page 581)															
								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
CANIF1MSK1, type R/W, offset 0x028, reset 0x0000.FFFF (see page 583)															
MSK															
CANIF2MSK1, type R/W, offset 0x088, reset 0x0000.FFFF (see page 583)															
MSK															
CANIF1MSK2, type R/W, offset 0x02C, reset 0x0000.FFFF (see page 584)															
MXTD	MDIR		MSK												

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
CANIF2MSK2, type R/W, offset 0x08C, reset 0x0000.FFFF (see page 584)															
MXTD	MDIR		MSK												
CANIF1ARB1, type R/W, offset 0x030, reset 0x0000.0000 (see page 585)															
ID															
CANIF2ARB1, type R/W, offset 0x090, reset 0x0000.0000 (see page 585)															
ID															
CANIF1ARB2, type R/W, offset 0x034, reset 0x0000.0000 (see page 586)															
MSGVAL	XTD	DIR	ID												
CANIF2ARB2, type R/W, offset 0x094, reset 0x0000.0000 (see page 586)															
MSGVAL	XTD	DIR	ID												
CANIF1MCTL, type R/W, offset 0x038, reset 0x0000.0000 (see page 588)															
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB				DLC			
CANIF2MCTL, type R/W, offset 0x098, reset 0x0000.0000 (see page 588)															
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB				DLC			
CANIF1DA1, type R/W, offset 0x03C, reset 0x0000.0000 (see page 590)															
DATA															
CANIF1DA2, type R/W, offset 0x040, reset 0x0000.0000 (see page 590)															
DATA															
CANIF1DB1, type R/W, offset 0x044, reset 0x0000.0000 (see page 590)															
DATA															
CANIF1DB2, type R/W, offset 0x048, reset 0x0000.0000 (see page 590)															
DATA															
CANIF2DA1, type R/W, offset 0x09C, reset 0x0000.0000 (see page 590)															
DATA															
CANIF2DA2, type R/W, offset 0x0A0, reset 0x0000.0000 (see page 590)															
DATA															
CANIF2DB1, type R/W, offset 0x0A4, reset 0x0000.0000 (see page 590)															
DATA															
CANIF2DB2, type R/W, offset 0x0A8, reset 0x0000.0000 (see page 590)															
DATA															
CANTXRQ1, type RO, offset 0x100, reset 0x0000.0000 (see page 591)															
TXRQST															
CANTXRQ2, type RO, offset 0x104, reset 0x0000.0000 (see page 591)															
TXRQST															

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
CANNWDA1, type RO, offset 0x120, reset 0x0000.0000 (see page 592)															
NEWDAT															
CANNWDA2, type RO, offset 0x124, reset 0x0000.0000 (see page 592)															
NEWDAT															
CANMSG1INT, type RO, offset 0x140, reset 0x0000.0000 (see page 593)															
INTPND															
CANMSG2INT, type RO, offset 0x144, reset 0x0000.0000 (see page 593)															
INTPND															
CANMSG1VAL, type RO, offset 0x160, reset 0x0000.0000 (see page 594)															
MSGVAL															
CANMSG2VAL, type RO, offset 0x164, reset 0x0000.0000 (see page 594)															
MSGVAL															
Ethernet Controller **Ethernet MAC** Base 0x4004.8000															
MACRIS/MACIACK, type RO, offset 0x000, reset 0x0000.0000 (Reads)															
									PHYINT	MDINT	RXER	FOV	TXEMP	TXER	RXINT
MACRIS/MACIACK, type WO, offset 0x000, reset 0x0000.0000 (Writes)															
									PHYINT	MDINT	RXER	FOV	TXEMP	TXER	RXINT
MACIM, type R/W, offset 0x004, reset 0x0000.007F															
									PHYINTM	MDINTM	RXERM	FOVM	TXEMPM	TXERM	RXINTM
MACRCTL, type R/W, offset 0x008, reset 0x0000.0008															
											RSTFIFO	BADCRC	PRMS	AMUL	RXEN
MACTCTL, type R/W, offset 0x00C, reset 0x0000.0000															
											DUPLEX		CRC	PADEN	TXEN
MACDATA, type RO, offset 0x010, reset 0x0000.0000 (Reads)															
RXDATA															
RXDATA															
MACDATA, type WO, offset 0x010, reset 0x0000.0000 (Writes)															
TXDATA															
TXDATA															
MACIA0, type R/W, offset 0x014, reset 0x0000.0000															
MACOCT4								MACOCT3							
MACOCT2								MACOCT1							
MACIA1, type R/W, offset 0x018, reset 0x0000.0000															
MACOCT6								MACOCT5							
MACTHR, type R/W, offset 0x01C, reset 0x0000.003F															
										THRESH					

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
MACMCTL, type R/W, offset 0x020, reset 0x0000.0000															
								REGADR						WRITE	START
MACMDV, type R/W, offset 0x024, reset 0x0000.0080															
								DIV							
MACMTXD, type R/W, offset 0x02C, reset 0x0000.0000															
MDTX															
MACMRXD, type R/W, offset 0x030, reset 0x0000.0000															
MDRX															
MACNP, type RO, offset 0x034, reset 0x0000.0000															
										NPR					
MACTR, type R/W, offset 0x038, reset 0x0000.0000															
															NEWTX
MACTS, type R/W, offset 0x03C, reset 0x0000.0000															
															TSEN

Ethernet Controller
MII Management

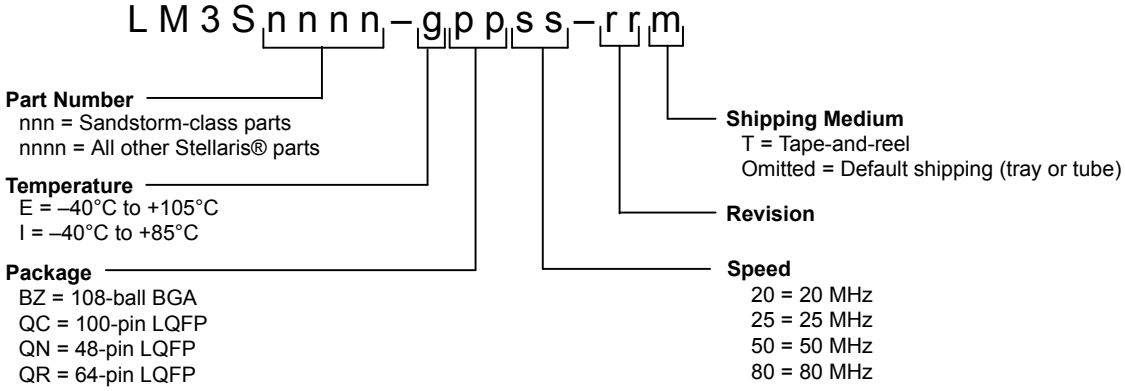
MR0, type R/W, address 0x00, reset 0x3100															
RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX	COLT							
MR1, type RO, address 0x01, reset 0x7849															
	100X_F	100X_H	10T_F	10T_H					MFPS	ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD
MR2, type RO, address 0x02, reset 0x000E															
OUI[21:6]															
MR3, type RO, address 0x03, reset 0x7237															
OUI[5:0]						MN						RN			
MR4, type R/W, address 0x04, reset 0x01E1															
NP		RF					A3	A2	A1	A0	S				
MR5, type RO, address 0x05, reset 0x0000															
NP	ACK	RF	A[7:0]								S				
MR6, type RO, address 0x06, reset 0x0000															
											PDF	LPNPA		PRX	LPANEGA
MR16, type R/W, address 0x10, reset 0x0140															
RPTR	INPOL		TXHIM	SQEI	NL10					APOL	RVSPOL			PCSBP	RXCC
MR17, type R/W, address 0x11, reset 0x0000															
JABBER_IE	RXER_IE	PRX_IE	PDF_IE	LPACK_IE	LSCHG_IE	RFAULT_IE	ANEGCOMP_IE	JABBER_INT	RXER_INT	PRX_INT	PDF_INT	LPACK_INT	LSCHG_INT	RFAULT_INT	ANEGCOMP_INT
MR18, type RO, address 0x12, reset 0x0000															
			ANEGF	DPLX	RATE	RXSD	RX_LOCK								
MR19, type R/W, address 0x13, reset 0x4000															
TXO															
MR23, type R/W, address 0x17, reset 0x0010															
								LED1[3:0]				LED0[3:0]			
MR24, type R/W, address 0x18, reset 0x00C0															
								PD_MODE	AUTO_SW	MDIX	MDIX_CM	MDIX_SD			

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
Analog Comparators Base 0x4003.C000															
ACMIS, type R/W1C, offset 0x000, reset 0x0000.0000 (see page 650)															
													IN2	IN1	IN0
ACRIS, type RO, offset 0x004, reset 0x0000.0000 (see page 651)															
													IN2	IN1	IN0
ACINTEN, type R/W, offset 0x008, reset 0x0000.0000 (see page 652)															
													IN2	IN1	IN0
ACREFCTL, type R/W, offset 0x010, reset 0x0000.0000 (see page 653)															
						EN	RNG					VREF			
ACSTAT0, type RO, offset 0x020, reset 0x0000.0000 (see page 654)															
														OVAL	
ACSTAT1, type RO, offset 0x040, reset 0x0000.0000 (see page 654)															
														OVAL	
ACSTAT2, type RO, offset 0x060, reset 0x0000.0000 (see page 654)															
														OVAL	
ACCTL0, type R/W, offset 0x024, reset 0x0000.0000 (see page 655)															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	
ACCTL1, type R/W, offset 0x044, reset 0x0000.0000 (see page 655)															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	
ACCTL2, type R/W, offset 0x064, reset 0x0000.0000 (see page 655)															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	

C Ordering and Contact Information

C.1 Ordering Information

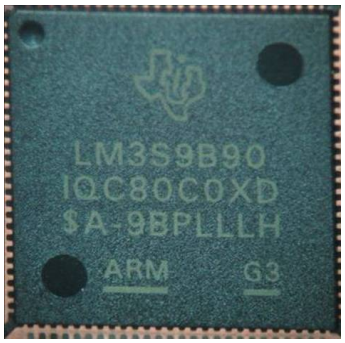
The figure below defines the full set of potential orderable part numbers for all the Stellaris® LM3S microcontrollers. See the Package Option Addendum for the valid orderable part numbers for the LM3S8938 microcontroller.



C.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, for example, LM3S9B90.
- In the second line, the first eight characters indicate the temperature, package, speed, revision, and product status. For example in the figure below, IQC80C0X indicates an Industrial temperature (I), 100-pin LQFP package (QC), 80-MHz (80), revision C0 (C0) device. The letter immediately following the revision indicates product status. An X indicates experimental and requires a waiver; an S indicates the part is fully qualified and released to production.
- The remaining characters contain internal tracking numbers.



C.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.

C.4 Support Information

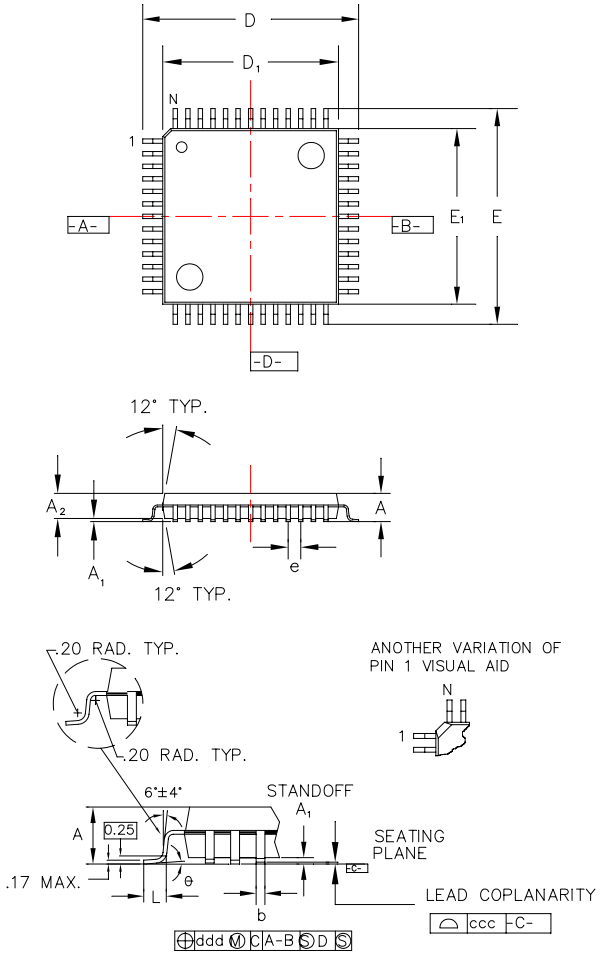
For support on Stellaris products, contact the TI Worldwide Product Information Center nearest you: http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm.

D Package Information

D.1 100-Pin LQFP Package

D.1.1 Package Dimensions

Figure D-1. Stellaris LM3S8938 100-Pin LQFP Package Dimensions



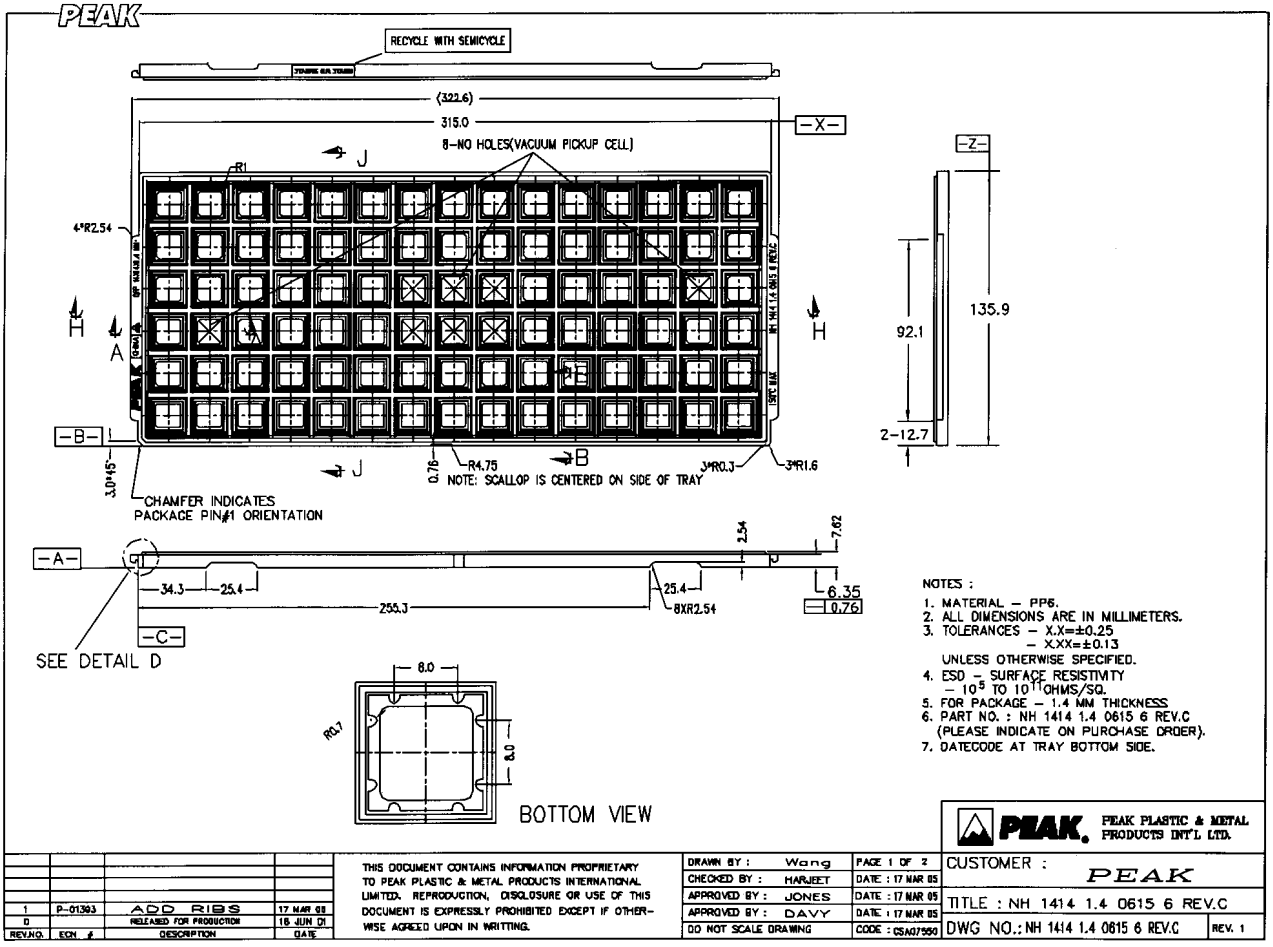
Note: The following notes apply to the package drawing.

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

D.1.2 Tray Dimensions

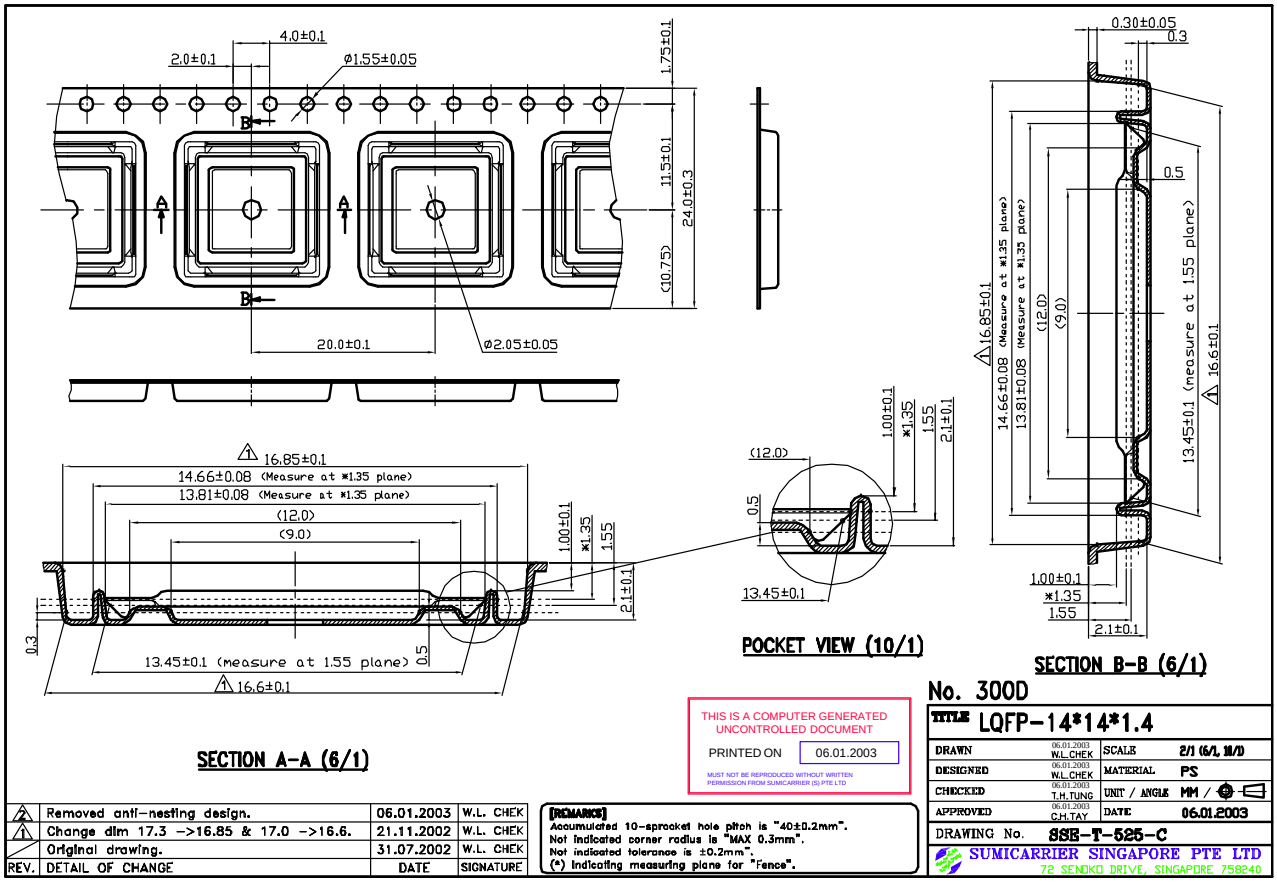
Figure D-2. 100-Pin LQFP Tray Dimensions



D.1.3 Tape and Reel Dimensions

Note: In the figure that follows, pin 1 is located in the top right corner of the device.

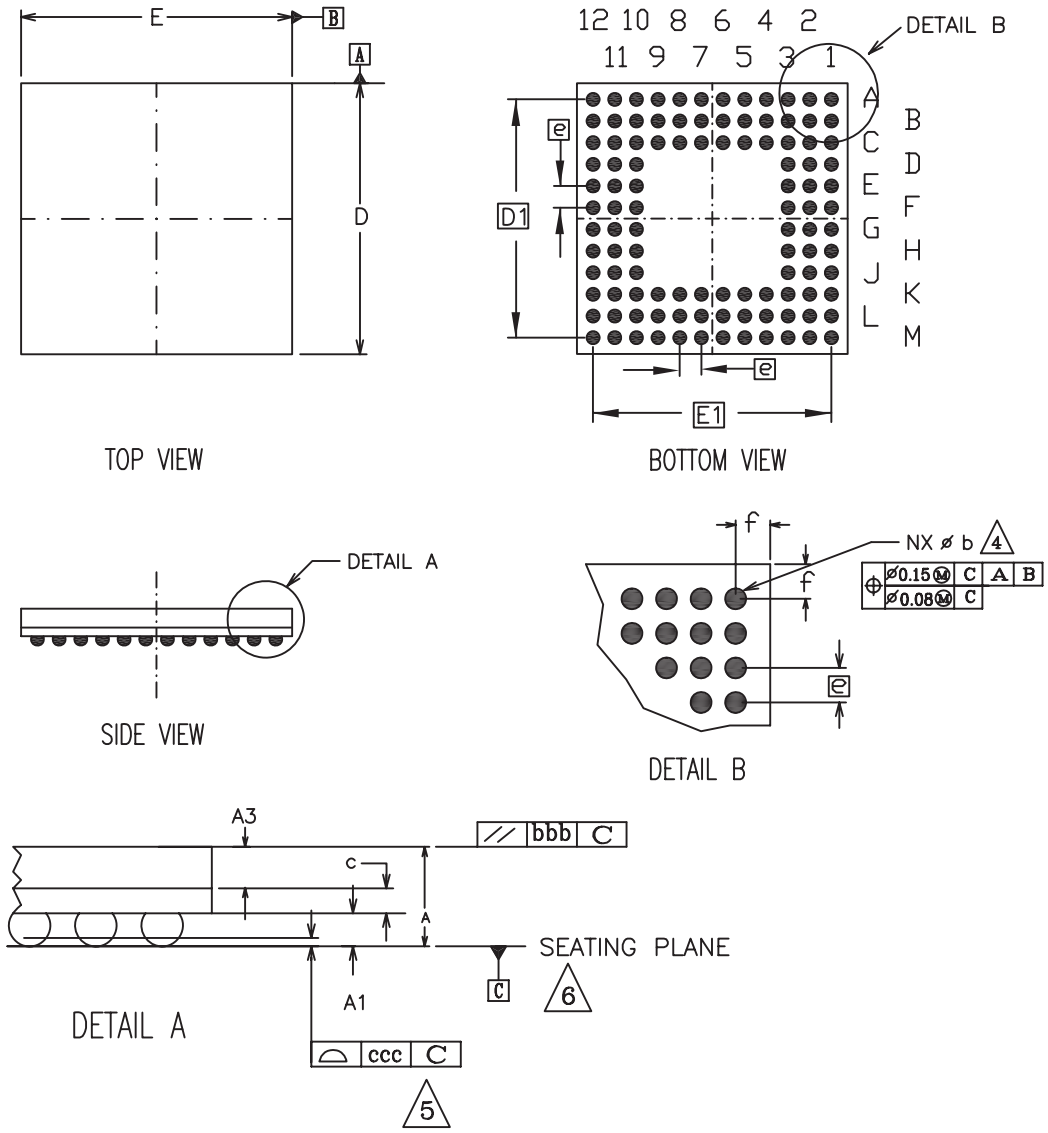
Figure D-3. 100-Pin LQFP Tape and Reel Dimensions



D.2 108-Ball BGA Package

D.2.1 Package Dimensions

Figure D-4. Stellaris LM3S8938 108-Ball BGA Package Dimensions



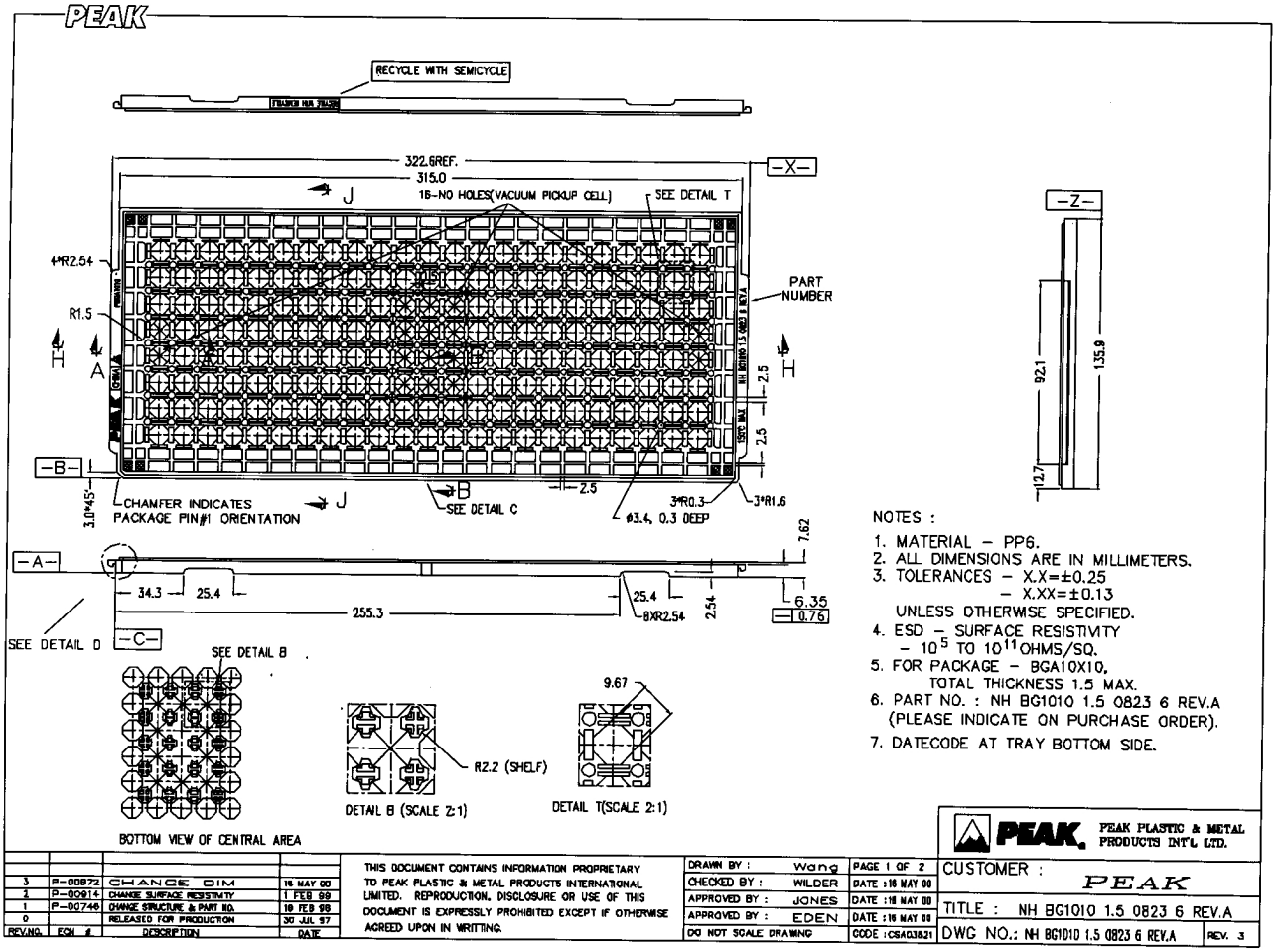
Note: The following notes apply to the package drawing.

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			

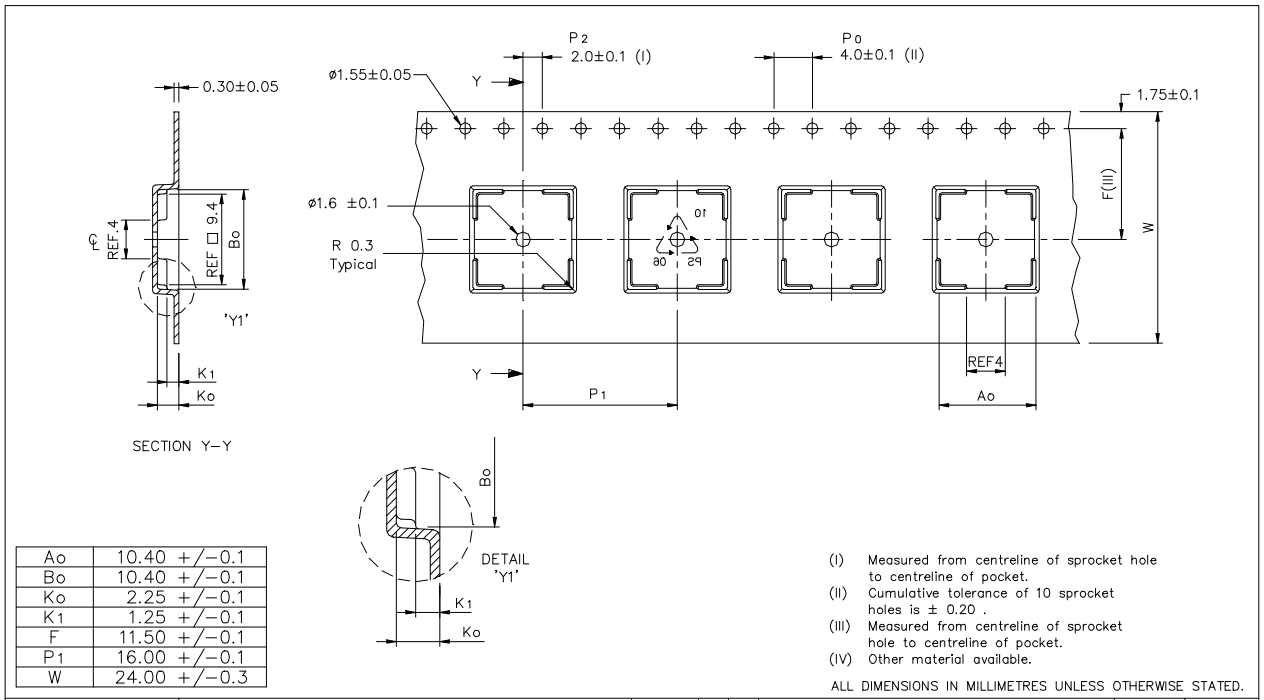
D.2.2 Tray Dimensions

Figure D-5. 108-Ball BGA Tray Dimensions



D.2.3 Tape and Reel Dimensions

Figure D-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 part number	Status (1)	Material type (2)	Package | Pins	Package qty | Carrier	RoHS (3)	Lead finish/ Ball material (4)	MSL rating/ Peak reflow (5)	Op temp (°C)	Part marking (6)
LM3S8938-EQC50-A2	NRND	Production	LQFP (PZ) | 100	90 | null	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 105	LM3S8938 EQC50
LM3S8938-EQC50-A2.A	NRND	Production	LQFP (PZ) | 100	90 | null	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 105	LM3S8938 EQC50
LM3S8938-EQC50-A2.B	NRND	Production	LQFP (PZ) | 100	90 | null	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 105	LM3S8938 EQC50
LM3S8938-IQC50-A2	NRND	Production	LQFP (PZ) | 100	90 | null	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 85	LM3S8938 IQC50
LM3S8938-IQC50-A2.A	NRND	Production	LQFP (PZ) | 100	90 | null	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 85	LM3S8938 IQC50
LM3S8938-IQC50-A2.B	NRND	Production	LQFP (PZ) | 100	90 | null	Yes	NIPDAU	Level-3-260C-168 HR	-40 to 85	LM3S8938 IQC50

(1) **Status:** For more details on status, see our product life cycle.

(2) **Material type:** When designated, preproduction parts are prototypes/experimental devices, and are not yet approved or released for full production. Testing and final process, including without limitation quality assurance, reliability performance testing, and/or process qualification, may not yet be complete, and this item is subject to further changes or possible discontinuation. If available for ordering, purchases will be subject to an additional waiver at checkout, and are intended for early internal evaluation purposes only. These items are sold without warranties of any kind.

(3) **RoHS values:** Yes, No, RoHS Exempt. See the TI RoHS Statement for additional information and value definition.

(4) **Lead finish/Ball material:** Parts may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead finish/Ball material values may wrap to two lines if the finish value exceeds the maximum column width.

(5) **MSL rating/Peak reflow:** The moisture sensitivity level ratings and peak solder (reflow) temperatures. In the event that a part has multiple moisture sensitivity ratings, only the lowest level per JEDEC standards is shown. Refer to the shipping label for the actual reflow temperature that will be used to mount the part to the printed circuit board.

(6) **Part marking:** There may be an additional marking, which relates to the logo, the lot trace code information, or the environmental category of the part.

Multiple part markings will be inside parentheses. Only one part marking contained in parentheses and separated by a "~" will appear on a part. If a line is indented then it is a continuation of the previous line and the two combined represent the entire part marking for that device.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2025, Texas Instruments Incorporated