




Stellaris® LM3S9B90 Microcontroller

DATA SHEET

Copyright

Copyright © 2007-2009 Texas Instruments Inc. All rights reserved. Stellaris and StellarisWare are registered trademarks of Texas Instruments. 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.

ADVANCE INFORMATION concerns new products in the sampling or preproduction phase of development. Characteristic data and other specifications are subject to change without notice.

 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
108 Wild Basin, Suite 350
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
<http://www.luminarymicro.com>



**TEXAS
INSTRUMENTS**



Cortex
Intelligent Processors by ARM

Table of Contents

| | |
|---|-----------|
| Revision History | 35 |
| About This Document | 37 |
| Audience | 37 |
| About This Manual | 37 |
| Related Documents | 37 |
| Documentation Conventions | 37 |
| 1 Architectural Overview | 40 |
| 1.1 Functional Overview | 42 |
| 1.1.1 ARM Cortex™-M3 | 42 |
| 1.1.2 On-Chip Memory | 44 |
| 1.1.3 External Peripheral Interface | 45 |
| 1.1.4 Serial Communications Peripherals | 46 |
| 1.1.5 System Integration | 52 |
| 1.1.6 Analog | 58 |
| 1.1.7 JTAG and ARM Serial Wire Debug | 59 |
| 1.1.8 Packaging and Temperature | 60 |
| 1.2 Target Applications | 60 |
| 1.3 High-Level Block Diagram | 61 |
| 1.4 Additional Features | 63 |
| 1.4.1 Memory Map | 63 |
| 1.4.2 Hardware Details | 63 |
| 2 ARM Cortex-M3 Processor Core | 64 |
| 2.1 Block Diagram | 65 |
| 2.2 Functional Description | 65 |
| 2.2.1 Programming Model | 65 |
| 2.2.2 Serial Wire and JTAG Debug | 72 |
| 2.2.3 Embedded Trace Macrocell (ETM) | 72 |
| 2.2.4 Trace Port Interface Unit (TPIU) | 72 |
| 2.2.5 ROM Table | 73 |
| 2.2.6 Memory Protection Unit (MPU) | 73 |
| 2.2.7 Nested Vectored Interrupt Controller (NVIC) | 73 |
| 2.2.8 System Timer (SysTick) | 74 |
| 3 Memory Map | 77 |
| 4 Interrupts | 80 |
| 5 JTAG Interface | 83 |
| 5.1 Block Diagram | 84 |
| 5.2 Signal Description | 84 |
| 5.3 Functional Description | 85 |
| 5.3.1 JTAG Interface Pins | 85 |
| 5.3.2 JTAG TAP Controller | 86 |
| 5.3.3 Shift Registers | 87 |
| 5.3.4 Operational Considerations | 87 |
| 5.4 Initialization and Configuration | 90 |
| 5.5 Register Descriptions | 90 |

| | | |
|----------|---|------------|
| 5.5.1 | Instruction Register (IR) | 90 |
| 5.5.2 | Data Registers | 92 |
| 6 | System Control | 95 |
| 6.1 | Signal Description | 95 |
| 6.2 | Functional Description | 95 |
| 6.2.1 | Device Identification | 95 |
| 6.2.2 | Reset Control | 95 |
| 6.2.3 | Non-Maskable Interrupt | 99 |
| 6.2.4 | Power Control | 100 |
| 6.2.5 | Clock Control | 101 |
| 6.2.6 | System Control | 106 |
| 6.3 | Initialization and Configuration | 107 |
| 6.4 | Register Map | 107 |
| 6.5 | Register Descriptions | 109 |
| 7 | Hibernation Module | 200 |
| 7.1 | Block Diagram | 201 |
| 7.2 | Signal Description | 201 |
| 7.3 | Functional Description | 202 |
| 7.3.1 | Register Access Timing | 202 |
| 7.3.2 | Clock Source | 202 |
| 7.3.3 | Battery Management | 204 |
| 7.3.4 | Real-Time Clock | 204 |
| 7.3.5 | Non-Volatile Memory | 205 |
| 7.3.6 | Power Control Using HIB | 205 |
| 7.3.7 | Power Control Using VDD3ON Mode | 205 |
| 7.3.8 | Initiating Hibernate | 205 |
| 7.3.9 | Interrupts and Status | 206 |
| 7.4 | Initialization and Configuration | 206 |
| 7.4.1 | Initialization | 206 |
| 7.4.2 | RTC Match Functionality (No Hibernation) | 207 |
| 7.4.3 | RTC Match/Wake-Up from Hibernation | 207 |
| 7.4.4 | External Wake-Up from Hibernation | 208 |
| 7.4.5 | RTC or External Wake-Up from Hibernation | 208 |
| 7.4.6 | Register Reset | 208 |
| 7.5 | Register Map | 209 |
| 7.6 | Register Descriptions | 209 |
| 8 | Internal Memory | 226 |
| 8.1 | Block Diagram | 226 |
| 8.2 | Functional Description | 226 |
| 8.2.1 | SRAM | 227 |
| 8.2.2 | ROM | 227 |
| 8.2.3 | Flash Memory | 227 |
| 8.3 | Flash Memory Initialization and Configuration | 228 |
| 8.3.1 | Flash Memory Programming | 228 |
| 8.3.2 | 32-Word Flash Memory Write Buffer | 230 |
| 8.3.3 | Nonvolatile Register Programming | 230 |
| 8.4 | Register Map | 231 |
| 8.5 | Flash Memory Register Descriptions (Flash Control Offset) | 232 |

| | | |
|-----------|--|------------|
| 8.6 | Memory Register Descriptions (System Control Offset) | 242 |
| 9 | Micro Direct Memory Access (μDMA) | 258 |
| 9.1 | Block Diagram | 259 |
| 9.2 | Functional Description | 259 |
| 9.2.1 | Channel Assignments | 260 |
| 9.2.2 | Priority | 261 |
| 9.2.3 | Arbitration Size | 261 |
| 9.2.4 | Request Types | 261 |
| 9.2.5 | Channel Configuration | 262 |
| 9.2.6 | Transfer Modes | 264 |
| 9.2.7 | Transfer Size and Increment | 272 |
| 9.2.8 | Peripheral Interface | 272 |
| 9.2.9 | Software Request | 272 |
| 9.2.10 | Interrupts and Errors | 273 |
| 9.3 | Initialization and Configuration | 273 |
| 9.3.1 | Module Initialization | 273 |
| 9.3.2 | Configuring a Memory-to-Memory Transfer | 274 |
| 9.3.3 | Configuring a Peripheral for Simple Transmit | 275 |
| 9.3.4 | Configuring a Peripheral for Ping-Pong Receive | 277 |
| 9.3.5 | Configuring Alternate Channels | 279 |
| 9.4 | Register Map | 279 |
| 9.5 | μDMA Channel Control Structure | 281 |
| 9.6 | μDMA Register Descriptions | 288 |
| 10 | General-Purpose Input/Outputs (GPIOs) | 318 |
| 10.1 | Signal Description | 318 |
| 10.2 | Functional Description | 321 |
| 10.2.1 | Data Control | 322 |
| 10.2.2 | Interrupt Control | 323 |
| 10.2.3 | Mode Control | 324 |
| 10.2.4 | Commit Control | 324 |
| 10.2.5 | Pad Control | 325 |
| 10.2.6 | Identification | 325 |
| 10.3 | Initialization and Configuration | 325 |
| 10.4 | Register Map | 326 |
| 10.5 | Register Descriptions | 329 |
| 11 | External Peripheral Interface (EPI) | 372 |
| 11.1 | EPI Block Diagram | 373 |
| 11.2 | Signal Description | 374 |
| 11.3 | Functional Description | 375 |
| 11.3.1 | Non-Blocking Reads | 376 |
| 11.3.2 | DMA Operation | 377 |
| 11.4 | Initialization and Configuration | 377 |
| 11.4.1 | SDRAM Mode | 378 |
| 11.4.2 | Host Bus Mode | 382 |
| 11.4.3 | General-Purpose Mode | 388 |
| 11.5 | Register Map | 391 |
| 11.6 | Register Descriptions | 392 |

| | | |
|-----------|--|------------|
| 12 | General-Purpose Timers | 433 |
| 12.1 | Block Diagram | 434 |
| 12.2 | Signal Description | 434 |
| 12.3 | Functional Description | 436 |
| 12.3.1 | GPTM Reset Conditions | 436 |
| 12.3.2 | 32-Bit Timer Operating Modes | 436 |
| 12.3.3 | 16-Bit Timer Operating Modes | 438 |
| 12.3.4 | DMA Operation | 443 |
| 12.4 | Initialization and Configuration | 443 |
| 12.4.1 | 32-Bit One-Shot/Periodic Timer Mode | 443 |
| 12.4.2 | 32-Bit Real-Time Clock (RTC) Mode | 444 |
| 12.4.3 | 16-Bit One-Shot/Periodic Timer Mode | 444 |
| 12.4.4 | 16-Bit Input Edge-Count Mode | 445 |
| 12.4.5 | 16-Bit Input Edge Timing Mode | 445 |
| 12.4.6 | 16-Bit PWM Mode | 446 |
| 12.5 | Register Map | 446 |
| 12.6 | Register Descriptions | 447 |
| 13 | Watchdog Timers | 476 |
| 13.1 | Block Diagram | 477 |
| 13.2 | Functional Description | 477 |
| 13.2.1 | Register Access Timing | 478 |
| 13.3 | Initialization and Configuration | 478 |
| 13.4 | Register Map | 478 |
| 13.5 | Register Descriptions | 479 |
| 14 | Analog-to-Digital Converter (ADC) | 501 |
| 14.1 | Block Diagram | 502 |
| 14.2 | Signal Description | 503 |
| 14.3 | Functional Description | 503 |
| 14.3.1 | Sample Sequencers | 504 |
| 14.3.2 | Module Control | 504 |
| 14.3.3 | Hardware Sample Averaging Circuit | 506 |
| 14.3.4 | Analog-to-Digital Converter | 506 |
| 14.3.5 | Differential Sampling | 508 |
| 14.3.6 | Internal Temperature Sensor | 511 |
| 14.3.7 | Digital Comparator Unit | 511 |
| 14.4 | Initialization and Configuration | 516 |
| 14.4.1 | Module Initialization | 516 |
| 14.4.2 | Sample Sequencer Configuration | 517 |
| 14.5 | Register Map | 517 |
| 14.6 | Register Descriptions | 519 |
| 15 | Universal Asynchronous Receivers/Transmitters (UARTs) | 573 |
| 15.1 | Block Diagram | 574 |
| 15.2 | Signal Description | 574 |
| 15.3 | Functional Description | 575 |
| 15.3.1 | Transmit/Receive Logic | 576 |
| 15.3.2 | Baud-Rate Generation | 576 |
| 15.3.3 | Data Transmission | 577 |
| 15.3.4 | Serial IR (SIR) | 577 |

| | | |
|-----------|--|------------|
| 15.3.5 | ISO 7816 Support | 578 |
| 15.3.6 | Modem Handshake Support | 579 |
| 15.3.7 | LIN Support | 580 |
| 15.3.8 | FIFO Operation | 581 |
| 15.3.9 | Interrupts | 581 |
| 15.3.10 | Loopback Operation | 582 |
| 15.3.11 | DMA Operation | 582 |
| 15.4 | Initialization and Configuration | 583 |
| 15.5 | Register Map | 584 |
| 15.6 | Register Descriptions | 585 |
| 16 | Synchronous Serial Interface (SSI) | 634 |
| 16.1 | Block Diagram | 635 |
| 16.2 | Signal Description | 635 |
| 16.3 | Functional Description | 636 |
| 16.3.1 | Bit Rate Generation | 636 |
| 16.3.2 | FIFO Operation | 637 |
| 16.3.3 | Interrupts | 637 |
| 16.3.4 | Frame Formats | 638 |
| 16.3.5 | DMA Operation | 645 |
| 16.4 | Initialization and Configuration | 646 |
| 16.5 | Register Map | 647 |
| 16.6 | Register Descriptions | 648 |
| 17 | Inter-Integrated Circuit (I²C) Interface | 676 |
| 17.1 | Block Diagram | 677 |
| 17.2 | Signal Description | 677 |
| 17.3 | Functional Description | 678 |
| 17.3.1 | I ² C Bus Functional Overview | 678 |
| 17.3.2 | Available Speed Modes | 680 |
| 17.3.3 | Interrupts | 681 |
| 17.3.4 | Loopback Operation | 682 |
| 17.3.5 | Command Sequence Flow Charts | 682 |
| 17.4 | Initialization and Configuration | 689 |
| 17.5 | Register Map | 690 |
| 17.6 | Register Descriptions (I ² C Master) | 691 |
| 17.7 | Register Descriptions (I ² C Slave) | 704 |
| 18 | Inter-Integrated Circuit Sound (I²S) Interface | 713 |
| 18.1 | Block Diagram | 714 |
| 18.2 | Signal Description | 714 |
| 18.3 | Functional Description | 715 |
| 18.3.1 | Transmit | 717 |
| 18.3.2 | Receive | 721 |
| 18.4 | Initialization and Configuration | 722 |
| 18.5 | Register Map | 724 |
| 18.6 | Register Descriptions | 725 |
| 19 | Controller Area Network (CAN) Module | 750 |
| 19.1 | Block Diagram | 751 |
| 19.2 | Signal Description | 751 |

| | | |
|-----------|---|------------|
| 19.3 | Functional Description | 752 |
| 19.3.1 | Initialization | 753 |
| 19.3.2 | Operation | 753 |
| 19.3.3 | Transmitting Message Objects | 754 |
| 19.3.4 | Configuring a Transmit Message Object | 755 |
| 19.3.5 | Updating a Transmit Message Object | 756 |
| 19.3.6 | Accepting Received Message Objects | 756 |
| 19.3.7 | Receiving a Data Frame | 757 |
| 19.3.8 | Receiving a Remote Frame | 757 |
| 19.3.9 | Receive/Transmit Priority | 758 |
| 19.3.10 | Configuring a Receive Message Object | 758 |
| 19.3.11 | Handling of Received Message Objects | 759 |
| 19.3.12 | Handling of Interrupts | 761 |
| 19.3.13 | Test Mode | 762 |
| 19.3.14 | Bit Timing Configuration Error Considerations | 764 |
| 19.3.15 | Bit Time and Bit Rate | 764 |
| 19.3.16 | Calculating the Bit Timing Parameters | 766 |
| 19.4 | Register Map | 768 |
| 19.5 | CAN Register Descriptions | 769 |
| 20 | Ethernet Controller | 800 |
| 20.1 | Block Diagram | 801 |
| 20.2 | Signal Description | 802 |
| 20.3 | Functional Description | 802 |
| 20.3.1 | MAC Operation | 802 |
| 20.3.2 | Internal MII Operation | 806 |
| 20.3.3 | PHY Operation | 806 |
| 20.3.4 | Interrupts | 807 |
| 20.3.5 | DMA Operation | 808 |
| 20.4 | Initialization and Configuration | 808 |
| 20.4.1 | Hardware Configuration | 808 |
| 20.4.2 | Software Configuration | 809 |
| 20.5 | Ethernet Register Map | 810 |
| 20.6 | Ethernet MAC Register Descriptions | 812 |
| 20.7 | MII Management Register Descriptions | 836 |
| 21 | Universal Serial Bus (USB) Controller | 857 |
| 21.1 | Block Diagram | 858 |
| 21.2 | Signal Description | 858 |
| 21.3 | Functional Description | 859 |
| 21.3.1 | Operation as a Device | 859 |
| 21.3.2 | Operation as a Host | 864 |
| 21.3.3 | OTG Mode | 868 |
| 21.3.4 | DMA Operation | 870 |
| 21.4 | Initialization and Configuration | 871 |
| 21.4.1 | Pin Configuration | 871 |
| 21.4.2 | Endpoint Configuration | 871 |
| 21.5 | Register Map | 872 |
| 21.6 | Register Descriptions | 883 |

| | | |
|-----------|---|-------------|
| 22 | Analog Comparators | 995 |
| 22.1 | Block Diagram | 996 |
| 22.2 | Signal Description | 996 |
| 22.3 | Functional Description | 997 |
| 22.3.1 | Internal Reference Programming | 998 |
| 22.4 | Initialization and Configuration | 999 |
| 22.5 | Register Map | 1000 |
| 22.6 | Register Descriptions | 1000 |
| 23 | Pin Diagram | 1008 |
| 24 | Signal Tables | 1009 |
| 25 | Operating Characteristics | 1038 |
| 26 | Electrical Characteristics | 1039 |
| 26.1 | DC Characteristics | 1039 |
| 26.1.1 | Maximum Ratings | 1039 |
| 26.1.2 | Recommended DC Operating Conditions | 1039 |
| 26.1.3 | On-Chip Low Drop-Out (LDO) Regulator Characteristics | 1040 |
| 26.1.4 | Hibernation Module Characteristics | 1040 |
| 26.1.5 | Flash Memory Characteristics | 1040 |
| 26.1.6 | GPIO Module Characteristics | 1041 |
| 26.1.7 | USB Module Characteristics | 1041 |
| 26.1.8 | Ethernet Controller Characteristics | 1041 |
| 26.1.9 | Current Specifications | 1041 |
| 26.2 | AC Characteristics | 1042 |
| 26.2.1 | Load Conditions | 1042 |
| 26.2.2 | Clocks | 1042 |
| 26.2.3 | JTAG and Boundary Scan | 1045 |
| 26.2.4 | Reset | 1046 |
| 26.2.5 | Hibernation Module | 1048 |
| 26.2.6 | General-Purpose I/O (GPIO) | 1048 |
| 26.2.7 | External Peripheral Interface (EPI) | 1049 |
| 26.2.8 | Analog-to-Digital Converter | 1053 |
| 26.2.9 | ADC Module Characteristics | 1054 |
| 26.2.10 | Synchronous Serial Interface (SSI) | 1054 |
| 26.2.11 | Inter-Integrated Circuit (I ² C) Interface | 1055 |
| 26.2.12 | Inter-Integrated Circuit Sound (I ² S) Interface | 1056 |
| 26.2.13 | Ethernet Controller | 1057 |
| 26.2.14 | Universal Serial Bus (USB) Controller | 1060 |
| 26.2.15 | Analog Comparator | 1060 |
| A | Boot Loader | 1061 |
| A.1 | Boot Loader Overview | 1061 |
| A.2 | Serial Interfaces | 1061 |
| A.2.1 | Serial Configuration | 1061 |
| A.2.2 | Serial Packet Handling | 1062 |
| A.2.3 | Serial Commands | 1063 |
| A.3 | Ethernet Interface | 1065 |
| B | ROM DriverLib Functions | 1066 |
| B.1 | DriverLib Functions Included in the Integrated ROM | 1066 |

| | | |
|----------|--|-------------|
| C | Advance Encryption Standard and Cyclic Redundancy Check Software in ROM ... | 1088 |
| C.1 | Advanced Encryption Standard Software | 1088 |
| C.2 | Cyclic Redundancy Check Software | 1088 |
| D | Register Quick Reference | 1089 |
| E | Ordering and Contact Information | 1132 |
| E.1 | Ordering Information | 1132 |
| E.2 | Part Markings | 1132 |
| E.3 | Kits | 1132 |
| E.4 | Support Information | 1133 |
| F | Package Information | 1134 |

List of Figures

| | | |
|--------------|--|-----|
| Figure 1-1. | Stellaris® LM3S9B90 Microcontroller High-Level Block Diagram | 62 |
| Figure 2-1. | CPU Block Diagram | 65 |
| Figure 2-2. | TPIU Block Diagram | 73 |
| Figure 5-1. | JTAG Module Block Diagram | 84 |
| Figure 5-2. | Test Access Port State Machine | 87 |
| Figure 5-3. | IDCODE Register Format | 93 |
| Figure 5-4. | BYPASS Register Format | 93 |
| Figure 5-5. | Boundary Scan Register Format | 94 |
| Figure 6-1. | Basic \overline{RST} Configuration | 97 |
| Figure 6-2. | External Circuitry to Extend Power-On Reset | 97 |
| Figure 6-3. | Reset Circuit Controlled by Switch | 98 |
| Figure 6-4. | Power Architecture | 101 |
| Figure 6-5. | Main Clock Tree | 103 |
| Figure 7-1. | Hibernation Module Block Diagram | 201 |
| Figure 7-2. | Clock Source Using Crystal | 203 |
| Figure 7-3. | Clock Source Using Dedicated Oscillator and VDD3ON Mode | 204 |
| Figure 8-1. | Internal Memory Block Diagram | 226 |
| Figure 9-1. | μ DMA Block Diagram | 259 |
| Figure 9-2. | Example of Ping-Pong μ DMA Transaction | 265 |
| Figure 9-3. | Memory Scatter-Gather, Setup and Configuration | 267 |
| Figure 9-4. | Memory Scatter-Gather, μ DMA Copy Sequence | 268 |
| Figure 9-5. | Peripheral Scatter-Gather, Setup and Configuration | 270 |
| Figure 9-6. | Peripheral Scatter-Gather, μ DMA Copy Sequence | 271 |
| Figure 10-1. | Digital I/O Pads | 321 |
| Figure 10-2. | Analog/Digital I/O Pads | 322 |
| Figure 10-3. | GPIO DATA Write Example | 323 |
| Figure 10-4. | GPIO DATA Read Example | 323 |
| Figure 11-1. | EPI Block Diagram | 374 |
| Figure 11-2. | SDRAM Non-Blocking Read Cycle | 380 |
| Figure 11-3. | SDRAM Normal Read Cycle | 381 |
| Figure 11-4. | SDRAM Write Cycle | 382 |
| Figure 12-1. | GPTM Module Block Diagram | 434 |
| Figure 12-2. | 16-Bit Input Edge-Count Mode Example | 440 |
| Figure 12-3. | 16-Bit Input Edge-Time Mode Example | 441 |
| Figure 12-4. | 16-Bit PWM Mode Example | 442 |
| Figure 12-5. | Timer Daisy Chain | 442 |
| Figure 13-1. | WDT Module Block Diagram | 477 |
| Figure 14-1. | Implementation of Two ADC Blocks | 502 |
| Figure 14-2. | ADC Module Block Diagram | 502 |
| Figure 14-3. | Internal Voltage Conversion Result | 507 |
| Figure 14-4. | External Voltage Conversion Result | 508 |
| Figure 14-5. | Differential Sampling Range, $V_{IN_ODD} = 1.5\text{ V}$ | 509 |
| Figure 14-6. | Differential Sampling Range, $V_{IN_ODD} = 0.75\text{ V}$ | 510 |
| Figure 14-7. | Differential Sampling Range, $V_{IN_ODD} = 2.25\text{ V}$ | 510 |
| Figure 14-8. | Internal Temperature Sensor Characteristic | 511 |

| | | |
|---------------|---|-----|
| Figure 14-9. | Low-Band Operation (CIC=0x0 and/or CTC=0x0) | 514 |
| Figure 14-10. | Mid-Band Operation (CIC=0x1 and/or CTC=0x1) | 515 |
| Figure 14-11. | High-Band Operation (CIC=0x3 and/or CTC=0x3) | 516 |
| Figure 15-1. | UART Module Block Diagram | 574 |
| Figure 15-2. | UART Character Frame | 576 |
| Figure 15-3. | IrDA Data Modulation | 578 |
| Figure 15-4. | LIN Message | 580 |
| Figure 15-5. | LIN Synchronization Field | 581 |
| Figure 16-1. | SSI Module Block Diagram | 635 |
| Figure 16-2. | TI Synchronous Serial Frame Format (Single Transfer) | 638 |
| Figure 16-3. | TI Synchronous Serial Frame Format (Continuous Transfer) | 639 |
| Figure 16-4. | Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 | 640 |
| Figure 16-5. | Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 | 640 |
| Figure 16-6. | Freescale SPI Frame Format with SPO=0 and SPH=1 | 641 |
| Figure 16-7. | Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 | 642 |
| Figure 16-8. | Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 | 642 |
| Figure 16-9. | Freescale SPI Frame Format with SPO=1 and SPH=1 | 643 |
| Figure 16-10. | MICROWIRE Frame Format (Single Frame) | 644 |
| Figure 16-11. | MICROWIRE Frame Format (Continuous Transfer) | 645 |
| Figure 16-12. | MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements | 645 |
| Figure 17-1. | I ² C Block Diagram | 677 |
| Figure 17-2. | I ² C Bus Configuration | 678 |
| Figure 17-3. | START and STOP Conditions | 678 |
| Figure 17-4. | Complete Data Transfer with a 7-Bit Address | 679 |
| Figure 17-5. | R/S Bit in First Byte | 679 |
| Figure 17-6. | Data Validity During Bit Transfer on the I ² C Bus | 679 |
| Figure 17-7. | Master Single TRANSMIT | 683 |
| Figure 17-8. | Master Single RECEIVE | 684 |
| Figure 17-9. | Master TRANSMIT with Repeated START | 685 |
| Figure 17-10. | Master RECEIVE with Repeated START | 686 |
| Figure 17-11. | Master RECEIVE with Repeated START after TRANSMIT with Repeated START | 687 |
| Figure 17-12. | Master TRANSMIT with Repeated START after RECEIVE with Repeated START | 688 |
| Figure 17-13. | Slave Command Sequence | 689 |
| Figure 18-1. | I ² S Block Diagram | 714 |
| Figure 18-2. | I ² S Data Transfer | 716 |
| Figure 18-3. | Left-Justified Data Transfer | 716 |
| Figure 18-4. | Right-Justified Data Transfer | 717 |
| Figure 19-1. | CAN Controller Block Diagram | 751 |
| Figure 19-2. | CAN Data/Remote Frame | 752 |
| Figure 19-3. | Message Objects in a FIFO Buffer | 761 |
| Figure 19-4. | CAN Bit Time | 765 |
| Figure 20-1. | Ethernet Controller | 801 |
| Figure 20-2. | Ethernet Controller Block Diagram | 801 |
| Figure 20-3. | Ethernet Frame | 803 |
| Figure 20-4. | Interface to an Ethernet Jack | 809 |
| Figure 21-1. | USB Module Block Diagram | 858 |
| Figure 22-1. | Analog Comparator Module Block Diagram | 996 |
| Figure 22-2. | Structure of Comparator Unit | 997 |

| | | |
|---------------|---|------|
| Figure 22-3. | Comparator Internal Reference Structure | 998 |
| Figure 23-1. | 100-Pin LQFP Package Pin Diagram | 1008 |
| Figure 26-1. | Load Conditions | 1042 |
| Figure 26-2. | JTAG Test Clock Input Timing | 1045 |
| Figure 26-3. | JTAG Test Access Port (TAP) Timing | 1046 |
| Figure 26-4. | External Reset Timing (RST) | 1046 |
| Figure 26-5. | Power-On Reset Timing | 1047 |
| Figure 26-6. | Brown-Out Reset Timing | 1047 |
| Figure 26-7. | Software Reset Timing | 1047 |
| Figure 26-8. | Watchdog Reset Timing | 1047 |
| Figure 26-9. | MOSC Failure Reset Timing | 1048 |
| Figure 26-10. | Hibernation Module Timing | 1048 |
| Figure 26-11. | SDRAM Initialization and Load Mode Register Timing | 1050 |
| Figure 26-12. | SDRAM Read Timing | 1050 |
| Figure 26-13. | SDRAM Write Timing | 1051 |
| Figure 26-14. | Host-Bus 8/16 Mode Read Timing | 1051 |
| Figure 26-15. | Host-Bus 8/16 Mode Write Timing | 1052 |
| Figure 26-16. | General-Purpose Mode Read and Write Timing | 1052 |
| Figure 26-17. | ADC Input Equivalency Diagram | 1053 |
| Figure 26-18. | SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement | 1054 |
| Figure 26-19. | SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer | 1055 |
| Figure 26-20. | SSI Timing for SPI Frame Format (FRF=00), with SPH=1 | 1055 |
| Figure 26-21. | I ² C Timing | 1055 |
| Figure 26-22. | I ² S Master Mode Transmit Timing | 1056 |
| Figure 26-23. | I ² S Master Mode Receive Timing | 1056 |
| Figure 26-24. | I ² S Slave Mode Transmit Timing | 1057 |
| Figure 26-25. | I ² S Slave Mode Receive Timing | 1057 |
| Figure 26-26. | External XTLP Oscillator Characteristics | 1059 |
| Figure F-1. | 100-Pin LQFP Package | 1134 |

List of Tables

| | | |
|--------------|---|-----|
| Table 1. | Revision History | 35 |
| Table 2. | Documentation Conventions | 37 |
| Table 2-1. | 16-Bit Cortex-M3 Instruction Set Summary | 66 |
| Table 2-2. | 32-Bit Cortex-M3 Instruction Set Summary | 68 |
| Table 3-1. | Memory Map | 77 |
| Table 4-1. | Exception Types | 80 |
| Table 4-2. | Interrupts | 81 |
| Table 5-1. | Signals for JTAG_SWD_SWO | 84 |
| Table 5-2. | JTAG Port Pins State after Power-On Reset or \overline{RST} assertion | 85 |
| Table 5-3. | JTAG Instruction Register Commands | 91 |
| Table 6-1. | Signals for System Control & Clocks | 95 |
| Table 6-2. | Reset Sources | 96 |
| Table 6-3. | Clock Source Options | 102 |
| Table 6-4. | System Control Register Map | 107 |
| Table 6-5. | Examples of Possible System Clock Frequencies | 129 |
| Table 7-1. | Signals for Hibernate | 201 |
| Table 7-2. | Hibernation Module Clock Operation | 207 |
| Table 7-3. | Hibernation Module Register Map | 209 |
| Table 8-1. | Flash Memory Protection Policy Combinations | 228 |
| Table 8-2. | User-Programmable Flash Memory Resident Registers | 231 |
| Table 8-3. | Flash Register Map | 231 |
| Table 9-1. | μ DMA Channel Assignments | 260 |
| Table 9-2. | Request Type Support | 261 |
| Table 9-3. | Control Structure Memory Map | 263 |
| Table 9-4. | Channel Control Structure | 263 |
| Table 9-5. | μ DMA Read Example: 8-Bit Peripheral | 272 |
| Table 9-6. | μ DMA Interrupt Assignments | 273 |
| Table 9-7. | Channel Control Structure Offsets for Channel 30 | 274 |
| Table 9-8. | Channel Control Word Configuration for Memory Transfer Example | 274 |
| Table 9-9. | Channel Control Structure Offsets for Channel 7 | 275 |
| Table 9-10. | Channel Control Word Configuration for Peripheral Transmit Example | 276 |
| Table 9-11. | Primary and Alternate Channel Control Structure Offsets for Channel 8 | 277 |
| Table 9-12. | Channel Control Word Configuration for Peripheral Ping-Pong Receive Example | 278 |
| Table 9-13. | μ DMA Register Map | 280 |
| Table 10-1. | GPIO Pins With Non-Zero Reset Values | 319 |
| Table 10-2. | GPIO Pins and Alternate Functions | 319 |
| Table 10-3. | GPIO Pad Configuration Examples | 325 |
| Table 10-4. | GPIO Interrupt Configuration Example | 326 |
| Table 10-5. | GPIO Pins With Non-Zero Reset Values | 327 |
| Table 10-6. | GPIO Register Map | 328 |
| Table 10-7. | GPIO Pins With Non-Zero Reset Values | 340 |
| Table 10-8. | GPIO Pins With Non-Zero Reset Values | 346 |
| Table 10-9. | GPIO Pins With Non-Zero Reset Values | 348 |
| Table 10-10. | GPIO Pins With Non-Zero Reset Values | 351 |
| Table 10-11. | GPIO Pins With Non-Zero Reset Values | 358 |
| Table 11-1. | Signals for External Peripheral Interface | 374 |

| | | |
|-------------|--|-----|
| Table 11-2. | EPI SDRAM Signal Connections | 378 |
| Table 11-3. | SDRAM Interface Configurations | 380 |
| Table 11-4. | Capabilities of Host Bus 8 and Host Bus 16 Modes | 383 |
| Table 11-5. | EPI Host-Bus 8 Signal Connections | 383 |
| Table 11-6. | EPI Host-Bus 16 Signal Connections | 385 |
| Table 11-7. | EPI General Purpose Signal Connections | 390 |
| Table 11-8. | External Peripheral Interface (EPI) Register Map | 391 |
| Table 12-1. | Available CCP Pins | 434 |
| Table 12-2. | Signals for General-Purpose Timers | 435 |
| Table 12-3. | 16-Bit Timer With Prescaler Configurations | 439 |
| Table 12-4. | Timers Register Map | 447 |
| Table 13-1. | Watchdog Timers Register Map | 479 |
| Table 14-1. | Signals for ADC | 503 |
| Table 14-2. | Samples and FIFO Depth of Sequencers | 504 |
| Table 14-3. | Differential Sampling Pairs | 508 |
| Table 14-4. | ADC Register Map | 517 |
| Table 15-1. | Signals for UART | 575 |
| Table 15-2. | Flow Control Mode | 580 |
| Table 15-3. | UART Register Map | 584 |
| Table 16-1. | Signals for SSI | 636 |
| Table 16-2. | SSI Register Map | 647 |
| Table 17-1. | Signals for I ² C | 677 |
| Table 17-2. | Examples of I ² C Master Timer Period versus Speed Mode | 681 |
| Table 17-3. | Inter-Integrated Circuit (I ² C) Interface Register Map | 690 |
| Table 17-4. | Write Field Decoding for I2CMCS[3:0] Field | 696 |
| Table 18-1. | Signals for I ² S | 715 |
| Table 18-2. | I ² S Transmit FIFO Interface | 717 |
| Table 18-3. | Crystal Frequency (Values from 3.5795 MHz to 5 MHz) | 718 |
| Table 18-4. | Crystal Frequency (Values from 5.12 MHz to 8.192 MHz) | 719 |
| Table 18-5. | Crystal Frequency (Values from 10 MHz to 14.3181 MHz) | 719 |
| Table 18-6. | Crystal Frequency (Values from 16 MHz to 16.384 MHz) | 720 |
| Table 18-7. | I ² S Receive FIFO Interface | 721 |
| Table 18-8. | Audio Formats Configuration | 723 |
| Table 18-9. | Inter-Integrated Circuit Sound (I ² S) Interface Register Map | 724 |
| Table 19-1. | Signals for Controller Area Network | 752 |
| Table 19-2. | Message Object Configurations | 757 |
| Table 19-3. | CAN Protocol Ranges | 765 |
| Table 19-4. | CAN Register Map | 768 |
| Table 20-1. | Signals for Ethernet PHY | 802 |
| Table 20-2. | TX & RX FIFO Organization | 804 |
| Table 20-3. | Ethernet Register Map | 810 |
| Table 21-1. | Signals for USB | 858 |
| Table 21-2. | Remainder (RxMaxP/4) | 870 |
| Table 21-3. | Actual Bytes Read | 870 |
| Table 21-4. | Packet Sizes That Clear RXRDY | 870 |
| Table 21-5. | Universal Serial Bus (USB) Controller Register Map | 872 |
| Table 22-1. | Signals for Analog Comparators | 996 |
| Table 22-2. | Internal Reference Voltage and ACREFACTL Field Values | 998 |

| | | |
|--------------|--|------|
| Table 22-3. | Analog Comparators Register Map | 1000 |
| Table 24-1. | GPIO Pins With Default Alternate Functions | 1009 |
| Table 24-2. | Signals by Pin Number | 1009 |
| Table 24-3. | Signals by Signal Name | 1019 |
| Table 24-4. | Signals by Function, Except for GPIO | 1028 |
| Table 24-5. | GPIO Pins and Alternate Functions | 1035 |
| Table 25-1. | Temperature Characteristics | 1038 |
| Table 25-2. | Thermal Characteristics | 1038 |
| Table 26-1. | Maximum Ratings | 1039 |
| Table 26-2. | Recommended DC Operating Conditions | 1039 |
| Table 26-3. | LDO Regulator Characteristics | 1040 |
| Table 26-4. | Hibernation Module DC Characteristics | 1040 |
| Table 26-5. | Flash Memory Characteristics | 1040 |
| Table 26-6. | GPIO Module DC Characteristics | 1041 |
| Table 26-7. | USB Controller DC Characteristics | 1041 |
| Table 26-8. | Ethernet Controller DC Characteristics | 1041 |
| Table 26-9. | Preliminary Current Consumption | 1041 |
| Table 26-10. | Phase Locked Loop (PLL) Characteristics | 1042 |
| Table 26-11. | Actual PLL Frequency | 1043 |
| Table 26-12. | PIOSC Clock Characteristics | 1043 |
| Table 26-13. | 30-kHz Clock Characteristics | 1044 |
| Table 26-14. | Hibernation Clock Characteristics | 1044 |
| Table 26-15. | HIB Oscillator Input Characteristics | 1044 |
| Table 26-16. | Main Oscillator Clock Characteristics | 1044 |
| Table 26-17. | MOSC Oscillator Input Characteristics | 1044 |
| Table 26-18. | JTAG Characteristics | 1045 |
| Table 26-19. | Reset Characteristics | 1046 |
| Table 26-20. | Hibernation Module AC Characteristics | 1048 |
| Table 26-21. | GPIO Characteristics | 1049 |
| Table 26-22. | EPI SDRAM Characteristics | 1049 |
| Table 26-23. | EPI SDRAM Interface Characteristics | 1049 |
| Table 26-24. | EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics | 1051 |
| Table 26-25. | EPI General-Purpose Interface Characteristics | 1052 |
| Table 26-26. | ADC Characteristics | 1053 |
| Table 26-27. | ADC Module External Reference Characteristics | 1054 |
| Table 26-28. | ADC Module Internal Reference Characteristics | 1054 |
| Table 26-29. | SSI Characteristics | 1054 |
| Table 26-30. | I ² S Master Clock (Receive and Transmit) | 1056 |
| Table 26-31. | I ² S Slave Clock (Receive and Transmit) | 1056 |
| Table 26-32. | I ² S Master Mode | 1056 |
| Table 26-33. | I ² S Slave Mode | 1057 |
| Table 26-34. | 100BASE-TX Transmitter Characteristics | 1057 |
| Table 26-35. | 100BASE-TX Transmitter Characteristics (informative) | 1058 |
| Table 26-36. | 100BASE-TX Receiver Characteristics | 1058 |
| Table 26-37. | 10BASE-T Transmitter Characteristics | 1058 |
| Table 26-38. | 10BASE-T Transmitter Characteristics (informative) | 1058 |
| Table 26-39. | 10BASE-T Receiver Characteristics | 1058 |
| Table 26-40. | Isolation Transformers | 1059 |

| | | |
|--------------|---|------|
| Table 26-41. | Ethernet Reference Crystal | 1059 |
| Table 26-42. | External XTLP Oscillator Characteristics | 1060 |
| Table 26-43. | Analog Comparator Characteristics | 1060 |
| Table 26-44. | Analog Comparator Voltage Reference Characteristics | 1060 |
| Table E-1. | Part Ordering Information | 1132 |

List of Registers

| | |
|--|------------|
| System Control | 95 |
| Register 1: Device Identification 0 (DID0), offset 0x000 | 110 |
| Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030 | 112 |
| Register 3: Raw Interrupt Status (RIS), offset 0x050 | 113 |
| Register 4: Interrupt Mask Control (IMC), offset 0x054 | 115 |
| Register 5: Masked Interrupt Status and Clear (MISC), offset 0x058 | 117 |
| Register 6: Reset Cause (RESC), offset 0x05C | 119 |
| Register 7: Run-Mode Clock Configuration (RCC), offset 0x060 | 121 |
| Register 8: XTAL to PLL Translation (PLLCFG), offset 0x064 | 126 |
| Register 9: GPIO Host-Bus Control (GPIOHBCTL), offset 0x06C | 127 |
| Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070 | 129 |
| Register 11: Main Oscillator Control (MOSCCTL), offset 0x07C | 132 |
| Register 12: Deep Sleep Clock Configuration (DSLPCCLKCFG), offset 0x144 | 133 |
| Register 13: Deep Sleep Flash Configuration (DSFLASHCFG), offset 0x14C | 135 |
| Register 14: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150 | 136 |
| Register 15: Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154 | 138 |
| Register 16: I ² S MCLK Configuration (I2SMCLKCFG), offset 0x170 | 139 |
| Register 17: Device Identification 1 (DID1), offset 0x004 | 141 |
| Register 18: Device Capabilities 0 (DC0), offset 0x008 | 143 |
| Register 19: Device Capabilities 1 (DC1), offset 0x010 | 144 |
| Register 20: Device Capabilities 2 (DC2), offset 0x014 | 147 |
| Register 21: Device Capabilities 3 (DC3), offset 0x018 | 149 |
| Register 22: Device Capabilities 4 (DC4), offset 0x01C | 152 |
| Register 23: Device Capabilities 5 (DC5), offset 0x020 | 154 |
| Register 24: Device Capabilities 6 (DC6), offset 0x024 | 155 |
| Register 25: Device Capabilities 7 (DC7), offset 0x028 | 156 |
| Register 26: Device Capabilities 8 ADC Channels (DC8), offset 0x02C | 160 |
| Register 27: Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190 | 163 |
| Register 28: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0 | 165 |
| Register 29: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 | 166 |
| Register 30: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 | 169 |
| Register 31: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 | 172 |
| Register 32: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 | 175 |
| Register 33: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 | 178 |
| Register 34: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 | 181 |
| Register 35: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 | 184 |
| Register 36: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 | 187 |
| Register 37: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 | 190 |
| Register 38: Software Reset Control 0 (SRCR0), offset 0x040 | 193 |
| Register 39: Software Reset Control 1 (SRCR1), offset 0x044 | 195 |
| Register 40: Software Reset Control 2 (SRCR2), offset 0x048 | 198 |
| Hibernation Module | 200 |
| Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000 | 210 |
| Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004 | 211 |
| Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008 | 212 |

| | | |
|--|---|------------|
| Register 4: | Hibernation RTC Load (HIBRTCLD), offset 0x00C | 213 |
| Register 5: | Hibernation Control (HIBCTL), offset 0x010 | 214 |
| Register 6: | Hibernation Interrupt Mask (HIBIM), offset 0x014 | 217 |
| Register 7: | Hibernation Raw Interrupt Status (HIBRIS), offset 0x018 | 219 |
| Register 8: | Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C | 221 |
| Register 9: | Hibernation Interrupt Clear (HIBIC), offset 0x020 | 223 |
| Register 10: | Hibernation RTC Trim (HIBRTCT), offset 0x024 | 224 |
| Register 11: | Hibernation Data (HIBDATA), offset 0x030-0x12C | 225 |
| Internal Memory | | 226 |
| Register 1: | Flash Memory Address (FMA), offset 0x000 | 233 |
| Register 2: | Flash Memory Data (FMD), offset 0x004 | 234 |
| Register 3: | Flash Memory Control (FMC), offset 0x008 | 235 |
| Register 4: | Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C | 237 |
| Register 5: | Flash Controller Interrupt Mask (FCIM), offset 0x010 | 238 |
| Register 6: | Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 | 239 |
| Register 7: | Flash Memory Control 2 (FMC2), offset 0x020 | 240 |
| Register 8: | Flash Write Buffer Valid (FWBVAL), offset 0x030 | 241 |
| Register 9: | Flash Write Buffer n (FWBn), offset 0x100 - 0x13C | 242 |
| Register 10: | ROM Control (RMCTL), offset 0x0F0 | 243 |
| Register 11: | ROM Version Register (RMVER), offset 0x0F4 | 244 |
| Register 12: | Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200 | 245 |
| Register 13: | Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400 | 246 |
| Register 14: | User Debug (USER_DBG), offset 0x1D0 | 247 |
| Register 15: | User Register 0 (USER_REG0), offset 0x1E0 | 248 |
| Register 16: | User Register 1 (USER_REG1), offset 0x1E4 | 249 |
| Register 17: | User Register 2 (USER_REG2), offset 0x1E8 | 250 |
| Register 18: | User Register 3 (USER_REG3), offset 0x1EC | 251 |
| Register 19: | Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204 | 252 |
| Register 20: | Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208 | 253 |
| Register 21: | Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C | 254 |
| Register 22: | Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404 | 255 |
| Register 23: | Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408 | 256 |
| Register 24: | Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C | 257 |
| Micro Direct Memory Access (μDMA) | | 258 |
| Register 1: | DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000 | 282 |
| Register 2: | DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004 | 283 |
| Register 3: | DMA Channel Control Word (DMACHCTL), offset 0x008 | 284 |
| Register 4: | DMA Status (DMASSTAT), offset 0x000 | 289 |
| Register 5: | DMA Configuration (DMACFG), offset 0x004 | 291 |
| Register 6: | DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008 | 292 |
| Register 7: | DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C | 293 |
| Register 8: | DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010 | 294 |
| Register 9: | DMA Channel Software Request (DMASWREQ), offset 0x014 | 295 |
| Register 10: | DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018 | 296 |
| Register 11: | DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C | 297 |
| Register 12: | DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020 | 298 |
| Register 13: | DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024 | 299 |
| Register 14: | DMA Channel Enable Set (DMAENASET), offset 0x028 | 300 |

| | | |
|--|--|-----|
| Register 15: | DMA Channel Enable Clear (DMAENACLR), offset 0x02C | 301 |
| Register 16: | DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030 | 302 |
| Register 17: | DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034 | 303 |
| Register 18: | DMA Channel Priority Set (DMAPRIOSET), offset 0x038 | 304 |
| Register 19: | DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C | 305 |
| Register 20: | DMA Bus Error Clear (DMAERRCLR), offset 0x04C | 306 |
| Register 21: | DMA Channel Alternate Select (DMACHALT), offset 0x500 | 307 |
| Register 22: | DMA Channel Interrupt Status (DMACHIS), offset 0x504 | 308 |
| Register 23: | DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0 | 309 |
| Register 24: | DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4 | 310 |
| Register 25: | DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8 | 311 |
| Register 26: | DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC | 312 |
| Register 27: | DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0 | 313 |
| Register 28: | DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0 | 314 |
| Register 29: | DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4 | 315 |
| Register 30: | DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8 | 316 |
| Register 31: | DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC | 317 |
| General-Purpose Input/Outputs (GPIOs) | 318 | |
| Register 1: | GPIO Data (GPIODATA), offset 0x000 | 330 |
| Register 2: | GPIO Direction (GPIODIR), offset 0x400 | 331 |
| Register 3: | GPIO Interrupt Sense (GPIOIS), offset 0x404 | 332 |
| Register 4: | GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 | 333 |
| Register 5: | GPIO Interrupt Event (GPIOIEV), offset 0x40C | 334 |
| Register 6: | GPIO Interrupt Mask (GPIOIM), offset 0x410 | 335 |
| Register 7: | GPIO Raw Interrupt Status (GPIORIS), offset 0x414 | 336 |
| Register 8: | GPIO Masked Interrupt Status (GPIOMIS), offset 0x418 | 337 |
| Register 9: | GPIO Interrupt Clear (GPIOICR), offset 0x41C | 339 |
| Register 10: | GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 | 340 |
| Register 11: | GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 | 342 |
| Register 12: | GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 | 343 |
| Register 13: | GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 | 344 |
| Register 14: | GPIO Open Drain Select (GPIOODR), offset 0x50C | 345 |
| Register 15: | GPIO Pull-Up Select (GPIOPUR), offset 0x510 | 346 |
| Register 16: | GPIO Pull-Down Select (GPIOPDR), offset 0x514 | 348 |
| Register 17: | GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 | 350 |
| Register 18: | GPIO Digital Enable (GPIODEN), offset 0x51C | 351 |
| Register 19: | GPIO Lock (GPIOLOCK), offset 0x520 | 353 |
| Register 20: | GPIO Commit (GPIOCR), offset 0x524 | 354 |
| Register 21: | GPIO Analog Mode Select (GPIOAMSEL), offset 0x528 | 356 |
| Register 22: | GPIO Port Control (GPIOPCTL), offset 0x52C | 358 |
| Register 23: | GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 | 360 |
| Register 24: | GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 | 361 |
| Register 25: | GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 | 362 |
| Register 26: | GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC | 363 |
| Register 27: | GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 | 364 |
| Register 28: | GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 | 365 |
| Register 29: | GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 | 366 |
| Register 30: | GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC | 367 |

| | | |
|--|---|------------|
| Register 31: | GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 | 368 |
| Register 32: | GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 | 369 |
| Register 33: | GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 | 370 |
| Register 34: | GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC | 371 |
| External Peripheral Interface (EPI) | | 372 |
| Register 1: | EPI Configuration (EPICFG), offset 0x000 | 393 |
| Register 2: | EPI Main Baud Rate (EPIBAUD), offset 0x004 | 394 |
| Register 3: | EPI SDRAM Configuration (EPISDRAMCFG), offset 0x010 | 395 |
| Register 4: | EPI Host-Bus 8 Configuration (EPIHB8CFG), offset 0x010 | 397 |
| Register 5: | EPI Host-Bus 16 Configuration (EPIHB16CFG), offset 0x010 | 400 |
| Register 6: | EPI General-Purpose Configuration (EPIGPCFG), offset 0x010 | 404 |
| Register 7: | EPI SDRAM Configuration 2 (EPISDRAMCFG2), offset 0x014 | 408 |
| Register 8: | EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2), offset 0x014 | 409 |
| Register 9: | EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2), offset 0x014 | 411 |
| Register 10: | EPI General-Purpose Configuration 2 (EPIGPCFG2), offset 0x014 | 413 |
| Register 11: | EPI Address Map (EPIADDRMAP), offset 0x01C | 414 |
| Register 12: | EPI Read Size 0 (EPIRSIZE0), offset 0x020 | 416 |
| Register 13: | EPI Read Size 1 (EPIRSIZE1), offset 0x030 | 416 |
| Register 14: | EPI Read Address 0 (EPIRADDR0), offset 0x024 | 417 |
| Register 15: | EPI Read Address 1 (EPIRADDR1), offset 0x034 | 417 |
| Register 16: | EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028 | 418 |
| Register 17: | EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038 | 418 |
| Register 18: | EPI Status (EPISTAT), offset 0x060 | 420 |
| Register 19: | EPI Read FIFO Count (EPIRFIFOCNT), offset 0x06C | 422 |
| Register 20: | EPI Read FIFO (EPIREADFIFO), offset 0x070 | 423 |
| Register 21: | EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074 | 423 |
| Register 22: | EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078 | 423 |
| Register 23: | EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C | 423 |
| Register 24: | EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080 | 423 |
| Register 25: | EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084 | 423 |
| Register 26: | EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088 | 423 |
| Register 27: | EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C | 423 |
| Register 28: | EPI FIFO Level Selects (EPIFIFOLVL), offset 0x200 | 424 |
| Register 29: | EPI Write FIFO Count (EPIWFIFOCNT), offset 0x204 | 426 |
| Register 30: | EPI Interrupt Mask (EPIIM), offset 0x210 | 427 |
| Register 31: | EPI Raw Interrupt Status (EPIRIS), offset 0x214 | 428 |
| Register 32: | EPI Masked Interrupt Status (EPIMIS), offset 0x218 | 430 |
| Register 33: | EPI Error Interrupt Status and Clear (EPIEISC), offset 0x21C | 431 |
| General-Purpose Timers | | 433 |
| Register 1: | GPTM Configuration (GPTMCFG), offset 0x000 | 448 |
| Register 2: | GPTM Timer A Mode (GPTMTAMR), offset 0x004 | 449 |
| Register 3: | GPTM Timer B Mode (GPTMTBMR), offset 0x008 | 451 |
| Register 4: | GPTM Control (GPTMCTL), offset 0x00C | 453 |
| Register 5: | GPTM Interrupt Mask (GPTMIMR), offset 0x018 | 456 |
| Register 6: | GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C | 458 |
| Register 7: | GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 | 461 |
| Register 8: | GPTM Interrupt Clear (GPTMICR), offset 0x024 | 464 |
| Register 9: | GPTM Timer A Interval Load (GPTMTAILR), offset 0x028 | 466 |

| | | |
|--|--|------------|
| Register 10: | GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C | 467 |
| Register 11: | GPTM Timer A Match (GPTMTAMATCHR), offset 0x030 | 468 |
| Register 12: | GPTM Timer B Match (GPTMTBMATCHR), offset 0x034 | 469 |
| Register 13: | GPTM Timer A Prescale (GPTMTAPR), offset 0x038 | 470 |
| Register 14: | GPTM Timer B Prescale (GPTMTBPR), offset 0x03C | 471 |
| Register 15: | GPTM Timer A (GPTMTAR), offset 0x048 | 472 |
| Register 16: | GPTM Timer B (GPTMTBR), offset 0x04C | 473 |
| Register 17: | GPTM Timer A Value (GPTMTAV), offset 0x050 | 474 |
| Register 18: | GPTM Timer B Value (GPTMTBV), offset 0x054 | 475 |
| Watchdog Timers | | 476 |
| Register 1: | Watchdog Load (WDTLOAD), offset 0x000 | 480 |
| Register 2: | Watchdog Value (WDTVALUE), offset 0x004 | 481 |
| Register 3: | Watchdog Control (WDTCTL), offset 0x008 | 482 |
| Register 4: | Watchdog Interrupt Clear (WDTICR), offset 0x00C | 484 |
| Register 5: | Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 | 485 |
| Register 6: | Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 | 486 |
| Register 7: | Watchdog Test (WDTTEST), offset 0x418 | 487 |
| Register 8: | Watchdog Lock (WDTLOCK), offset 0xC00 | 488 |
| Register 9: | Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 | 489 |
| Register 10: | Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 | 490 |
| Register 11: | Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 | 491 |
| Register 12: | Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC | 492 |
| Register 13: | Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 | 493 |
| Register 14: | Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 | 494 |
| Register 15: | Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 | 495 |
| Register 16: | Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC | 496 |
| Register 17: | Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 | 497 |
| Register 18: | Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 | 498 |
| Register 19: | Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 | 499 |
| Register 20: | Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC | 500 |
| Analog-to-Digital Converter (ADC) | | 501 |
| Register 1: | ADC Active Sample Sequencer (ADCACTSS), offset 0x000 | 520 |
| Register 2: | ADC Raw Interrupt Status (ADCRIS), offset 0x004 | 521 |
| Register 3: | ADC Interrupt Mask (ADCIM), offset 0x008 | 523 |
| Register 4: | ADC Interrupt Status and Clear (ADCISC), offset 0x00C | 525 |
| Register 5: | ADC Overflow Status (ADCOSTAT), offset 0x010 | 528 |
| Register 6: | ADC Event Multiplexer Select (ADCEMUX), offset 0x014 | 530 |
| Register 7: | ADC Underflow Status (ADCUSTAT), offset 0x018 | 532 |
| Register 8: | ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 | 533 |
| Register 9: | ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 | 535 |
| Register 10: | ADC Sample Averaging Control (ADCSAC), offset 0x030 | 537 |
| Register 11: | ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034 | 538 |
| Register 12: | ADC Control (ADCCCTL), offset 0x038 | 540 |
| Register 13: | ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 | 541 |
| Register 14: | ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 | 543 |
| Register 15: | ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 | 546 |
| Register 16: | ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 | 546 |
| Register 17: | ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 | 546 |

| | | |
|--|--|------------|
| Register 18: | ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 | 546 |
| Register 19: | ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C | 547 |
| Register 20: | ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C | 547 |
| Register 21: | ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C | 547 |
| Register 22: | ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC | 547 |
| Register 23: | ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050 | 549 |
| Register 24: | ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054 | 551 |
| Register 25: | ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 | 553 |
| Register 26: | ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 | 553 |
| Register 27: | ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 | 554 |
| Register 28: | ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 | 554 |
| Register 29: | ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070 | 556 |
| Register 30: | ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090 | 556 |
| Register 31: | ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074 | 557 |
| Register 32: | ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094 | 557 |
| Register 33: | ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 | 559 |
| Register 34: | ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 | 560 |
| Register 35: | ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0 | 561 |
| Register 36: | ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4 | 562 |
| Register 37: | ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00 | 563 |
| Register 38: | ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00 | 568 |
| Register 39: | ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04 | 568 |
| Register 40: | ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08 | 568 |
| Register 41: | ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C | 568 |
| Register 42: | ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10 | 568 |
| Register 43: | ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14 | 568 |
| Register 44: | ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18 | 568 |
| Register 45: | ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C | 568 |
| Register 46: | ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40 | 572 |
| Register 47: | ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44 | 572 |
| Register 48: | ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48 | 572 |
| Register 49: | ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C | 572 |
| Register 50: | ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50 | 572 |
| Register 51: | ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54 | 572 |
| Register 52: | ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58 | 572 |
| Register 53: | ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C | 572 |
| Universal Asynchronous Receivers/Transmitters (UARTs) | | 573 |
| Register 1: | UART Data (UARTDR), offset 0x000 | 586 |
| Register 2: | UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 | 588 |
| Register 3: | UART Flag (UARTFR), offset 0x018 | 591 |
| Register 4: | UART IrDA Low-Power Register (UARTILPR), offset 0x020 | 594 |
| Register 5: | UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 | 595 |
| Register 6: | UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 | 596 |
| Register 7: | UART Line Control (UARTLCRH), offset 0x02C | 597 |
| Register 8: | UART Control (UARTCTL), offset 0x030 | 599 |
| Register 9: | UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 | 603 |
| Register 10: | UART Interrupt Mask (UARTIM), offset 0x038 | 605 |
| Register 11: | UART Raw Interrupt Status (UARTRIS), offset 0x03C | 609 |

| | | |
|--|--|------------|
| Register 12: | UART Masked Interrupt Status (UARTMIS), offset 0x040 | 613 |
| Register 13: | UART Interrupt Clear (UARTICR), offset 0x044 | 616 |
| Register 14: | UART DMA Control (UARTDMACTL), offset 0x048 | 618 |
| Register 15: | UART LIN Control (UARTLCTL), offset 0x090 | 619 |
| Register 16: | UART LIN Snap Shot (UARTLSS), offset 0x094 | 620 |
| Register 17: | UART LIN Timer (UARTLTIM), offset 0x098 | 621 |
| Register 18: | UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 | 622 |
| Register 19: | UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 | 623 |
| Register 20: | UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 | 624 |
| Register 21: | UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC | 625 |
| Register 22: | UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 | 626 |
| Register 23: | UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 | 627 |
| Register 24: | UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 | 628 |
| Register 25: | UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC | 629 |
| Register 26: | UART PrimeCell Identification 0 (UARTPCelIID0), offset 0xFF0 | 630 |
| Register 27: | UART PrimeCell Identification 1 (UARTPCelIID1), offset 0xFF4 | 631 |
| Register 28: | UART PrimeCell Identification 2 (UARTPCelIID2), offset 0xFF8 | 632 |
| Register 29: | UART PrimeCell Identification 3 (UARTPCelIID3), offset 0xFFC | 633 |
| Synchronous Serial Interface (SSI) | | 634 |
| Register 1: | SSI Control 0 (SSICR0), offset 0x000 | 649 |
| Register 2: | SSI Control 1 (SSICR1), offset 0x004 | 651 |
| Register 3: | SSI Data (SSIDR), offset 0x008 | 653 |
| Register 4: | SSI Status (SSISR), offset 0x00C | 654 |
| Register 5: | SSI Clock Prescale (SSICPSR), offset 0x010 | 656 |
| Register 6: | SSI Interrupt Mask (SSIIM), offset 0x014 | 657 |
| Register 7: | SSI Raw Interrupt Status (SSIRIS), offset 0x018 | 658 |
| Register 8: | SSI Masked Interrupt Status (SSIMIS), offset 0x01C | 660 |
| Register 9: | SSI Interrupt Clear (SSIICR), offset 0x020 | 662 |
| Register 10: | SSI DMA Control (SSIDMACTL), offset 0x024 | 663 |
| Register 11: | SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 | 664 |
| Register 12: | SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 | 665 |
| Register 13: | SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 | 666 |
| Register 14: | SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC | 667 |
| Register 15: | SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 | 668 |
| Register 16: | SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 | 669 |
| Register 17: | SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 | 670 |
| Register 18: | SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC | 671 |
| Register 19: | SSI PrimeCell Identification 0 (SSIPCellIID0), offset 0xFF0 | 672 |
| Register 20: | SSI PrimeCell Identification 1 (SSIPCellIID1), offset 0xFF4 | 673 |
| Register 21: | SSI PrimeCell Identification 2 (SSIPCellIID2), offset 0xFF8 | 674 |
| Register 22: | SSI PrimeCell Identification 3 (SSIPCellIID3), offset 0xFFC | 675 |
| Inter-Integrated Circuit (I²C) Interface | | 676 |
| Register 1: | I ² C Master Slave Address (I2CMSA), offset 0x000 | 692 |
| Register 2: | I ² C Master Control/Status (I2CMCS), offset 0x004 | 693 |
| Register 3: | I ² C Master Data (I2CMDR), offset 0x008 | 698 |
| Register 4: | I ² C Master Timer Period (I2CMTPR), offset 0x00C | 699 |
| Register 5: | I ² C Master Interrupt Mask (I2CMIMR), offset 0x010 | 700 |
| Register 6: | I ² C Master Raw Interrupt Status (I2CMRIS), offset 0x014 | 701 |

| | | |
|--|--|------------|
| Register 7: | I ² C Master Masked Interrupt Status (I2CMMIS), offset 0x018 | 702 |
| Register 8: | I ² C Master Interrupt Clear (I2CMICR), offset 0x01C | 703 |
| Register 9: | I ² C Master Configuration (I2CMCR), offset 0x020 | 704 |
| Register 10: | I ² C Slave Own Address (I2CSOAR), offset 0x000 | 705 |
| Register 11: | I ² C Slave Control/Status (I2CSCSR), offset 0x004 | 706 |
| Register 12: | I ² C Slave Data (I2CSDR), offset 0x008 | 708 |
| Register 13: | I ² C Slave Interrupt Mask (I2CSIMR), offset 0x00C | 709 |
| Register 14: | I ² C Slave Raw Interrupt Status (I2CSRIS), offset 0x010 | 710 |
| Register 15: | I ² C Slave Masked Interrupt Status (I2CSMIS), offset 0x014 | 711 |
| Register 16: | I ² C Slave Interrupt Clear (I2CSICR), offset 0x018 | 712 |
| Inter-Integrated Circuit Sound (I²S) Interface | | 713 |
| Register 1: | I ² S Transmit FIFO Data (I2STXFIFO), offset 0x000 | 726 |
| Register 2: | I ² S Transmit FIFO Configuration (I2STXFIFOCFG), offset 0x004 | 727 |
| Register 3: | I ² S Transmit Module Configuration (I2STXCFG), offset 0x008 | 728 |
| Register 4: | I ² S Transmit FIFO Limit (I2STXLIMIT), offset 0x00C | 730 |
| Register 5: | I ² S Transmit Interrupt Status and Mask (I2STXISM), offset 0x010 | 731 |
| Register 6: | I ² S Transmit FIFO Level (I2STXLEV), offset 0x018 | 732 |
| Register 7: | I ² S Receive FIFO Data (I2SRXFIFO), offset 0x800 | 733 |
| Register 8: | I ² S Receive FIFO Configuration (I2SRXFIFOCFG), offset 0x804 | 734 |
| Register 9: | I ² S Receive Module Configuration (I2SRXCFG), offset 0x808 | 735 |
| Register 10: | I ² S Receive FIFO Limit (I2SRXLIMIT), offset 0x80C | 738 |
| Register 11: | I ² S Receive Interrupt Status and Mask (I2SRXISM), offset 0x810 | 739 |
| Register 12: | I ² S Receive FIFO Level (I2SRXLEV), offset 0x818 | 740 |
| Register 13: | I ² S Module Configuration (I2SCFG), offset 0xC00 | 741 |
| Register 14: | I ² S Interrupt Mask (I2SIM), offset 0xC10 | 743 |
| Register 15: | I ² S Raw Interrupt Status (I2SRIS), offset 0xC14 | 745 |
| Register 16: | I ² S Masked Interrupt Status (I2SMIS), offset 0xC18 | 747 |
| Register 17: | I ² S Interrupt Clear (I2SIC), offset 0xC1C | 749 |
| Controller Area Network (CAN) Module | | 750 |
| Register 1: | CAN Control (CANCTL), offset 0x000 | 770 |
| Register 2: | CAN Status (CANSTS), offset 0x004 | 772 |
| Register 3: | CAN Error Counter (CANERR), offset 0x008 | 775 |
| Register 4: | CAN Bit Timing (CANBIT), offset 0x00C | 776 |
| Register 5: | CAN Interrupt (CANINT), offset 0x010 | 778 |
| Register 6: | CAN Test (CANTST), offset 0x014 | 779 |
| Register 7: | CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018 | 781 |
| Register 8: | CAN IF1 Command Request (CANIF1CRQ), offset 0x020 | 782 |
| Register 9: | CAN IF2 Command Request (CANIF2CRQ), offset 0x080 | 782 |
| Register 10: | CAN IF1 Command Mask (CANIF1CMSK), offset 0x024 | 783 |
| Register 11: | CAN IF2 Command Mask (CANIF2CMSK), offset 0x084 | 783 |
| Register 12: | CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028 | 786 |
| Register 13: | CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088 | 786 |
| Register 14: | CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C | 787 |
| Register 15: | CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C | 787 |
| Register 16: | CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030 | 789 |
| Register 17: | CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090 | 789 |

| | | |
|----------------------------------|--|------------|
| Register 18: | CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034 | 790 |
| Register 19: | CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094 | 790 |
| Register 20: | CAN IF1 Message Control (CANIF1MCTL), offset 0x038 | 792 |
| Register 21: | CAN IF2 Message Control (CANIF2MCTL), offset 0x098 | 792 |
| Register 22: | CAN IF1 Data A1 (CANIF1DA1), offset 0x03C | 795 |
| Register 23: | CAN IF1 Data A2 (CANIF1DA2), offset 0x040 | 795 |
| Register 24: | CAN IF1 Data B1 (CANIF1DB1), offset 0x044 | 795 |
| Register 25: | CAN IF1 Data B2 (CANIF1DB2), offset 0x048 | 795 |
| Register 26: | CAN IF2 Data A1 (CANIF2DA1), offset 0x09C | 795 |
| Register 27: | CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0 | 795 |
| Register 28: | CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4 | 795 |
| Register 29: | CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8 | 795 |
| Register 30: | CAN Transmission Request 1 (CANTXRQ1), offset 0x100 | 796 |
| Register 31: | CAN Transmission Request 2 (CANTXRQ2), offset 0x104 | 796 |
| Register 32: | CAN New Data 1 (CANNWDA1), offset 0x120 | 797 |
| Register 33: | CAN New Data 2 (CANNWDA2), offset 0x124 | 797 |
| Register 34: | CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140 | 798 |
| Register 35: | CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144 | 798 |
| Register 36: | CAN Message 1 Valid (CANMSG1VAL), offset 0x160 | 799 |
| Register 37: | CAN Message 2 Valid (CANMSG2VAL), offset 0x164 | 799 |
| Ethernet Controller | | 800 |
| Register 1: | Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK), offset 0x000 | 813 |
| Register 2: | Ethernet MAC Interrupt Mask (MACIM), offset 0x004 | 816 |
| Register 3: | Ethernet MAC Receive Control (MACRCTL), offset 0x008 | 818 |
| Register 4: | Ethernet MAC Transmit Control (MACTCTL), offset 0x00C | 820 |
| Register 5: | Ethernet MAC Data (MACDATA), offset 0x010 | 822 |
| Register 6: | Ethernet MAC Individual Address 0 (MACIA0), offset 0x014 | 824 |
| Register 7: | Ethernet MAC Individual Address 1 (MACIA1), offset 0x018 | 825 |
| Register 8: | Ethernet MAC Threshold (MACTHR), offset 0x01C | 826 |
| Register 9: | Ethernet MAC Management Control (MACMCTL), offset 0x020 | 828 |
| Register 10: | Ethernet MAC Management Divider (MACMDV), offset 0x024 | 830 |
| Register 11: | Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C | 831 |
| Register 12: | Ethernet MAC Management Receive Data (MACMRXD), offset 0x030 | 832 |
| Register 13: | Ethernet MAC Number of Packets (MACNP), offset 0x034 | 833 |
| Register 14: | Ethernet MAC Transmission Request (MACTR), offset 0x038 | 834 |
| Register 15: | Ethernet MAC LED Encoding (MACLED), offset 0x040 | 835 |
| Register 16: | Ethernet PHY MDIX (MDIX), offset 0x044 | 836 |
| Register 17: | Ethernet PHY Management Register 0 – Control (MR0), address 0x00 | 837 |
| Register 18: | Ethernet PHY Management Register 1 – Status (MR1), address 0x01 | 839 |
| Register 19: | Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02 | 841 |
| Register 20: | Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03 | 842 |
| Register 21: | Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04 | 843 |
| Register 22: | Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05 | 845 |
| Register 23: | Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06 | 847 |
| Register 24: | Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10 | 848 |

| | | |
|--|---|-----|
| Register 25: | Ethernet PHY Management Register 17 – Mode Control/Status (MR17), address 0x11 | 849 |
| Register 26: | Ethernet PHY Management Register 27 – Special Control/Status (MR27), address 0x1B | 851 |
| Register 27: | Ethernet PHY Management Register 29 – Interrupt Status (MR29), address 0x1D | 852 |
| Register 28: | Ethernet PHY Management Register 30 – Interrupt Mask (MR30), address 0x1E | 854 |
| Register 29: | Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31), address 0x1F | 856 |
| Universal Serial Bus (USB) Controller | 857 | |
| Register 1: | USB Device Functional Address (USBFADDR), offset 0x000 | 884 |
| Register 2: | USB Power (USBPOWER), offset 0x001 | 885 |
| Register 3: | USB Transmit Interrupt Status (USBTXIS), offset 0x002 | 888 |
| Register 4: | USB Receive Interrupt Status (USBRXIS), offset 0x004 | 890 |
| Register 5: | USB Transmit Interrupt Enable (USBTXIE), offset 0x006 | 892 |
| Register 6: | USB Receive Interrupt Enable (USBRXIE), offset 0x008 | 894 |
| Register 7: | USB General Interrupt Status (USBIS), offset 0x00A | 896 |
| Register 8: | USB Interrupt Enable (USBIE), offset 0x00B | 899 |
| Register 9: | USB Frame Value (USBFRAME), offset 0x00C | 902 |
| Register 10: | USB Endpoint Index (USBEPIDX), offset 0x00E | 903 |
| Register 11: | USB Test Mode (USBTEST), offset 0x00F | 904 |
| Register 12: | USB FIFO Endpoint 0 (USBFIFO0), offset 0x020 | 906 |
| Register 13: | USB FIFO Endpoint 1 (USBFIFO1), offset 0x024 | 906 |
| Register 14: | USB FIFO Endpoint 2 (USBFIFO2), offset 0x028 | 906 |
| Register 15: | USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C | 906 |
| Register 16: | USB FIFO Endpoint 4 (USBFIFO4), offset 0x030 | 906 |
| Register 17: | USB FIFO Endpoint 5 (USBFIFO5), offset 0x034 | 906 |
| Register 18: | USB FIFO Endpoint 6 (USBFIFO6), offset 0x038 | 906 |
| Register 19: | USB FIFO Endpoint 7 (USBFIFO7), offset 0x03C | 906 |
| Register 20: | USB FIFO Endpoint 8 (USBFIFO8), offset 0x040 | 906 |
| Register 21: | USB FIFO Endpoint 9 (USBFIFO9), offset 0x044 | 906 |
| Register 22: | USB FIFO Endpoint 10 (USBFIFO10), offset 0x048 | 906 |
| Register 23: | USB FIFO Endpoint 11 (USBFIFO11), offset 0x04C | 906 |
| Register 24: | USB FIFO Endpoint 12 (USBFIFO12), offset 0x050 | 906 |
| Register 25: | USB FIFO Endpoint 13 (USBFIFO13), offset 0x054 | 906 |
| Register 26: | USB FIFO Endpoint 14 (USBFIFO14), offset 0x058 | 906 |
| Register 27: | USB FIFO Endpoint 15 (USBFIFO15), offset 0x05C | 906 |
| Register 28: | USB Device Control (USBDEVCTL), offset 0x060 | 908 |
| Register 29: | USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062 | 910 |
| Register 30: | USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063 | 910 |
| Register 31: | USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064 | 911 |
| Register 32: | USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066 | 911 |
| Register 33: | USB Connect Timing (USBCONTIM), offset 0x07A | 912 |
| Register 34: | USB OTG VBUS Pulse Timing (USBVPLEN), offset 0x07B | 913 |
| Register 35: | USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D | 914 |
| Register 36: | USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E | 915 |
| Register 37: | USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080 | 916 |
| Register 38: | USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088 | 916 |
| Register 39: | USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090 | 916 |
| Register 40: | USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098 | 916 |

| | | |
|--------------|---|-----|
| Register 41: | USB Transmit Functional Address Endpoint 4 (USBTXFUNCADDR4), offset 0x0A0 | 916 |
| Register 42: | USB Transmit Functional Address Endpoint 5 (USBTXFUNCADDR5), offset 0x0A8 | 916 |
| Register 43: | USB Transmit Functional Address Endpoint 6 (USBTXFUNCADDR6), offset 0x0B0 | 916 |
| Register 44: | USB Transmit Functional Address Endpoint 7 (USBTXFUNCADDR7), offset 0x0B8 | 916 |
| Register 45: | USB Transmit Functional Address Endpoint 8 (USBTXFUNCADDR8), offset 0x0C0 | 916 |
| Register 46: | USB Transmit Functional Address Endpoint 9 (USBTXFUNCADDR9), offset 0x0C8 | 916 |
| Register 47: | USB Transmit Functional Address Endpoint 10 (USBTXFUNCADDR10), offset 0x0D0 | 916 |
| Register 48: | USB Transmit Functional Address Endpoint 11 (USBTXFUNCADDR11), offset 0x0D8 | 916 |
| Register 49: | USB Transmit Functional Address Endpoint 12 (USBTXFUNCADDR12), offset 0x0E0 | 916 |
| Register 50: | USB Transmit Functional Address Endpoint 13 (USBTXFUNCADDR13), offset 0x0E8 | 916 |
| Register 51: | USB Transmit Functional Address Endpoint 14 (USBTXFUNCADDR14), offset 0x0F0 | 916 |
| Register 52: | USB Transmit Functional Address Endpoint 15 (USBTXFUNCADDR15), offset 0x0F8 | 916 |
| Register 53: | USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082 | 918 |
| Register 54: | USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A | 918 |
| Register 55: | USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092 | 918 |
| Register 56: | USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A | 918 |
| Register 57: | USB Transmit Hub Address Endpoint 4 (USBTXHUBADDR4), offset 0x0A2 | 918 |
| Register 58: | USB Transmit Hub Address Endpoint 5 (USBTXHUBADDR5), offset 0x0AA | 918 |
| Register 59: | USB Transmit Hub Address Endpoint 6 (USBTXHUBADDR6), offset 0x0B2 | 918 |
| Register 60: | USB Transmit Hub Address Endpoint 7 (USBTXHUBADDR7), offset 0x0BA | 918 |
| Register 61: | USB Transmit Hub Address Endpoint 8 (USBTXHUBADDR8), offset 0x0C2 | 918 |
| Register 62: | USB Transmit Hub Address Endpoint 9 (USBTXHUBADDR9), offset 0x0CA | 918 |
| Register 63: | USB Transmit Hub Address Endpoint 10 (USBTXHUBADDR10), offset 0x0D2 | 918 |
| Register 64: | USB Transmit Hub Address Endpoint 11 (USBTXHUBADDR11), offset 0x0DA | 918 |
| Register 65: | USB Transmit Hub Address Endpoint 12 (USBTXHUBADDR12), offset 0x0E2 | 918 |
| Register 66: | USB Transmit Hub Address Endpoint 13 (USBTXHUBADDR13), offset 0x0EA | 918 |
| Register 67: | USB Transmit Hub Address Endpoint 14 (USBTXHUBADDR14), offset 0x0F2 | 918 |
| Register 68: | USB Transmit Hub Address Endpoint 15 (USBTXHUBADDR15), offset 0x0FA | 918 |
| Register 69: | USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083 | 920 |
| Register 70: | USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B | 920 |
| Register 71: | USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093 | 920 |
| Register 72: | USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B | 920 |
| Register 73: | USB Transmit Hub Port Endpoint 4 (USBTXHUBPORT4), offset 0x0A3 | 920 |
| Register 74: | USB Transmit Hub Port Endpoint 5 (USBTXHUBPORT5), offset 0x0AB | 920 |
| Register 75: | USB Transmit Hub Port Endpoint 6 (USBTXHUBPORT6), offset 0x0B3 | 920 |
| Register 76: | USB Transmit Hub Port Endpoint 7 (USBTXHUBPORT7), offset 0x0BB | 920 |
| Register 77: | USB Transmit Hub Port Endpoint 8 (USBTXHUBPORT8), offset 0x0C3 | 920 |
| Register 78: | USB Transmit Hub Port Endpoint 9 (USBTXHUBPORT9), offset 0x0CB | 920 |
| Register 79: | USB Transmit Hub Port Endpoint 10 (USBTXHUBPORT10), offset 0x0D3 | 920 |
| Register 80: | USB Transmit Hub Port Endpoint 11 (USBTXHUBPORT11), offset 0x0DB | 920 |
| Register 81: | USB Transmit Hub Port Endpoint 12 (USBTXHUBPORT12), offset 0x0E3 | 920 |
| Register 82: | USB Transmit Hub Port Endpoint 13 (USBTXHUBPORT13), offset 0x0EB | 920 |
| Register 83: | USB Transmit Hub Port Endpoint 14 (USBTXHUBPORT14), offset 0x0F3 | 920 |
| Register 84: | USB Transmit Hub Port Endpoint 15 (USBTXHUBPORT15), offset 0x0FB | 920 |
| Register 85: | USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1), offset 0x08C | 922 |
| Register 86: | USB Receive Functional Address Endpoint 2 (USBRXFUNCADDR2), offset 0x094 | 922 |
| Register 87: | USB Receive Functional Address Endpoint 3 (USBRXFUNCADDR3), offset 0x09C | 922 |
| Register 88: | USB Receive Functional Address Endpoint 4 (USBRXFUNCADDR4), offset 0x0A4 | 922 |

| | | |
|---------------|--|-----|
| Register 89: | USB Receive Functional Address Endpoint 5 (USBRXFUNCADDR5), offset 0x0AC | 922 |
| Register 90: | USB Receive Functional Address Endpoint 6 (USBRXFUNCADDR6), offset 0x0B4 | 922 |
| Register 91: | USB Receive Functional Address Endpoint 7 (USBRXFUNCADDR7), offset 0x0BC | 922 |
| Register 92: | USB Receive Functional Address Endpoint 8 (USBRXFUNCADDR8), offset 0x0C4 | 922 |
| Register 93: | USB Receive Functional Address Endpoint 9 (USBRXFUNCADDR9), offset 0x0CC | 922 |
| Register 94: | USB Receive Functional Address Endpoint 10 (USBRXFUNCADDR10), offset 0x0D4 | 922 |
| Register 95: | USB Receive Functional Address Endpoint 11 (USBRXFUNCADDR11), offset 0x0DC | 922 |
| Register 96: | USB Receive Functional Address Endpoint 12 (USBRXFUNCADDR12), offset 0x0E4 | 922 |
| Register 97: | USB Receive Functional Address Endpoint 13 (USBRXFUNCADDR13), offset 0x0EC | 922 |
| Register 98: | USB Receive Functional Address Endpoint 14 (USBRXFUNCADDR14), offset 0x0F4 | 922 |
| Register 99: | USB Receive Functional Address Endpoint 15 (USBRXFUNCADDR15), offset 0x0FC | 922 |
| Register 100: | USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E | 924 |
| Register 101: | USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096 | 924 |
| Register 102: | USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E | 924 |
| Register 103: | USB Receive Hub Address Endpoint 4 (USBRXHUBADDR4), offset 0x0A6 | 924 |
| Register 104: | USB Receive Hub Address Endpoint 5 (USBRXHUBADDR5), offset 0x0AE | 924 |
| Register 105: | USB Receive Hub Address Endpoint 6 (USBRXHUBADDR6), offset 0x0B6 | 924 |
| Register 106: | USB Receive Hub Address Endpoint 7 (USBRXHUBADDR7), offset 0x0BE | 924 |
| Register 107: | USB Receive Hub Address Endpoint 8 (USBRXHUBADDR8), offset 0x0C6 | 924 |
| Register 108: | USB Receive Hub Address Endpoint 9 (USBRXHUBADDR9), offset 0x0CE | 924 |
| Register 109: | USB Receive Hub Address Endpoint 10 (USBRXHUBADDR10), offset 0x0D6 | 924 |
| Register 110: | USB Receive Hub Address Endpoint 11 (USBRXHUBADDR11), offset 0x0DE | 924 |
| Register 111: | USB Receive Hub Address Endpoint 12 (USBRXHUBADDR12), offset 0x0E6 | 924 |
| Register 112: | USB Receive Hub Address Endpoint 13 (USBRXHUBADDR13), offset 0x0EE | 924 |
| Register 113: | USB Receive Hub Address Endpoint 14 (USBRXHUBADDR14), offset 0x0F6 | 924 |
| Register 114: | USB Receive Hub Address Endpoint 15 (USBRXHUBADDR15), offset 0x0FE | 924 |
| Register 115: | USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F | 926 |
| Register 116: | USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097 | 926 |
| Register 117: | USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F | 926 |
| Register 118: | USB Receive Hub Port Endpoint 4 (USBRXHUBPORT4), offset 0x0A7 | 926 |
| Register 119: | USB Receive Hub Port Endpoint 5 (USBRXHUBPORT5), offset 0x0AF | 926 |
| Register 120: | USB Receive Hub Port Endpoint 6 (USBRXHUBPORT6), offset 0x0B7 | 926 |
| Register 121: | USB Receive Hub Port Endpoint 7 (USBRXHUBPORT7), offset 0x0BF | 926 |
| Register 122: | USB Receive Hub Port Endpoint 8 (USBRXHUBPORT8), offset 0x0C7 | 926 |
| Register 123: | USB Receive Hub Port Endpoint 9 (USBRXHUBPORT9), offset 0x0CF | 926 |
| Register 124: | USB Receive Hub Port Endpoint 10 (USBRXHUBPORT10), offset 0x0D7 | 926 |
| Register 125: | USB Receive Hub Port Endpoint 11 (USBRXHUBPORT11), offset 0x0DF | 926 |
| Register 126: | USB Receive Hub Port Endpoint 12 (USBRXHUBPORT12), offset 0x0E7 | 926 |
| Register 127: | USB Receive Hub Port Endpoint 13 (USBRXHUBPORT13), offset 0x0EF | 926 |
| Register 128: | USB Receive Hub Port Endpoint 14 (USBRXHUBPORT14), offset 0x0F7 | 926 |
| Register 129: | USB Receive Hub Port Endpoint 15 (USBRXHUBPORT15), offset 0x0FF | 926 |
| Register 130: | USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110 | 928 |
| Register 131: | USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120 | 928 |
| Register 132: | USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130 | 928 |
| Register 133: | USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140 | 928 |
| Register 134: | USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150 | 928 |
| Register 135: | USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160 | 928 |
| Register 136: | USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170 | 928 |

| | |
|--|-----|
| Register 137: USB Maximum Transmit Data Endpoint 8 (USBTXMAXP8), offset 0x180 | 928 |
| Register 138: USB Maximum Transmit Data Endpoint 9 (USBTXMAXP9), offset 0x190 | 928 |
| Register 139: USB Maximum Transmit Data Endpoint 10 (USBTXMAXP10), offset 0x1A0 | 928 |
| Register 140: USB Maximum Transmit Data Endpoint 11 (USBTXMAXP11), offset 0x1B0 | 928 |
| Register 141: USB Maximum Transmit Data Endpoint 12 (USBTXMAXP12), offset 0x1C0 | 928 |
| Register 142: USB Maximum Transmit Data Endpoint 13 (USBTXMAXP13), offset 0x1D0 | 928 |
| Register 143: USB Maximum Transmit Data Endpoint 14 (USBTXMAXP14), offset 0x1E0 | 928 |
| Register 144: USB Maximum Transmit Data Endpoint 15 (USBTXMAXP15), offset 0x1F0 | 928 |
| Register 145: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102 | 930 |
| Register 146: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103 | 934 |
| Register 147: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108 | 936 |
| Register 148: USB Type Endpoint 0 (USBTYPE0), offset 0x10A | 937 |
| Register 149: USB NAK Limit (USBNAKLMT), offset 0x10B | 938 |
| Register 150: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112 | 939 |
| Register 151: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122 | 939 |
| Register 152: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132 | 939 |
| Register 153: USB Transmit Control and Status Endpoint 4 Low (USBTXCSRL4), offset 0x142 | 939 |
| Register 154: USB Transmit Control and Status Endpoint 5 Low (USBTXCSRL5), offset 0x152 | 939 |
| Register 155: USB Transmit Control and Status Endpoint 6 Low (USBTXCSRL6), offset 0x162 | 939 |
| Register 156: USB Transmit Control and Status Endpoint 7 Low (USBTXCSRL7), offset 0x172 | 939 |
| Register 157: USB Transmit Control and Status Endpoint 8 Low (USBTXCSRL8), offset 0x182 | 939 |
| Register 158: USB Transmit Control and Status Endpoint 9 Low (USBTXCSRL9), offset 0x192 | 939 |
| Register 159: USB Transmit Control and Status Endpoint 10 Low (USBTXCSRL10), offset 0x1A2 | 939 |
| Register 160: USB Transmit Control and Status Endpoint 11 Low (USBTXCSRL11), offset 0x1B2 | 939 |
| Register 161: USB Transmit Control and Status Endpoint 12 Low (USBTXCSRL12), offset 0x1C2 | 939 |
| Register 162: USB Transmit Control and Status Endpoint 13 Low (USBTXCSRL13), offset 0x1D2 | 939 |
| Register 163: USB Transmit Control and Status Endpoint 14 Low (USBTXCSRL14), offset 0x1E2 | 939 |
| Register 164: USB Transmit Control and Status Endpoint 15 Low (USBTXCSRL15), offset 0x1F2 | 939 |
| Register 165: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113 | 944 |
| Register 166: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123 | 944 |
| Register 167: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133 | 944 |
| Register 168: USB Transmit Control and Status Endpoint 4 High (USBTXCSRH4), offset 0x143 | 944 |
| Register 169: USB Transmit Control and Status Endpoint 5 High (USBTXCSRH5), offset 0x153 | 944 |
| Register 170: USB Transmit Control and Status Endpoint 6 High (USBTXCSRH6), offset 0x163 | 944 |
| Register 171: USB Transmit Control and Status Endpoint 7 High (USBTXCSRH7), offset 0x173 | 944 |
| Register 172: USB Transmit Control and Status Endpoint 8 High (USBTXCSRH8), offset 0x183 | 944 |
| Register 173: USB Transmit Control and Status Endpoint 9 High (USBTXCSRH9), offset 0x193 | 944 |
| Register 174: USB Transmit Control and Status Endpoint 10 High (USBTXCSRH10), offset 0x1A3 | 944 |
| Register 175: USB Transmit Control and Status Endpoint 11 High (USBTXCSRH11), offset 0x1B3 | 944 |
| Register 176: USB Transmit Control and Status Endpoint 12 High (USBTXCSRH12), offset 0x1C3 | 944 |
| Register 177: USB Transmit Control and Status Endpoint 13 High (USBTXCSRH13), offset 0x1D3 | 944 |
| Register 178: USB Transmit Control and Status Endpoint 14 High (USBTXCSRH14), offset 0x1E3 | 944 |
| Register 179: USB Transmit Control and Status Endpoint 15 High (USBTXCSRH15), offset 0x1F3 | 944 |
| Register 180: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114 | 948 |
| Register 181: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124 | 948 |
| Register 182: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134 | 948 |
| Register 183: USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144 | 948 |
| Register 184: USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154 | 948 |

| | |
|---|-----|
| Register 185: USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164 | 948 |
| Register 186: USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174 | 948 |
| Register 187: USB Maximum Receive Data Endpoint 8 (USBRXMAXP8), offset 0x184 | 948 |
| Register 188: USB Maximum Receive Data Endpoint 9 (USBRXMAXP9), offset 0x194 | 948 |
| Register 189: USB Maximum Receive Data Endpoint 10 (USBRXMAXP10), offset 0x1A4 | 948 |
| Register 190: USB Maximum Receive Data Endpoint 11 (USBRXMAXP11), offset 0x1B4 | 948 |
| Register 191: USB Maximum Receive Data Endpoint 12 (USBRXMAXP12), offset 0x1C4 | 948 |
| Register 192: USB Maximum Receive Data Endpoint 13 (USBRXMAXP13), offset 0x1D4 | 948 |
| Register 193: USB Maximum Receive Data Endpoint 14 (USBRXMAXP14), offset 0x1E4 | 948 |
| Register 194: USB Maximum Receive Data Endpoint 15 (USBRXMAXP15), offset 0x1F4 | 948 |
| Register 195: USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1), offset 0x116 | 950 |
| Register 196: USB Receive Control and Status Endpoint 2 Low (USBRXCSRL2), offset 0x126 | 950 |
| Register 197: USB Receive Control and Status Endpoint 3 Low (USBRXCSRL3), offset 0x136 | 950 |
| Register 198: USB Receive Control and Status Endpoint 4 Low (USBRXCSRL4), offset 0x146 | 950 |
| Register 199: USB Receive Control and Status Endpoint 5 Low (USBRXCSRL5), offset 0x156 | 950 |
| Register 200: USB Receive Control and Status Endpoint 6 Low (USBRXCSRL6), offset 0x166 | 950 |
| Register 201: USB Receive Control and Status Endpoint 7 Low (USBRXCSRL7), offset 0x176 | 950 |
| Register 202: USB Receive Control and Status Endpoint 8 Low (USBRXCSRL8), offset 0x186 | 950 |
| Register 203: USB Receive Control and Status Endpoint 9 Low (USBRXCSRL9), offset 0x196 | 950 |
| Register 204: USB Receive Control and Status Endpoint 10 Low (USBRXCSRL10), offset 0x1A6 | 950 |
| Register 205: USB Receive Control and Status Endpoint 11 Low (USBRXCSRL11), offset 0x1B6 | 950 |
| Register 206: USB Receive Control and Status Endpoint 12 Low (USBRXCSRL12), offset 0x1C6 | 950 |
| Register 207: USB Receive Control and Status Endpoint 13 Low (USBRXCSRL13), offset 0x1D6 | 950 |
| Register 208: USB Receive Control and Status Endpoint 14 Low (USBRXCSRL14), offset 0x1E6 | 950 |
| Register 209: USB Receive Control and Status Endpoint 15 Low (USBRXCSRL15), offset 0x1F6 | 950 |
| Register 210: USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117 | 955 |
| Register 211: USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127 | 955 |
| Register 212: USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137 | 955 |
| Register 213: USB Receive Control and Status Endpoint 4 High (USBRXCSRH4), offset 0x147 | 955 |
| Register 214: USB Receive Control and Status Endpoint 5 High (USBRXCSRH5), offset 0x157 | 955 |
| Register 215: USB Receive Control and Status Endpoint 6 High (USBRXCSRH6), offset 0x167 | 955 |
| Register 216: USB Receive Control and Status Endpoint 7 High (USBRXCSRH7), offset 0x177 | 955 |
| Register 217: USB Receive Control and Status Endpoint 8 High (USBRXCSRH8), offset 0x187 | 955 |
| Register 218: USB Receive Control and Status Endpoint 9 High (USBRXCSRH9), offset 0x197 | 955 |
| Register 219: USB Receive Control and Status Endpoint 10 High (USBRXCSRH10), offset 0x1A7 | 955 |
| Register 220: USB Receive Control and Status Endpoint 11 High (USBRXCSRH11), offset 0x1B7 | 955 |
| Register 221: USB Receive Control and Status Endpoint 12 High (USBRXCSRH12), offset 0x1C7 | 955 |
| Register 222: USB Receive Control and Status Endpoint 13 High (USBRXCSRH13), offset 0x1D7 | 955 |
| Register 223: USB Receive Control and Status Endpoint 14 High (USBRXCSRH14), offset 0x1E7 | 955 |
| Register 224: USB Receive Control and Status Endpoint 15 High (USBRXCSRH15), offset 0x1F7 | 955 |
| Register 225: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118 | 960 |
| Register 226: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128 | 960 |
| Register 227: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138 | 960 |
| Register 228: USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148 | 960 |
| Register 229: USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158 | 960 |
| Register 230: USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168 | 960 |
| Register 231: USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178 | 960 |
| Register 232: USB Receive Byte Count Endpoint 8 (USBRXCOUNT8), offset 0x188 | 960 |

| | |
|--|-----|
| Register 233: USB Receive Byte Count Endpoint 9 (USBRXCOUNT9), offset 0x198 | 960 |
| Register 234: USB Receive Byte Count Endpoint 10 (USBRXCOUNT10), offset 0x1A8 | 960 |
| Register 235: USB Receive Byte Count Endpoint 11 (USBRXCOUNT11), offset 0x1B8 | 960 |
| Register 236: USB Receive Byte Count Endpoint 12 (USBRXCOUNT12), offset 0x1C8 | 960 |
| Register 237: USB Receive Byte Count Endpoint 13 (USBRXCOUNT13), offset 0x1D8 | 960 |
| Register 238: USB Receive Byte Count Endpoint 14 (USBRXCOUNT14), offset 0x1E8 | 960 |
| Register 239: USB Receive Byte Count Endpoint 15 (USBRXCOUNT15), offset 0x1F8 | 960 |
| Register 240: USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A | 962 |
| Register 241: USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A | 962 |
| Register 242: USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A | 962 |
| Register 243: USB Host Transmit Configure Type Endpoint 4 (USBTXTYPE4), offset 0x14A | 962 |
| Register 244: USB Host Transmit Configure Type Endpoint 5 (USBTXTYPE5), offset 0x15A | 962 |
| Register 245: USB Host Transmit Configure Type Endpoint 6 (USBTXTYPE6), offset 0x16A | 962 |
| Register 246: USB Host Transmit Configure Type Endpoint 7 (USBTXTYPE7), offset 0x17A | 962 |
| Register 247: USB Host Transmit Configure Type Endpoint 8 (USBTXTYPE8), offset 0x18A | 962 |
| Register 248: USB Host Transmit Configure Type Endpoint 9 (USBTXTYPE9), offset 0x19A | 962 |
| Register 249: USB Host Transmit Configure Type Endpoint 10 (USBTXTYPE10), offset 0x1AA | 962 |
| Register 250: USB Host Transmit Configure Type Endpoint 11 (USBTXTYPE11), offset 0x1BA | 962 |
| Register 251: USB Host Transmit Configure Type Endpoint 12 (USBTXTYPE12), offset 0x1CA | 962 |
| Register 252: USB Host Transmit Configure Type Endpoint 13 (USBTXTYPE13), offset 0x1DA | 962 |
| Register 253: USB Host Transmit Configure Type Endpoint 14 (USBTXTYPE14), offset 0x1EA | 962 |
| Register 254: USB Host Transmit Configure Type Endpoint 15 (USBTXTYPE15), offset 0x1FA | 962 |
| Register 255: USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B | 964 |
| Register 256: USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B | 964 |
| Register 257: USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B | 964 |
| Register 258: USB Host Transmit Interval Endpoint 4 (USBTXINTERVAL4), offset 0x14B | 964 |
| Register 259: USB Host Transmit Interval Endpoint 5 (USBTXINTERVAL5), offset 0x15B | 964 |
| Register 260: USB Host Transmit Interval Endpoint 6 (USBTXINTERVAL6), offset 0x16B | 964 |
| Register 261: USB Host Transmit Interval Endpoint 7 (USBTXINTERVAL7), offset 0x17B | 964 |
| Register 262: USB Host Transmit Interval Endpoint 8 (USBTXINTERVAL8), offset 0x18B | 964 |
| Register 263: USB Host Transmit Interval Endpoint 9 (USBTXINTERVAL9), offset 0x19B | 964 |
| Register 264: USB Host Transmit Interval Endpoint 10 (USBTXINTERVAL10), offset 0x1AB | 964 |
| Register 265: USB Host Transmit Interval Endpoint 11 (USBTXINTERVAL11), offset 0x1BB | 964 |
| Register 266: USB Host Transmit Interval Endpoint 12 (USBTXINTERVAL12), offset 0x1CB | 964 |
| Register 267: USB Host Transmit Interval Endpoint 13 (USBTXINTERVAL13), offset 0x1DB | 964 |
| Register 268: USB Host Transmit Interval Endpoint 14 (USBTXINTERVAL14), offset 0x1EB | 964 |
| Register 269: USB Host Transmit Interval Endpoint 15 (USBTXINTERVAL15), offset 0x1FB | 964 |
| Register 270: USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C | 966 |
| Register 271: USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C | 966 |
| Register 272: USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C | 966 |
| Register 273: USB Host Configure Receive Type Endpoint 4 (USBRXTYPE4), offset 0x14C | 966 |
| Register 274: USB Host Configure Receive Type Endpoint 5 (USBRXTYPE5), offset 0x15C | 966 |
| Register 275: USB Host Configure Receive Type Endpoint 6 (USBRXTYPE6), offset 0x16C | 966 |
| Register 276: USB Host Configure Receive Type Endpoint 7 (USBRXTYPE7), offset 0x17C | 966 |
| Register 277: USB Host Configure Receive Type Endpoint 8 (USBRXTYPE8), offset 0x18C | 966 |
| Register 278: USB Host Configure Receive Type Endpoint 9 (USBRXTYPE9), offset 0x19C | 966 |
| Register 279: USB Host Configure Receive Type Endpoint 10 (USBRXTYPE10), offset 0x1AC | 966 |
| Register 280: USB Host Configure Receive Type Endpoint 11 (USBRXTYPE11), offset 0x1BC | 966 |

| | |
|---|-----|
| Register 281: USB Host Configure Receive Type Endpoint 12 (USBRXTYPE12), offset 0x1CC | 966 |
| Register 282: USB Host Configure Receive Type Endpoint 13 (USBRXTYPE13), offset 0x1DC | 966 |
| Register 283: USB Host Configure Receive Type Endpoint 14 (USBRXTYPE14), offset 0x1EC | 966 |
| Register 284: USB Host Configure Receive Type Endpoint 15 (USBRXTYPE15), offset 0x1FC | 966 |
| Register 285: USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D | 968 |
| Register 286: USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D | 968 |
| Register 287: USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D | 968 |
| Register 288: USB Host Receive Polling Interval Endpoint 4 (USBRXINTERVAL4), offset 0x14D | 968 |
| Register 289: USB Host Receive Polling Interval Endpoint 5 (USBRXINTERVAL5), offset 0x15D | 968 |
| Register 290: USB Host Receive Polling Interval Endpoint 6 (USBRXINTERVAL6), offset 0x16D | 968 |
| Register 291: USB Host Receive Polling Interval Endpoint 7 (USBRXINTERVAL7), offset 0x17D | 968 |
| Register 292: USB Host Receive Polling Interval Endpoint 8 (USBRXINTERVAL8), offset 0x18D | 968 |
| Register 293: USB Host Receive Polling Interval Endpoint 9 (USBRXINTERVAL9), offset 0x19D | 968 |
| Register 294: USB Host Receive Polling Interval Endpoint 10 (USBRXINTERVAL10), offset 0x1AD | 968 |
| Register 295: USB Host Receive Polling Interval Endpoint 11 (USBRXINTERVAL11), offset 0x1BD | 968 |
| Register 296: USB Host Receive Polling Interval Endpoint 12 (USBRXINTERVAL12), offset 0x1CD | 968 |
| Register 297: USB Host Receive Polling Interval Endpoint 13 (USBRXINTERVAL13), offset 0x1DD | 968 |
| Register 298: USB Host Receive Polling Interval Endpoint 14 (USBRXINTERVAL14), offset 0x1ED | 968 |
| Register 299: USB Host Receive Polling Interval Endpoint 15 (USBRXINTERVAL15), offset 0x1FD | 968 |
| Register 300: USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304 | 970 |
| Register 301: USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308 | 970 |
| Register 302: USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C | 970 |
| Register 303: USB Request Packet Count in Block Transfer Endpoint 4 (USBRQPKTCOUNT4), offset 0x310 | 970 |
| Register 304: USB Request Packet Count in Block Transfer Endpoint 5 (USBRQPKTCOUNT5), offset 0x314 | 970 |
| Register 305: USB Request Packet Count in Block Transfer Endpoint 6 (USBRQPKTCOUNT6), offset 0x318 | 970 |
| Register 306: USB Request Packet Count in Block Transfer Endpoint 7 (USBRQPKTCOUNT7), offset 0x31C | 970 |
| Register 307: USB Request Packet Count in Block Transfer Endpoint 8 (USBRQPKTCOUNT8), offset 0x320 | 970 |
| Register 308: USB Request Packet Count in Block Transfer Endpoint 9 (USBRQPKTCOUNT9), offset 0x324 | 970 |
| Register 309: USB Request Packet Count in Block Transfer Endpoint 10 (USBRQPKTCOUNT10), offset 0x328 | 970 |
| Register 310: USB Request Packet Count in Block Transfer Endpoint 11 (USBRQPKTCOUNT11), offset 0x32C | 970 |
| Register 311: USB Request Packet Count in Block Transfer Endpoint 12 (USBRQPKTCOUNT12), offset 0x330 | 970 |
| Register 312: USB Request Packet Count in Block Transfer Endpoint 13 (USBRQPKTCOUNT13), offset 0x334 | 970 |
| Register 313: USB Request Packet Count in Block Transfer Endpoint 14 (USBRQPKTCOUNT14), offset 0x338 | 970 |
| Register 314: USB Request Packet Count in Block Transfer Endpoint 15 (USBRQPKTCOUNT15), offset 0x33C | 970 |

| | |
|--|------------|
| Register 315: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340 | 972 |
| Register 316: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342 | 974 |
| Register 317: USB External Power Control (USBEPCC), offset 0x400 | 976 |
| Register 318: USB External Power Control Raw Interrupt Status (USBEPCCRIS), offset 0x404 | 979 |
| Register 319: USB External Power Control Interrupt Mask (USBEPCCIM), offset 0x408 | 980 |
| Register 320: USB External Power Control Interrupt Status and Clear (USBEPCCISC), offset 0x40C | 981 |
| Register 321: USB Device RESUME Raw Interrupt Status (USBDRRIS), offset 0x410 | 982 |
| Register 322: USB Device RESUME Interrupt Mask (USBDRIM), offset 0x414 | 983 |
| Register 323: USB Device RESUME Interrupt Status and Clear (USBDRISC), offset 0x418 | 984 |
| Register 324: USB General-Purpose Control and Status (USBGPCS), offset 0x41C | 985 |
| Register 325: USB VBUS Droop Control (USBVDC), offset 0x430 | 986 |
| Register 326: USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS), offset 0x434 | 987 |
| Register 327: USB VBUS Droop Control Interrupt Mask (USBVDCIM), offset 0x438 | 988 |
| Register 328: USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC), offset 0x43C | 989 |
| Register 329: USB ID Valid Detect Raw Interrupt Status (USBIDVRIS), offset 0x444 | 990 |
| Register 330: USB ID Valid Detect Interrupt Mask (USBIDVIM), offset 0x448 | 991 |
| Register 331: USB ID Valid Detect Interrupt Status and Clear (USBIDVISC), offset 0x44C | 992 |
| Register 332: USB DMA Select (USBDMASEL), offset 0x450 | 993 |
| Analog Comparators | 995 |
| Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000 | 1001 |
| Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004 | 1002 |
| Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008 | 1003 |
| Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010 | 1004 |
| Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020 | 1005 |
| Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040 | 1005 |
| Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x060 | 1005 |
| Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x024 | 1006 |
| Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044 | 1006 |
| Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064 | 1006 |

Revision History

The revision history table notes changes made between the indicated revisions of the LM3S9B90 data sheet.

Table 1. Revision History

| Date | Revision | Description |
|-----------|----------|---|
| May 2009 | 5285 | Started tracking revision history. |
| June 2009 | 5779 | <ul style="list-style-type: none"> ■ In System Control chapter, clarified power-on reset and external reset pin descriptions in "Reset Sources" section. ■ Added missing comparator output pin bits to DC3 register; reset value changed as well. ■ 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 registers. ■ In Request Type Support table in DMA chapter, corrected general-purpose timer row. ■ In General-Purpose Timers chapter, clarified DMA operation. ■ Added table "Preliminary Current Consumption" to Characteristics chapter. ■ Corrected Nom and Max values in "Hibernation Detailed Current Specifications" table. ■ Corrected Nom and Max values in EPI Characteristics table. ■ Added "CSn to output invalid" parameter to EPI table "EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics" and figure "Host-Bus 8/16 Mode Read Timing". ■ Corrected INL, DNL, OFF and GAIN values in ADC Characteristics table. ■ Updated ROM DriverLib appendix with RevC0 functions. ■ Updated part ordering numbers. ■ Additional minor data sheet clarifications and corrections. |

Table 1. Revision History (*continued*)

| Date | Revision | Description |
|-----------|----------|---|
| July 2009 | 5930 | <ul style="list-style-type: none"> ■ Added "Non-Blocking Read Cycle", "Normal Read Cycle", and "Write Cycle" sections to EPI chapter. ■ Corrected values for MAXADC0SPD and MAXADC1SPD bits in DC1, RCGC0, SCGC0, and DCGC0 registers. ■ Corrected figure "TI Synchronous Serial Frame Format (Single Transfer)". ■ Added description for Ethernet PHY power-saving modes. ■ Changed HIB pin from type TTL to type OD. ■ Made a number of corrections to the Electrical Characteristics chapter: <ul style="list-style-type: none"> – Deleted V_{BAT} and V_{REFA} parameters from and added footnotes to Recommended DC Operating Conditions table. – Modified Hibernation Module DC Characteristics table. – Deleted Nominal and Maximum Current Specifications section. – Modified EPI SDRAM Characteristics table: <ul style="list-style-type: none"> • Changed t_{EPIR} to t_{SDRAMR} and deleted values for 2-mA and 4-mA drive. • Changed t_{EPIF} to t_{SDRAMF} and deleted values for 2-mA and 4-mA drive. – Changed values for t_{COV}, t_{COI}, and t_{COT} parameters in EPI SDRAM Interface Characteristics table. – Deleted SDRAM Read Command Timing, SDRAM Write Command Timing, SDRAM Write Burst Timing, SDRAM Precharge Command Timing and SDRAM CAS Latency Timing figures and replaced with SDRAM Read Timing and SDRAM Write Timing figures. – Modified Host-Bus 8/16 Mode Write Timing figure. – Modified General-Purpose Mode Read and Write Timing figure. – Modified values for t_{DV} and t_{DI} parameters, and deleted t_{OD} parameter from EPI General-Purpose Interface Characteristics figure. – Major changes to ADC Characteristics tables, including adding additional tables and diagram. ■ Added missing ROM_I2SIntStatus function to ROM DriverLib Functions appendix. ■ Corrected ordering part numbers. ■ Additional minor data sheet clarifications and corrections. |

About This Document

This data sheet provides reference information for the LM3S9B90 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 documents are referenced by the data sheet, and available on the documentation CD or from the Stellaris® web site at www.luminarymicro.com:

- *ARM® Cortex™-M3 Technical Reference Manual*
- *ARM® CoreSight Technical Reference Manual*
- *ARM® v7-M Architecture Application Level Reference Manual*
- *Stellaris® Peripheral Driver Library User's Guide*
- *Stellaris® ROM User's Guide*

The following related documents are also referenced:

- *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 37.

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 0xnnn | A hexadecimal increment to a register's address, relative to that module's base address as specified in "Memory Map" on page 77. |

Table 2. Documentation Conventions (*continued*)

| Notation | Meaning |
|---------------------------------------|---|
| Register <i>N</i> | Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software. |
| reserved | Register bits marked <i>reserved</i> 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/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. |
| 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 <i>SIGNAL</i> and <i>SIGNAL</i> below). |
| deassert a signal | Change the value of the signal from the logically True state to the logically False state. |
| <i>SIGNAL</i> | Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert <i>SIGNAL</i> is to drive it Low; to deassert <i>SIGNAL</i> is to drive it High. |
| <i>SIGNAL</i> | Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert <i>SIGNAL</i> is to drive it High; to deassert <i>SIGNAL</i> 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. |

Table 2. Documentation Conventions (continued)

| Notation | Meaning |
|----------|---|
| 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

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

The LM3S9B90 microcontroller has the following features:

- ARM® Cortex™-M3 Processor Core
 - 80-MHz operation; 100 DMIPS performance
 - ARM Cortex SysTick Timer
 - Nested Vectored Interrupt Controller (NVIC)
- On-Chip Memory
 - 256 KB single-cycle Flash memory
 - 96 KB single-cycle SRAM
 - Internal ROM loaded with StellarisWare® software:
 - Stellaris® Peripheral Driver Library
 - Stellaris® Boot Loader
 - Advanced Encryption Standard (AES) cryptography tables
 - Cyclic Redundancy Check (CRC) error detection functionality
- External Peripheral Interface (EPI)
 - 8/16/32-bit dedicated parallel bus for external peripherals
 - Supports SDRAM, SRAM/Flash memory, FPGAs, CPLDs
- Advanced Serial Integration
 - 10/100 Ethernet MAC and PHY

- Two CAN 2.0 A/B controllers
- USB 2.0 OTG/Host/Device
- Three UARTs with IrDA and ISO 7816 support (one UART with full modem controls)
- Two I²C modules
- Two Synchronous Serial Interface modules (SSI)
- Integrated Interchip Sound (I²S) module
- System Integration
 - Direct Memory Access Controller (DMA)
 - System control and clocks including on-chip precision 16-MHz oscillator
 - Four 32-bit timers (up to eight 16-bit)
 - Eight Capture Compare PWM pins (CCP)
 - Lower-power battery-backed hibernation module
 - Real-Time Clock
 - Two Watchdog Timers
 - One timer runs off the main oscillator
 - One timer runs off the precision internal oscillator
 - Up to 60 GPIOs, depending on configuration
 - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
 - Independently configurable to 2, 4 or 8 mA drive capability
 - Up to 4 GPIOs can have 18 mA drive capability
- Analog
 - Two 10-bit Analog-to-Digital Converters (ADC) with sixteen analog input channels and sample rate of one million samples/second
 - Three analog comparators
 - 16 digital comparators
 - On-chip voltage regulator
- JTAG and ARM Serial Wire Debug (SWD)
- 100-pin LQFP package
- Industrial (-40°C to 85°C) Temperature Range

The LM3S9B90 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 LM3S9B90 microcontroller features a battery-backed Hibernation module to efficiently power down the LM3S9B90 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 LM3S9B90 microcontroller perfectly for battery applications.

In addition, the LM3S9B90 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 LM3S9B90 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 1132 for ordering information for Stellaris® family devices.

1.1 Functional Overview

The following sections provide an overview of the features of the LM3S9B90 microcontroller. The page number in parentheses indicates where that feature is discussed in detail. Ordering and support information can be found in "Ordering and Contact Information" on page 1132.

1.1.1 ARM Cortex™-M3

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

1.1.1.1 Processor Core (see page 64)

All members of the Stellaris® product family, including the LM3S9B90 microcontroller, are designed around an ARM Cortex™-M3 processor core. The ARM Cortex-M3 processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

- 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
- Thumb-2 mixed 16-/32-bit instruction set, delivers the high performance expected of from a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller-class applications
 - Single-cycle multiply instruction and hardware divide
 - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
 - Unaligned data access, enabling data to be efficiently packed into memory

- Harvard architecture characterized by separate buses for instruction and data
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- 80-MHz operation
- 1.25 DMIPS/MHz

“ARM Cortex-M3 Processor Core” on page 64 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.1.1.2 System Timer (SysTick) (see page 74)

ARM Cortex-M3 includes an integrated system timer, SysTick. SysTick provides a simple, 24-bit, clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways, for example:

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

1.1.1.3 Nested Vectored Interrupt Controller (NVIC) (see page 80)

The LM3S9B90 controller includes the ARM Nested Vectored Interrupt Controller (NVIC). The NVIC and Cortex-M3 prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The interrupt vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, meaning that back-to-back interrupts can be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 47 interrupts.

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

“Interrupts” on page 80 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.1.2 On-Chip Memory

The following sections describe the on-chip memory modules.

1.1.2.1 SRAM (see page 227)

The LM3S9B90 microcontroller provides 96 KB of single-cycle on-chip SRAM. The internal SRAM of the Stellaris® devices is located at offset 0x2000.0000 of the device memory map.

Because read-modify-write (RMW) operations are very time consuming, ARM has introduced *bit-banding* technology in the 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.

Data can be transferred to and from the SRAM using the Micro Direct Memory Access Controller (μDMA).

1.1.2.2 Flash Memory (see page 227)

The LM3S9B90 microcontroller provides 256 KB of single-cycle on-chip Flash memory (above 50 MHz, the Flash memory can be accessed in a single cycle as long as the code is linear; branches incur a one-cycle stall). The Flash memory is organized as a set of 2-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.1.2.3 ROM (see page 1066)

The LM3S9B90 ROM is preprogrammed with the following software and programs:

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

The Stellaris® Peripheral Driver Library is a royalty-free software library for controlling on-chip peripherals with a boot-loader capability. The library performs both peripheral initialization and control functions, with a choice of polled or interrupt-driven peripheral support. In addition, the library is designed to take full advantage of the stellar interrupt performance of the ARM® Cortex™-M3 core. No special pragmas or custom assembly code prologue/epilogue functions are required. For applications that require in-field programmability, the royalty-free Stellaris® Boot Loader can act as an application loader and support in-field firmware updates.

The Advanced Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government. AES is a strong encryption method with reasonable performance and size. In addition, it is fast in both hardware and software, is fairly easy to implement, and requires little

memory. The Texas Instruments encryption package is available with full source code, and is based on lesser general public license (LGPL) source. An LGPL means that the code can be used within an application without any copyleft implications for the application (the code does not automatically become open source). Modifications to the package source, however, must be open source.

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

1.1.3 External Peripheral Interface (see page 372)

The External Peripheral Interface (EPI) provides access to external devices using a parallel path. Unlike communications peripherals such as SSI, UART, and I²C, the EPI is designed to act like a bus to external peripherals and memory.

The EPI has the following features:

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

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

- Synchronous Dynamic Random Access Memory (SDRAM)
 - Supports x16 (single data rate) SDRAM at up to 50 MHz
 - Supports low-cost SDRAMs up to 64 MB (512 Mb)
 - Includes automatic refresh and access to all banks/rows
 - Includes a Sleep/Standby mode to keep contents active with minimal power draw
 - Multiplexed address/data interface for reduced pin count
- Host-bus

- Traditional x8 and x16 MCU bus interface capabilities
- Similar device compatibility options as PIC, ATmega, 8051, and others
- Access to SRAM, NOR Flash memory, and other devices, with up to 1 MB of addressing
- Support of both muxed and de-muxed address and data
- Access to a range of devices supporting the non-address FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) `EMPTY` and `FULL` signals
- Speed controlled, with read and write data wait-state counters
- Chip select modes include ALE, CSn, Dual CSn and ALE with dual CSn
- Manual chip-enable (or use extra address pins)
- General Purpose
 - Wide parallel interfaces for fast communications with CPLDs and FPGAs
 - Data widths up to 32-bits
 - Data rates up to 150 Mbytes/second
 - Optional “address” sizes from 4-bits to 16-bits
 - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
 - 1 to 32 bits, FIFOed with speed control
 - Useful for custom peripherals or for digital data acquisition and actuator controls

1.1.4 Serial Communications Peripherals

The LM3S9B90 controller supports both asynchronous and synchronous serial communications with:

- Ethernet MAC and PHY
- Two CAN 2.0 A/B Controllers
- USB 2.0 (full speed and low speed) OTG/Host/Device
- Three UARTs with IrDA and ISO 7816 support (one UART with full modem controls)
- Two I²C modules
- Two Synchronous Serial Interface Modules (SSI)
- Integrated Interchip Sound (I²S) Module

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

1.1.4.1 Ethernet Controller (see page 800)

Ethernet is a frame-based computer networking technology for local area networks (LANs). Ethernet has been standardized as IEEE 802.3. This specification 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 and 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
 - MDI/MDI-X cross-over support through software assist
 - Register-programmable transmit amplitude
 - Automatic polarity correction and 10BASE-T signal reception
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive channel request asserted on packet receipt
 - Transmit channel request asserted on empty transmit FIFO

1.1.4.2 Controller Area Network (see page 750)

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

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

The LM3S9B90 microcontroller includes two CAN units with the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable Loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the CANnTX and CANnRX signals

1.1.4.3 USB (see page 857)

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

The LM3S9B90 controller supports three configurations in USB 2.0 full and low speed: USB Device, USB Host, and USB On-The-Go (negotiated on-the-go as host or device when connected to other USB-enabled systems).

The USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation
- Integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 32 endpoints
 - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
 - 15 configurable IN endpoints and 15 configurable OUT endpoints

- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop and valid ID detection and interrupt
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive for up to three IN endpoints and three OUT endpoints
 - Channel requests asserted when FIFO contains required amount of data

1.1.4.4 UART (see page 573)

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 LM3S9B90 controller includes three fully programmable 16C550-type UARTs. Although the functionality is similar to a 16C550 UART, this UART design is not register compatible. The UART can generate individually masked interrupts from the Rx, Tx, modem status, and error conditions. The module generates a single combined interrupt when any of the interrupts are asserted and are unmasked.

The three UARTs have the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- False-start bit detection
- 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 μ s) bit durations

- Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Full modem handshake support (on UART1)
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
 - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

1.1.4.5 I²C (see page 676)

The Inter-Integrated Circuit (I²C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL). The I²C bus interfaces to external I²C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I²C bus may also be used for system testing and diagnostic purposes in product development and manufacture.

Each device on the I²C bus can be designated as either a master or a slave. Each I²C module supports both sending and receiving data as either a master or a slave and can operate simultaneously as both a master and a slave. Both the I²C master and slave can generate interrupts.

The LM3S9B90 controller includes two I²C modules with the following features:

- Devices on the I²C bus can be designated as either a master or a slave
 - Supports both transmitting and receiving data as either a master or a slave
 - Supports simultaneous master and slave operation
- Four I²C modes
 - Master transmit
 - Master receive
 - Slave transmit
 - Slave receive
- Two transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
- Master and slave interrupt generation
 - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)

- Slave generates interrupts when data has been transferred or requested by a master or when a START or STOP condition is detected
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

1.1.4.6 SSI (see page 634)

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

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

The LM3S9B90 controller includes two SSI modules with the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive single request asserted when data is in the FIFO; burst request asserted when FIFO contains 4 entries
 - Transmit single request asserted when there is space in the FIFO; burst request asserted when FIFO contains 4 entries

1.1.4.7 Inter-Integrated Circuit Sound (I²S) Interface (see page 713)

The I²S interface is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I²S as well as Left-Justified and Right-Justified serial audio formats. Data can be in one of four modes: Stereo, Mono, Compact 16-bit Stereo and Compact 8-Bit Stereo.

The transmit and receive modules each have an 8-entry audio-sample FIFO. An audio sample can consist of a Left and Right Stereo sample, a Mono sample, or a Left and Right Compact Stereo sample. In Compact 16-Bit Stereo, each FIFO entry contains both the 16-bit left and 16-bit right samples, allowing efficient data transfers and requiring less memory space. In Compact 8-bit Stereo,

each FIFO entry contains an 8-bit left and an 8-bit right sample, reducing memory requirements further.

Both the transmitter and receiver are capable of being a master or a slave.

The Stellaris® I²S interface has the following features:

- Configurable audio format supporting I²S, Left-justification, and Right-justification
- Configurable sample size from 8 to 32 bits
- Mono and Stereo support
- 8-, 16-, and 32-bit FIFO interface for packing memory
- Independent transmit and receive 8-entry FIFOs
- Configurable FIFO-level interrupt and μ DMA requests
- Independent transmit and receive MCLK direction control
- Transmit and receive internal MCLK sources
- Independent transmit and receive control for serial clock and word select
- MCLK and SCLK can be independently set to master or slave
- Configurable transmit zero or last sample when FIFO empty
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Burst requests
 - Channel requests asserted when FIFO contains required amount of data

1.1.5 System Integration

The LM3S9B90 controller provides a variety of standard system functions integrated into the device, including:

- Micro Direct Memory Access Controller (μ DMA)
- System control and clocks including on-chip precision 16-MHz oscillator
- ARM Cortex SysTick Timer
- Four 32-bit timers (up to eight 16-bit)
- Eight Capture Compare PWM pins (CCP)
- Lower-power battery-backed hibernation module
- Real-Time Clock
- Two Watchdog Timers

- Up to 60 GPIOs, depending on configuration
 - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
 - Independently configurable to 2, 4 or 8 mA drive capability
 - Up to 4 GPIOs can have 18 mA drive capability

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

1.1.5.1 Direct Memory Access (see page 258)

The LM3S9B90 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μ DMA). The μ DMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the available bus bandwidth. The μ DMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The μ DMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable μ DMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
 - Basic for simple transfer scenarios
 - Ping-pong for continuous data flow
 - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
 - Independently configured and operated channels
 - Dedicated channels for supported on-chip modules: GP Timer, USB, UART, Ethernet, ADC, EPI, SSI, I²S
 - Alternate channel assignments
 - One channel each for receive and transmit path for bidirectional modules
 - Dedicated channel for software-initiated transfers
 - Per-channel configurable bus arbitration scheme
 - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between μ DMA controller and the processor core
 - μ DMA controller access is subordinate to core access
 - RAM striping

- Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

1.1.5.2 System Control and Clocks (see page 95)

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

- Device identification information: version, part number, SRAM size, Flash memory size, and so on
- Power control
 - On-chip fixed Low Drop-Out (LDO) voltage regulator
 - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
 - Low-power options for microcontroller: Sleep and Deep-sleep modes with clock gating
 - Low-power options for on-chip modules: software controls shutdown of individual peripherals and memory
 - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Multiple clock sources for microcontroller system clock
 - Precision Oscillator (PIOSC): on-chip resource providing a 16 MHz $\pm 1\%$ frequency at room temperature
 - 16 MHz $\pm 3\%$ across temperature
 - Can be recalibrated with 7-bit trim resolution
 - Software power down control for low power modes
 - Main Oscillator (MOSC): a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins.
 - External oscillator used with or without on-chip PLL: select supported frequencies from 1 MHz to 16.384 MHz.
 - External crystal: from DC to maximum device speed

- Internal 30-kHz Oscillator: on chip resource providing a 30 kHz \pm 50% frequency, used during power-saving modes
- Hibernation Module clock source: eliminates need for additional crystal for main clock source
 - 32.768-kHz external oscillator
 - 4.194304-MHz external crystal
- Flexible reset sources
 - Power-on reset (POR)
 - Reset pin assertion
 - Brown-out reset (BOR) detector alerts to system power drops
 - Software reset
 - Watchdog timer reset
 - MOSC failure

1.1.5.3 Four Programmable Timers (see page 433)

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

The General-Purpose Timer Module (GPTM) contains four GPTM blocks with the following functional options:

- Count up or down
- 16- or 32-bit programmable one-shot timer
- 16- or 32-bit programmable periodic timer
- 16-bit general-purpose timer with an 8-bit prescaler
- 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
- Eight Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- ADC event trigger
- User-enabled stalling when the controller asserts CPU Halt flag during debug (excluding RTC mode)
- 16-bit input-edge count- or time-capture modes
- 16-bit PWM mode with software-programmable output inversion of the PWM signal

- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Dedicated channel for each timer
 - Burst request generated on timer interrupt

1.1.5.4 CCP Pins (see page 439)

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

The LM3S9B90 microcontroller includes eight Capture Compare PWM pins (CCP) that can be programmed to operate in the following modes:

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

1.1.5.5 Hibernation Module (see page 200)

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

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

1.1.5.6 Watchdog Timers (see page 476)

A watchdog timer is used to regain control when a system has failed due to a software error or to the failure of an external device to respond in the expected way. The Stellaris® Watchdog Timer can generate a nonmaskable interrupt (NMI) or a reset when a time-out value is reached. In addition, the Watchdog Timer is ARM FiRM-compliant and can be configured to generate an interrupt to the 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.

The LM3S9B90 microcontroller has two Watchdog Timer modules: Watchdog Timer 0 uses the system clock for its timer clock; Watchdog Timer 1 uses the PIOSC as its timer clock. The Stellaris® Watchdog Timer module has the following features:

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

1.1.5.7 Programmable GPIOs (see page 318)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections. The Stellaris® GPIO module is comprised of nine physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-60 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 1009 for the signals available to each GPIO pin).

- Up to 60 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant input/outputs
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced Host Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
 - Interrupt generation masking
 - Edge-triggered on rising, falling, or both
 - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines

- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
 - Weak pull-up or pull-down resistors
 - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can 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

1.1.6 Analog

The LM3S9B90 controller provides analog functions integrated into the device, including:

- Two 10-bit Analog-to-Digital Converters (ADC) with sixteen analog input channels and sample rate of one million samples/second
- Three analog comparators
- 16 digital comparators
- On-chip voltage regulator

The following provides more detail on these analog functions.

1.1.6.1 ADC (see page 501)

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 sixteen input channels plus an internal temperature sensor. Four buffered sample sequencers allow rapid sampling of up to eight analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. A digital comparator function is included that allows the conversion value to be diverted to a comparison unit that provides 16 digital comparators.

The LM3S9B90 microcontroller provides two ADC modules with the following features:

- Sixteen 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 sequencers 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
- Digital comparison unit providing 16 digital comparators
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Dedicated channel for each sample sequencer
 - Burst request asserted when interrupt is triggered

1.1.6.2 Analog Comparators (see page 995)

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result. The LM3S9B90 microcontroller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

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

The LM3S9B90 microcontroller provides three independent integrated analog comparators with the following functions:

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

1.1.7 JTAG and ARM Serial Wire Debug (see page 83)

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic. The TAP, Instruction Register (IR), and Data Registers (DR) can be used to test the interconnections of assembled printed circuit boards and obtain manufacturing information on the components. The JTAG Port also provides a means of accessing and controlling

design-for-test features such as I/O pin observation and control, scan testing, and debugging. Texas Instruments replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module providing all the normal JTAG debug and test functionality plus real-time access to system memory without halting the core or requiring any target resident code. See the *CoreSight™ Design Kit Technical Reference Manual* for details on SWJ-DP. The SWJ-DP interface has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
 - Serial Wire JTAG Debug Port (SWJ-DP)
 - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
 - Data Watchpoint and 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

1.1.8 Packaging and Temperature

- Industrial-range 100-pin RoHS-compliant LQFP package

1.2 Target Applications

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

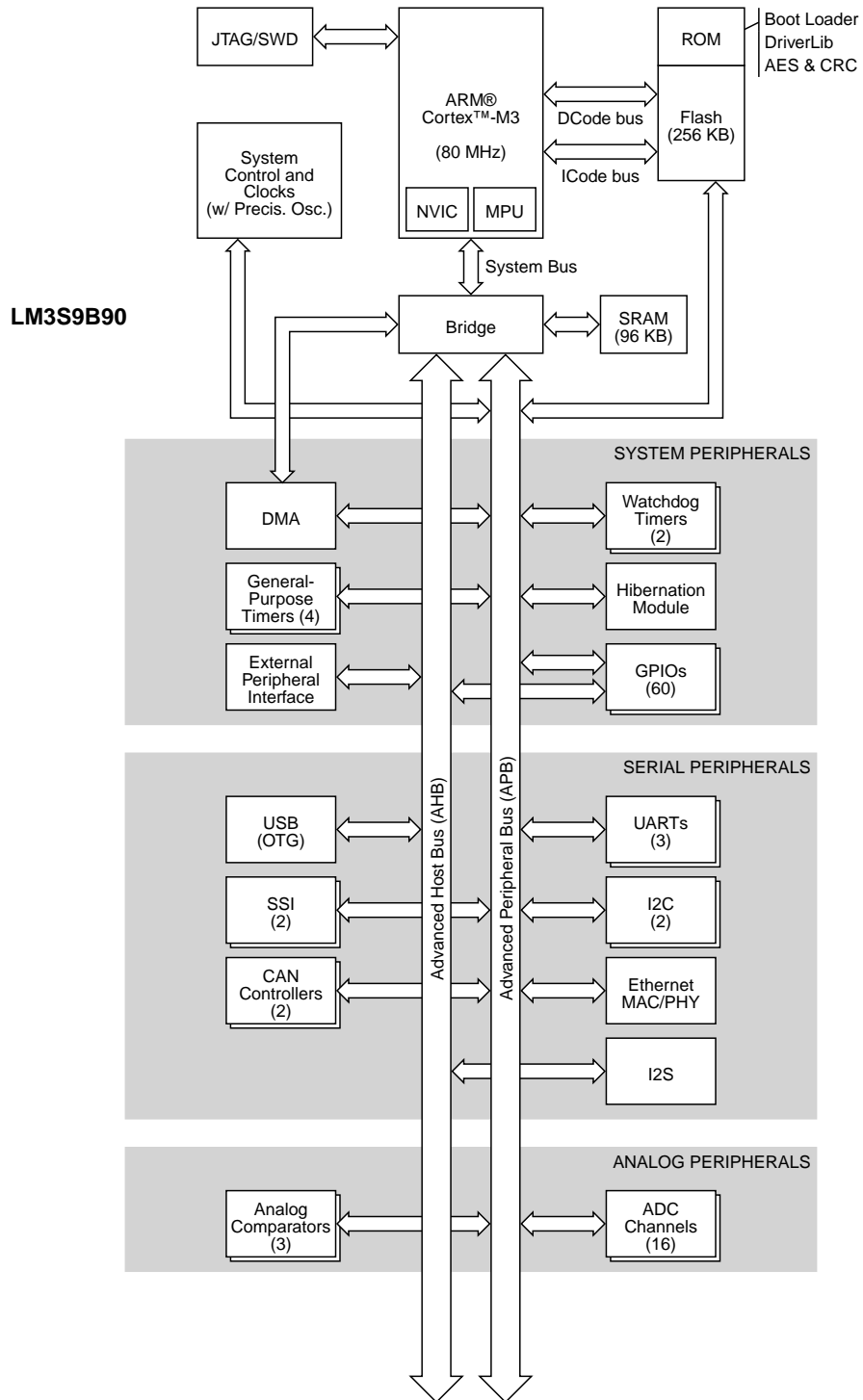
- Remote monitoring
- Electronic point-of-sale (POS) machines
- Test and measurement equipment
- Network appliances and switches
- Factory automation
- HVAC and building control
- Gaming equipment
- Motion control
- Medical instrumentation

- Fire and security
- Power and energy
- Transportation

1.3 High-Level Block Diagram

Figure 1-1 depicts the features on the Stellaris® LM3S9B90 microcontroller. Note that there are two on-chip buses that connect the core to the peripherals. The Advanced Peripheral Bus (APB) bus is the legacy bus. The Advanced Host Bus (AHB) bus provides better back-to-back access performance than the APB bus.

Figure 1-1. Stellaris[®] LM3S9B90 Microcontroller High-Level Block Diagram



1.4 Additional Features

1.4.1 Memory Map (see page 77)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S9B90 controller can be found in “Memory Map” on page 77. Register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map. The *ARM® Cortex™-M3 Technical Reference Manual* provides further information on the memory map.

1.4.2 Hardware Details

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

- “Pin Diagram” on page 1008
- “Signal Tables” on page 1009
- “Operating Characteristics” on page 1038
- “Electrical Characteristics” on page 1039
- “Package Information” on page 1134

2 ARM Cortex-M3 Processor Core

The ARM Cortex-M3 processor provides a high-performance, low-cost platform that meets the system requirements of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts. Features include:

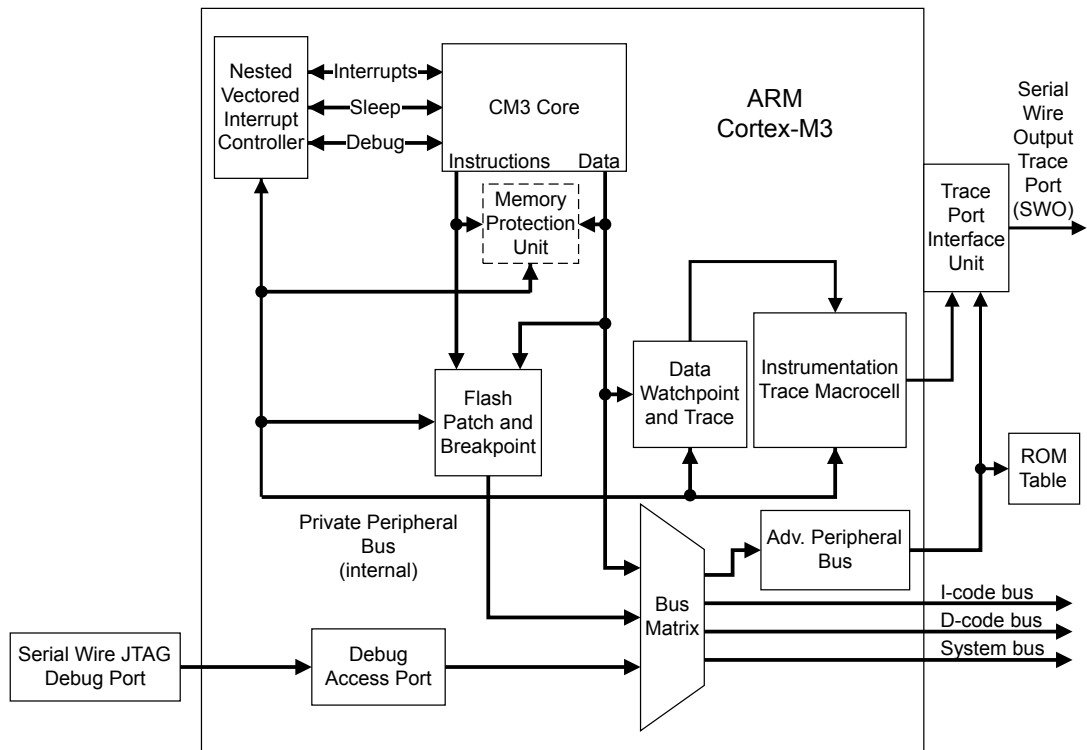
- 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
- Thumb-2 mixed 16-/32-bit instruction set, delivers the high performance expected of from a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller-class applications
 - Single-cycle multiply instruction and hardware divide
 - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
 - Unaligned data access, enabling data to be efficiently packed into memory
- Harvard architecture characterized by separate buses for instruction and data
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- 80-MHz operation
- 1.25 DMIPS/MHz

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

For more information on the ARM Cortex-M3 processor core, see the *ARM® Cortex™-M3 Technical Reference Manual*. For information on SWJ-DP, see the *ARM® CoreSight Technical Reference Manual*.

2.1 Block Diagram

Figure 2-1. CPU Block Diagram



2.2 Functional Description

Important: The *ARM® Cortex™-M3 Technical Reference Manual* describes all the features of an ARM Cortex-M3 in detail. However, these features differ based on the implementation. This section describes the Stellaris® implementation.

Texas Instruments implements the ARM Cortex-M3 core as shown in Figure 2-1 on page 65. The Cortex-M3 uses the entire 16-bit Thumb instruction set and the base Thumb-2 32-bit instruction set. In addition, as noted in the *ARM® Cortex™-M3 Technical Reference Manual*, several Cortex-M3 components are flexible in their implementation: SW/JTAG-DP, ETM, TPIU, the ROM table, the MPU, and the Nested Vectored Interrupt Controller (NVIC). Each of these is addressed in the sections that follow.

2.2.1 Programming Model

This section provides a brief overview of the programming model for the Cortex-M3 core. More detailed information can be found in the *ARM® Cortex™-M3 Technical Reference Manual*.

- Privileged access and user access - Code can execute as privileged or unprivileged. Unprivileged execution limits or excludes access to some resources. Privileged execution has access to all resources. Handler mode is always privileged. Thread mode can be privileged or unprivileged. Thread mode is privileged out of reset, but you can change it to user or unprivileged by setting the CONTROL[0] bit using the MSR instruction. User access prevents:
 - Use of some instructions such as CPS to set FAULTMASK and PRIMASK

- Access to most registers in System Control Space (SCS)

When Thread mode has been changed from privileged to user, it cannot change itself back to privileged. Only a Handler can change the privilege of Thread mode. Handler mode is always privileged.

- Register set - The processor has the following 32-bit registers:
 - 13 general-purpose registers, r0-r12
 - Stack point alias of banked registers, SP_process and SP_main
 - Link register, r14
 - Program counter, r15
 - One program status register, xPSR.
- Data types - The processor supports the following data types:
 - 32-bit words
 - 16-bit halfwords
 - 8-bit bytes
- Memory formats - The processor views memory as a linear collection of bytes numbered in ascending order from 0. For example, bytes 0-3 hold the first stored word and bytes 4-7 hold the second stored word. The processor accesses code and data in little-endian format, which means that the byte with the lowest address in a word is the least-significant byte of the word. The byte with the highest address in a word is the most significant. The byte at address 0 of the memory system connects to data lines 7-0.
- Instruction set - The Cortex-M3 instruction set contains both 16 and 32-bit instructions. These instructions are summarized in Table 2-1 on page 66 and Table 2-2 on page 68, respectively.

Table 2-1. 16-Bit Cortex-M3 Instruction Set Summary

| Operation | Assembler |
|---|------------------------------|
| Add register value and C flag to register value | ADC <Rd>, <Rm> |
| Add immediate 3-bit value to register | ADD <Rd>, <Rn>, #<immed_3> |
| Add immediate 8-bit value to register | ADD <Rd>, #<immed_8> |
| Add low register value to low register value | ADD <Rd>, <Rn>, <Rm> |
| Add high register value to low or high register value | ADD <Rd>, <Rm> |
| Add 4* (immediate 8-bit value) with PC to register | ADD <Rd>, PC, #<immed_8> * 4 |
| Add 4* (immediate 8-bit value) with SP to register | ADD <Rd>, SP, #<immed_8> * 4 |
| Add 4* (immediate 7-bit value) to SP | ADD SP, #<immed_7> * 4 |
| Bitwise AND register values | AND <Rd>, <Rm> |
| Arithmetic shift right by immediate number | ASR <Rd>, <Rm>, #<immed_5> |
| Arithmetic shift right by number in register | ASR <Rd>, <Rs> |
| Branch conditional | B<cond> <target address> |
| Branch unconditional | B <target_address> |
| Bit clear | BIC <Rd>, <Rm> |

Table 2-1. 16-Bit Cortex-M3 Instruction Set Summary (continued)

| Operation | Assembler |
|--|-----------------------------------|
| Software breakpoint | BKPT <immed_8> |
| Branch with link | BL <Rm> |
| Branch with link and exchange | BLX <Rm> |
| Branch and exchange | BX <Rm> |
| Compare not zero and branch | CBNZ <Rn>, <label> |
| Compare zero and branch | CBZ <Rn>, <label> |
| Compare negation of register value with another register value | CMN <Rn>, <Rm> |
| Compare immediate 8-bit value | CMP <Rn>, #<immed_8> |
| Compare registers | CMP <Rn>, <Rm> |
| Compare high register to low or high register | CMP <Rn>, <Rm> |
| Change processor state | CPS <effect>, <iflags> |
| Copy high or low register value to another high or low register | CPY <Rd> <Rm> |
| Bitwise exclusive OR register values | EOR <Rd>, <Rm> |
| Condition the following instruction | IT <cond> |
| Condition the following two instructions | IT<x> <cond> |
| Condition the following three instructions | IT<x><y> <cond> |
| Condition the following four instructions | IT<x><y><z> <cond> |
| Multiple sequential memory word loads | LDMIA <Rn>!, <registers> |
| Load memory word from base register address + 5-bit immediate offset | LDR <Rd>, [<Rn>, #<immed_5> * 4] |
| Load memory word from base register address + register offset | LDR <Rd>, [<Rn>, <Rm>] |
| Load memory word from PC address + 8-bit immediate offset | LDR <Rd>, [PC, #<immed_8> * 4] |
| Load memory word from SP address + 8-bit immediate offset | LDR, <Rd>, [SP, #<immed_8> * 4] |
| Load memory byte [7:0] from register address + 5-bit immediate offset | LDRB <Rd>, [<Rn>, #<immed_5>] |
| Load memory byte [7:0] from register address + register offset | LDRB <Rd>, [<Rn>, <Rm>] |
| Load memory halfword [15:0] from register address + 5-bit immediate offset | LDRH <Rd>, [<Rn>, #<immed_5> * 2] |
| Load halfword [15:0] from register address + register offset | LDRH <Rd>, [<Rn>, <Rm>] |
| Load signed byte [7:0] from register address + register offset | LDRSB <Rd>, [<Rn>, <Rm>] |
| Load signed halfword [15:0] from register address + register offset | LDRSH <Rd>, [<Rn>, <Rm>] |
| Logical shift left by immediate number | LSL <Rd>, <Rm>, #<immed_5> |
| Logical shift left by number in register | LSL <Rd>, <Rs> |
| Logical shift right by immediate number | LSR <Rd>, <Rm>, #<immed_5> |
| Logical shift right by number in register | LSR <Rd>, <Rs> |
| Move immediate 8-bit value to register | MOV <Rd>, #<immed_8> |
| Move low register value to low register | MOV <Rd>, <Rn> |
| Move high or low register value to high or low register | MOV <Rd>, <Rm> |
| Multiply register values | MUL <Rd>, <Rm> |
| Move complement of register value to register | MVN <Rd>, <Rm> |
| Negate register value and store in register | NEG <Rd>, <Rm> |
| No operation | NOP <c> |
| Bitwise logical OR register values | ORR <Rd>, <Rm> |
| Pop registers from stack | POP <registers> |
| Pop registers and PC from stack | POP <registers, PC> |

Table 2-1. 16-Bit Cortex-M3 Instruction Set Summary (continued)

| Operation | Assembler |
|---|-----------------------------------|
| Push registers onto stack | PUSH <registers> |
| Push LR and registers onto stack | PUSH <registers, LR> |
| Reverse bytes in word and copy to register | REV <Rd>, <Rn> |
| Reverse bytes in two halfwords and copy to register | REV16 <Rd>, <Rn> |
| Reverse bytes in low halfword [15:0], sign-extend, and copy to register | REVSH <Rd>, <Rn> |
| Rotate right by amount in register | ROR <Rd>, <Rs> |
| Subtract register value and C flag from register value | SBC <Rd>, <Rm> |
| Send event | SEV <c> |
| Store multiple register words to sequential memory locations | STMIA <Rn>!, <registers> |
| Store register word to register address + 5-bit immediate offset | STR <Rd>, [<Rn>, #<immed_5> * 4] |
| Store register word to register address | STR <Rd>, [<Rn>, <Rm>] |
| Store register word to SP address + 8-bit immediate offset | STR <Rd>, [SP, #<immed_8> * 4] |
| Store register byte [7:0] to register address + 5-bit immediate offset | STRB <Rd>, [<Rn>, #<immed_5>] |
| Store register byte [7:0] to register address | STRB <Rd>, [<Rn>, <Rm>] |
| Store register halfword [15:0] to register address + 5-bit immediate offset | STRH <Rd>, [<Rn>, #<immed_5> * 2] |
| Store register halfword [15:0] to register address + register offset | STRH <Rd>, [<Rn>, <Rm>] |
| Subtract immediate 3-bit value from register | SUB <Rd>, <Rn>, #<immed_3> |
| Subtract immediate 8-bit value from register value | SUB <Rd>, #<immed_8> |
| Subtract register values | SUB <Rd>, <Rn>, <Rm> |
| Subtract 4 (immediate 7-bit value) from SP | SUB SP, #<immed_7> * 4 |
| Operating system service call with 8-bit immediate call code | SVC <immed_8> |
| Extract byte [7:0] from register, move to register, and sign-extend to 32 bits | SXTB <Rd>, <Rm> |
| Extract halfword [15:0] from register, move to register, and sign-extend to 32 bits | SXTH <Rd>, <Rm> |
| Test register value for set bits by ANDing it with another register value | TST <Rn>, <Rm> |
| Extract byte [7:0] from register, move to register, and zero-extend to 32 bits | UXTB <Rd>, <Rm>#10 |
| Extract halfword [15:0] from register, move to register, and zero-extend to 32 bits | UXTH <Rd>, <Rm> |
| Wait for event | WFE <c> |
| Wait for interrupt | WFI <c> |

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary

| Operation | Assembler |
|--|---|
| Add register value, immediate 12-bit value, and C bit | ADC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12> |
| Add register value, shifted register value, and C bit | ADC{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Add register value and immediate 12-bit value | ADD{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |
| Add register value and shifted register value | ADD{S}.W <Rd>, <Rm>{, <shift>} |
| Add register value and immediate 12-bit value | ADDW.W <Rd>, <Rn>, #<immed_12> |
| Bitwise AND register value with immediate 12-bit value | AND{S}.W <Rd>, <Rn>, #<modify_constant(immed_12> |
| Bitwise AND register value with shifted register value | AND{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Arithmetic shift right by number in register | ASR{S}.W <Rd>, <Rn>, <Rm> |
| Conditional branch | B{cond}.W <label> |
| Clear bit field | BFC.W <Rd>, #<lsb>, #<width> |

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary (continued)

| Operation | Assembler |
|---|--|
| Insert bit field from one register value into another | BFI.W <Rd>, <Rn>, #<lsb>, #<width> |
| Bitwise AND register value with complement of immediate 12-bit value | BIC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |
| Bitwise AND register value with complement of shifted register value | BIC{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Branch with link | BL <label> |
| Branch with link (immediate) | BL<c> <label> |
| Unconditional branch | B.W <label> |
| Clear exclusive clears the local record of the executing processor that an address has had a request for an exclusive access. | CLREX <c> |
| Return number of leading zeros in register value | CLZ.W <Rd>, <Rn> |
| Compare register value with two's complement of immediate 12-bit value | CMN.W <Rn>, #<modify_constant(immed_12)> |
| Compare register value with two's complement of shifted register value | CMN.W <Rn>, <Rm>{, <shift>} |
| Compare register value with immediate 12-bit value | CMP.W <Rn>, #<modify_constant(immed_12)> |
| Compare register value with shifted register value | CMP.W <Rn>, <Rm>{, <shift>} |
| Data memory barrier | DMB <c> |
| Data synchronization barrier | DSB <c> |
| Exclusive OR register value with immediate 12-bit value | EOR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |
| Exclusive OR register value with shifted register value | EOR{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Instruction synchronization barrier | ISB <c> |
| Load multiple memory registers, increment after or decrement before | LDM{IA DB}.W <Rn>{!}, <registers> |
| Memory word from base register address + immediate 12-bit offset | LDR.W <Rxf>, [<Rn>, #<offset_12>] |
| Memory word to PC from register address + immediate 12-bit offset | LDR.W PC, [<Rn>, #<offset_12>] |
| Memory word to PC from base register address immediate 8-bit offset, postindexed | LDR.W PC, [Rn], #<+/-<offset_8> |
| Memory word from base register address immediate 8-bit offset, postindexed | LDR.W <Rxf>, [<Rn>], #+/-<offset_8> |
| Memory word from base register address immediate 8-bit offset, preindexed | LDR.W <Rxf>, [<Rn>, #<+/-<offset_8>!] LDRT.W <Rxf>, [<Rn>, #<offset_8>] |
| Memory word to PC from base register address immediate 8-bit offset, preindexed | LDR.W PC, [<Rn>, #+/-<offset_8>!] |
| Memory word from register address shifted left by 0, 1, 2, or 3 places | LDR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Memory word to PC from register address shifted left by 0, 1, 2, or 3 places | LDR.W PC, [<Rn>, <Rm>{, LSL #<shift>}] |
| Memory word from PC address immediate 12-bit offset | LDR.W <Rxf>, [PC, #+/-<offset_12>] |
| Memory word to PC from PC address immediate 12-bit offset | LDR.W PC, [PC, #+/-<offset_12>] |
| Memory byte [7:0] from base register address + immediate 12-bit offset | LDRB.W <Rxf>, [<Rn>, #<offset_12>] |
| Memory byte [7:0] from base register address immediate 8-bit offset, postindexed | LDRB.W <Rxf>. [<Rn>], #+/-<offset_8> |
| Memory byte [7:0] from register address shifted left by 0, 1, 2, or 3 places | LDRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Memory byte [7:0] from base register address immediate 8-bit offset, preindexed | LDRB.W <Rxf>, [<Rn>, #<+/-<offset_8>!] |
| Memory byte from PC address immediate 12-bit offset | LDRB.W <Rxf>, [PC, #+/-<offset_12>] |
| Memory doubleword from register address 8-bit offset 4, preindexed | LDRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]! |
| Memory doubleword from register address 8-bit offset 4, postindexed | LDRD.W <Rxf>, <Rxf2>, [<Rn>], #+/-<offset_8> * 4 |
| Load register exclusive calculates an address from a base register value and an immediate offset, loads a word from memory, writes it to a register | LDREX<c> <Rt>,[<Rn>{, #<imm>}] |

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary (continued)

| Operation | Assembler |
|--|---|
| Load register exclusive halfword calculates an address from a base register value and an immediate offset, loads a halfword from memory, writes it to a register | LDREXH<c> <Rt>,<Rn>{,<#<imm>}] |
| Load register exclusive byte calculates an address from a base register value and an immediate offset, loads a byte from memory, writes it to a register | LDREXB<c> <Rt>,<Rn>{,<#<imm>}] |
| Memory halfword [15:0] from base register address + immediate 12-bit offset | LDRH.W <Rxf>, [<Rn>, #<offset_12>] |
| Memory halfword [15:0] from base register address immediate 8-bit offset, preindexed | LDRH.W <Rxf>, [<Rn>, #<+/-<offset_8>]! |
| Memory halfword [15:0] from base register address immediate 8-bit offset, postindexed | LDRH.W <Rxf>. [<Rn>], #<+/-<offset_8> |
| Memory halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places | LDRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Memory halfword from PC address immediate 12-bit offset | LDRH.W <Rxf>, [PC, #<+/-<offset_12>] |
| Memory signed byte [7:0] from base register address + immediate 12-bit offset | LDRSB.W <Rxf>, [<Rn>, #<offset_12>] |
| Memory signed byte [7:0] from base register address immediate 8-bit offset, postindexed | LDRSB.W <Rxf>. [<Rn>], #<+/-<offset_8> |
| Memory signed byte [7:0] from base register address immediate 8-bit offset, preindexed | LDRSB.W <Rxf>, [<Rn>, #<+/-<offset_8>]! |
| Memory signed byte [7:0] from register address shifted left by 0, 1, 2, or 3 places | LDRSB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Memory signed byte from PC address immediate 12-bit offset | LDRSB.W <Rxf>, [PC, #<+/-<offset_12>] |
| Memory signed halfword [15:0] from base register address + immediate 12-bit offset | LDRSH.W <Rxf>, [<Rn>, #<offset_12>] |
| Memory signed halfword [15:0] from base register address immediate 8-bit offset, postindexed | LDRSH.W <Rxf>. [<Rn>], #<+/-<offset_8> |
| Memory signed halfword [15:0] from base register address immediate 8-bit offset, preindexed | LDRSH.W <Rxf>, [<Rn>, #<+/-<offset_8>]! |
| Memory signed halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places | LDRSH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Memory signed halfword from PC address immediate 12-bit offset | LDRSH.W <Rxf>, [PC, #<+/-<offset_12>] |
| Logical shift left register value by number in register | LSL{S}.W <Rd>, <Rn>, <Rm> |
| Logical shift right register value by number in register | LSR{S}.W <Rd>, <Rn>, <Rm> |
| Multiply two signed or unsigned register values and add the low 32 bits to a register value | MLA.W <Rd>, <Rn>, <Rm>, <Racc> |
| Multiply two signed or unsigned register values and subtract the low 32 bits from a register value | MLS.W <Rd>, <Rn>, <Rm>, <Racc> |
| Move immediate 12-bit value to register | MOV{S}.W <Rd>, #<modify_constant(immed_12)> |
| Move shifted register value to register | MOV{S}.W <Rd>, <Rm>{, <shift>} |
| Move immediate 16-bit value to top halfword [31:16] of register | MOVT.W <Rd>, #<immed_16> |
| Move immediate 16-bit value to bottom halfword [15:0] of register and clear top halfword [31:16] | MOVW.W <Rd>, #<immed_16> |
| Move to register from status | MRS<c> <Rd>, <psr> |
| Move to status register | MSR<c> <psr>_<fields>,<Rn> |
| Multiply two signed or unsigned register values | MUL.W <Rd>, <Rn>, <Rm> |
| No operation | NOP.W |
| Logical OR NOT register value with immediate 12-bit value | ORN{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |
| Logical OR NOT register value with shifted register value | ORN{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary (continued)

| Operation | Assembler |
|---|---|
| Logical OR register value with immediate 12-bit value | ORR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |
| Logical OR register value with shifted register value | ORR{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Reverse bit order | RBIT.W <Rd>, <Rm> |
| Reverse bytes in word | REV.W <Rd>, <Rm> |
| Reverse bytes in each halfword | REV16.W <Rd>, <Rn> |
| Reverse bytes in bottom halfword and sign-extend | REVSH.W <Rd>, <Rn> |
| Rotate right by number in register | ROR{S}.W <Rd>, <Rn>, <Rm> |
| Rotate right with extend | RRX{S}.W <Rd>, <Rm> |
| Subtract a register value from an immediate 12-bit value | RSB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |
| Subtract a register value from a shifted register value | RSB{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Subtract immediate 12-bit value and C bit from register value | SBC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |
| Subtract shifted register value and C bit from register value | SBC{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Copy selected bits to register and sign-extend | SBFX.W <Rd>, <Rn>, #<lsb>, #<width> |
| Signed divide | SDIV<c> <Rd>, <Rn>, <Rm> |
| Send event | SEV<c> |
| Multiply signed words and add signed-extended value to 2-register value | SMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm> |
| Multiply two signed register values | SMULL.W <RdLo>, <RdHi>, <Rn>, <Rm> |
| Signed saturate | SSAT.W <c> <Rd>, #<imm>, <Rn>{, <shift>} |
| Multiple register words to consecutive memory locations | STM{IA DB}.W <Rn>{!}, <registers> |
| Register word to register address + immediate 12-bit offset | STR.W <Rxf>, [<Rn>, #<offset_12>] |
| Register word to register address immediate 8-bit offset, postindexed | STR.W <Rxf>, [<Rn>], #+/-<offset_8> |
| Register word to register address shifted by 0, 1, 2, or 3 places | STR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Register word to register address immediate 8-bit offset, preindexed Store, preindexed | STR.W <Rxf>, [<Rn>, #+/-<offset_8>]{!} STRT.W <Rxf>, [<Rn>, #<offset_8>] |
| Register byte [7:0] to register address immediate 8-bit offset, preindexed | STRB{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!} |
| Register byte [7:0] to register address + immediate 12-bit offset | STRB.W <Rxf>, [<Rn>, #<offset_12>] |
| Register byte [7:0] to register address immediate 8-bit offset, postindexed | STRB.W <Rxf>, [<Rn>], #+/-<offset_8> |
| Register byte [7:0] to register address shifted by 0, 1, 2, or 3 places | STRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Store doubleword, preindexed | STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]{!} |
| Store doubleword, postindexed | STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4] |
| Store register exclusive calculates an address from a base register value and an immediate offset, and stores a word from a register to memory if the executing processor has exclusive access to the memory addressed. | STREX <c> <Rd>, <Rt>, [<Rn>{, #<imm>}] |
| Store register exclusive byte derives an address from a base register value, and stores a byte from a register to memory if the executing processor has exclusive access to the memory addressed | STREXB <c> <Rd>, <Rt>, [<Rn>] |
| Store register exclusive halfword derives an address from a base register value, and stores a halfword from a register to memory if the executing processor has exclusive access to the memory addressed. | STREXH <c> <Rd>, <Rt>, [<Rn>] |
| Register halfword [15:0] to register address + immediate 12-bit offset | STRH.W <Rxf>, [<Rn>, #<offset_12>] |
| Register halfword [15:0] to register address shifted by 0, 1, 2, or 3 places | STRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}] |
| Register halfword [15:0] to register address immediate 8-bit offset, preindexed | STRH{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!} |
| Register halfword [15:0] to register address immediate 8-bit offset, postindexed | STRH.W <Rxf>, [<Rn>], #+/-<offset_8> |
| Subtract immediate 12-bit value from register value | SUB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)> |

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary (continued)

| Operation | Assembler |
|---|--|
| Subtract shifted register value from register value | SUB{S}.W <Rd>, <Rn>, <Rm>{, <shift>} |
| Subtract immediate 12-bit value from register value | SUBW.W <Rd>, <Rn>, #<immed_12> |
| Sign extend byte to 32 bits | SXTB.W <Rd>, <Rm>{, <rotation>} |
| Sign extend halfword to 32 bits | SXTH.W <Rd>, <Rm>{, <rotation>} |
| Table branch byte | TBB [<Rn>, <Rm>] |
| Table branch halfword | TBH [<Rn>, <Rm>, LSL #1] |
| Exclusive OR register value with immediate 12-bit value | TEQ.W <Rn>, #<modify_constant(immed_12)> |
| Exclusive OR register value with shifted register value | TEQ.W <Rn>, <Rm>{, <shift>} |
| Logical AND register value with 12-bit immediate value | TST.W <Rn>, #<modify_constant(immed_12)> |
| Logical AND register value with shifted register value | TST.W <Rn>, <Rm>{, <shift>} |
| Copy bit field from register value to register and zero-extend to 32 bits | UBFX.W <Rd>, <Rn>, #<lsb>, #<width> |
| Unsigned divide | UDIV<c> <Rd>, <Rn>, <Rm> |
| Multiply two unsigned register values and add to a 2-register value | UMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm> |
| Multiply two unsigned register values | UMULL.W <RdLo>, <RdHi>, <Rn>, <Rm> |
| Unsigned saturate | USAT <c> <Rd>, #<imm>, <Rn>{, <shift>} |
| Copy unsigned byte to register and zero-extend to 32 bits | UXTB.W <Rd>, <Rm>{, <rotation>} |
| Copy unsigned halfword to register and zero-extend to 32 bits | UXTH.W <Rd>, <Rm>{, <rotation>} |
| Wait for event | WFE.W |
| Wait for interrupt | WFI.W |

2.2.2 Serial Wire and JTAG Debug

Texas Instruments replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. As a result, Chapter 12, “Debug Port,” of the *ARM® Cortex™-M3 Technical Reference Manual* does not apply to Stellaris® devices.

The SWJ-DP interface combines the SWD and JTAG debug ports into one module. See the *CoreSight™ Design Kit Technical Reference Manual* for details on SWJ-DP.

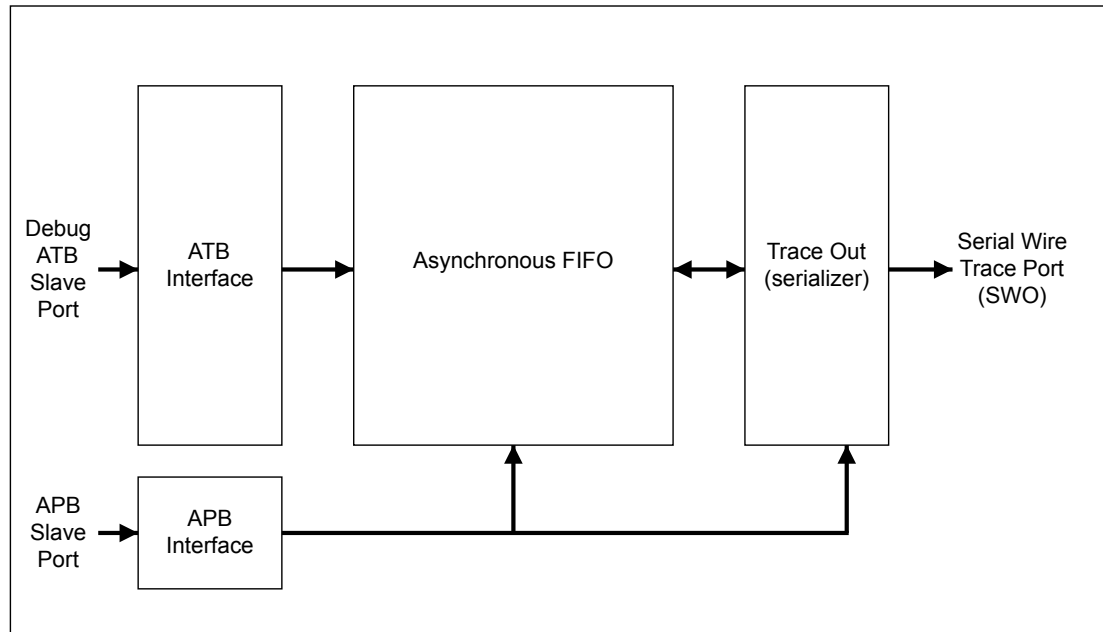
2.2.3 Embedded Trace Macrocell (ETM)

ETM is not implemented in the Stellaris® devices. As a result, Chapters 15 and 16 of the *ARM® Cortex™-M3 Technical Reference Manual* can be ignored.

2.2.4 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. Stellaris® devices implement the TPIU as shown in Figure 2-2. This implementation is similar to the non-ETM version described in the *ARM® Cortex™-M3 Technical Reference Manual*, however, SWJ-DP only provides the Serial Wire Viewer (SWV) output format for the TPIU.

Figure 2-2. TPIU Block Diagram



2.2.5 ROM Table

The default ROM table is implemented as described in the *ARM® Cortex™-M3 Technical Reference Manual*.

2.2.6 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) is included on the LM3S9B90 controller and 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.

2.2.7 Nested Vectored Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC):

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

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts.

You can only fully access the NVIC from privileged mode, but you can pend interrupts in user-mode by enabling the Configuration Control Register (see the *ARM® Cortex™-M3 Technical Reference Manual*). Any other user-mode access causes a bus fault.

All NVIC registers are accessible using byte, halfword, and word unless otherwise stated.

2.2.7.1 Interrupts

The ARM® Cortex™-M3 *Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S9B90 microcontroller supports 47 interrupts with eight priority levels.

In addition to the peripheral interrupts, the system also provides for a non-maskable interrupt (NMI). The NMI is generally used in safety critical applications where the immediate execution of an interrupt handler is required. The NMI signal is available as an external signal so that it may be generated by external circuitry. The NMI is also used internally as part of the main oscillator verification circuitry. More information on the non-maskable interrupt is located in “Non-Maskable Interrupt” on page 99.

2.2.8 System Timer (SysTick)

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 used 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.

2.2.8.1 Functional Description

The timer consists of three registers:

- SysTick Control and Status Register - a control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status
- SysTick Reload Value Register - the reload value for the counter, used to provide the counter's wrap value
- SysTick Current Value Register - the current value of the counter

A fourth register, the SysTick Calibration Value Register, is not implemented in the Stellaris® devices.

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

Writing to the SysTick Current Value register clears the register and the COUNTFLAG 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.

If the core is in debug state (halted), the counter does not decrement. The timer is clocked with respect to a reference clock, which can be either the core clock or an external clock source.

2.2.8.2 SysTick Control and Status Register

Use the SysTick Control and Status Register to enable the SysTick features. The reset is 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 | COUNTFLAG | R/W | 0 | Count Flag When set, this bit indicates that the timer has counted to 0 since the last time this register was read. This bit is cleared by a read of the register. 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 COUNTFLAG bit is not changed by the debugger read. |
| 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 | CLKSOURCE | R/W | 0 | Clock Source Value Description 0 External reference clock. (Not implemented for Stellaris® microcontrollers.) 1 Core clock Because an external reference clock is not supported, this bit must be set in order for SysTick to operate. |
| 1 | TICKINT | R/W | 0 | Tick Interrupt When set, this bit causes an interrupt to be generated to the NVIC when SysTick counts to 0. When clear, interrupt generation is disabled. Software can use the COUNTFLAG to determine if the counter has ever reached 0. |
| 0 | ENABLE | R/W | 0 | Enable When set, this bit 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 COUNTFLAG bit is set and an interrupt is generated if enabled by TICKINT. The counter then loads the Reload value again and begins counting. When this bit is clear, the counter is disabled. |

2.2.8.3 SysTick Reload Value Register

The SysTick Reload Value Register specifies the start value to load into the SysTick Current Value Register when the counter reaches 0. The start value can be between 1 and 0x00FF.FFFF. A start value of 0 is possible but has no effect because the SysTick interrupt and COUNTFLAG 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.

When configuring SysTick as a single-shot timer, a new value is written on each tick interrupt, and the actual count down value must be written. For example, if a tick is next required after 400 clock pulses, 400 must be written into the RELOAD field.

| 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 | - | Reload Value Value to load into the SysTick Current Value Register when the counter reaches 0. |

2.2.8.4 SysTick Current Value Register

The SysTick Current Value Register contains the current value of the counter.

| 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 | W1C | - | 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 to 0. Clearing this register also clears the COUNTFLAG bit of the SysTick Control and Status Register. |

2.2.8.5 SysTick Calibration Value Register

The SysTick Calibration Value register is not implemented.

3 Memory Map

The memory map for the LM3S9B90 controller is provided in Table 3-1.

In this manual, register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map. See also Chapter 4, "Memory Map" in the *ARM® Cortex™-M3 Technical Reference Manual*.

Note that within the memory map, all reserved space returns a bus fault when read or written.

Table 3-1. Memory Map

| Start | End | Description | For details, see page ... |
|-------------------------|-------------|---|---------------------------|
| Memory | | | |
| 0x0000.0000 | 0x0003.FFFF | On-chip Flash | 227 |
| 0x0004.0000 | 0x00FF.FFFF | Reserved | - |
| 0x0100.0000 | 0x0100.5EFF | On-chip ROM with Boot Loader, DriverLib, AES+CRC software | 227 |
| 0x0100.5F00 | 0x1FFF.FFFF | Reserved | - |
| 0x2000.0000 | 0x2001.7FFF | Bit-banded on-chip SRAM | 227 |
| 0x2001.8000 | 0x21FF.FFFF | Reserved | - |
| 0x2200.0000 | 0x222F.FFFF | Bit-band alias of 0x2000.0000 through 0x200F.FFFF | 227 |
| 0x2230.0000 | 0x3FFF.FFFF | Reserved | - |
| FIRM Peripherals | | | |
| 0x4000.0000 | 0x4000.0FFF | Watchdog timer 0 | 479 |
| 0x4000.1000 | 0x4000.1FFF | Watchdog timer 1 | 479 |
| 0x4000.2000 | 0x4000.3FFF | Reserved | - |
| 0x4000.4000 | 0x4000.4FFF | GPIO Port A | 329 |
| 0x4000.5000 | 0x4000.5FFF | GPIO Port B | 329 |
| 0x4000.6000 | 0x4000.6FFF | GPIO Port C | 329 |
| 0x4000.7000 | 0x4000.7FFF | GPIO Port D | 329 |
| 0x4000.8000 | 0x4000.8FFF | SSI0 | 648 |
| 0x4000.9000 | 0x4000.9FFF | SSI1 | 648 |
| 0x4000.A000 | 0x4000.BFFF | Reserved | - |
| 0x4000.C000 | 0x4000.CFFF | UART0 | 585 |
| 0x4000.D000 | 0x4000.DFFF | UART1 | 585 |
| 0x4000.E000 | 0x4000.EFFF | UART2 | 585 |
| 0x4000.F000 | 0x4001.FFFF | Reserved | - |
| Peripherals | | | |
| 0x4002.0000 | 0x4002.07FF | I ² C Master 0 | 691 |
| 0x4002.0800 | 0x4002.0FFF | I ² C Slave 0 | 704 |
| 0x4002.1000 | 0x4002.17FF | I ² C Master 1 | 691 |
| 0x4002.1800 | 0x4002.1FFF | I ² C Slave 1 | 704 |
| 0x4002.2000 | 0x4002.3FFF | Reserved | - |
| 0x4002.4000 | 0x4002.4FFF | GPIO Port E | 329 |
| 0x4002.5000 | 0x4002.5FFF | GPIO Port F | 329 |

Table 3-1. Memory Map (continued)

| Start | End | Description | For details, see page ... |
|-------------|-------------|---|---------------------------|
| 0x4002.6000 | 0x4002.6FFF | GPIO Port G | 329 |
| 0x4002.7000 | 0x4002.7FFF | GPIO Port H | 329 |
| 0x4002.8000 | 0x4002.FFFF | Reserved | - |
| 0x4003.0000 | 0x4003.0FFF | Timer 0 | 447 |
| 0x4003.1000 | 0x4003.1FFF | Timer 1 | 447 |
| 0x4003.2000 | 0x4003.2FFF | Timer 2 | 447 |
| 0x4003.3000 | 0x4003.3FFF | Timer 3 | 447 |
| 0x4003.4000 | 0x4003.7FFF | Reserved | - |
| 0x4003.8000 | 0x4003.8FFF | ADC0 | 519 |
| 0x4003.9000 | 0x4003.9FFF | ADC1 | 519 |
| 0x4003.A000 | 0x4003.BFFF | Reserved | - |
| 0x4003.C000 | 0x4003.CFFF | Analog Comparators | 995 |
| 0x4003.D000 | 0x4003.DFFF | GPIO Port J | 329 |
| 0x4003.E000 | 0x4003.FFFF | Reserved | - |
| 0x4004.0000 | 0x4004.0FFF | CAN0 Controller | 769 |
| 0x4004.1000 | 0x4004.1FFF | CAN1 Controller | 769 |
| 0x4004.2000 | 0x4004.7FFF | Reserved | - |
| 0x4004.8000 | 0x4004.8FFF | Ethernet Controller | 812 |
| 0x4004.9000 | 0x4004.FFFF | Reserved | - |
| 0x4005.0000 | 0x4005.0FFF | USB | 883 |
| 0x4005.1000 | 0x4005.3FFF | Reserved | - |
| 0x4005.4000 | 0x4005.4FFF | I ² S0 | 725 |
| 0x4005.5000 | 0x4005.7FFF | Reserved | - |
| 0x4005.8000 | 0x4005.8FFF | GPIO Port A (AHB aperture) | 329 |
| 0x4005.9000 | 0x4005.9FFF | GPIO Port B (AHB aperture) | 329 |
| 0x4005.A000 | 0x4005.AFFF | GPIO Port C (AHB aperture) | 329 |
| 0x4005.B000 | 0x4005.BFFF | GPIO Port D (AHB aperture) | 329 |
| 0x4005.C000 | 0x4005.CFFF | GPIO Port E (AHB aperture) | 329 |
| 0x4005.D000 | 0x4005.DFFF | GPIO Port F (AHB aperture) | 329 |
| 0x4005.E000 | 0x4005.EFFF | GPIO Port G (AHB aperture) | 329 |
| 0x4005.F000 | 0x4005.FFFF | GPIO Port H (AHB aperture) | 329 |
| 0x4006.0000 | 0x4006.0FFF | GPIO Port J (AHB aperture) | 329 |
| 0x4006.1000 | 0x400C.FFFF | Reserved | - |
| 0x400D.0000 | 0x400D.FFFF | EPI0 | 392 |
| 0x400E.0000 | 0x400F.BFFF | Reserved | - |
| 0x400F.C000 | 0x400F.CFFF | Hibernation Module | 209 |
| 0x400F.D000 | 0x400F.DFFF | Flash memory control | 232 |
| 0x400F.E000 | 0x400F.EFFF | System control | 109 |
| 0x400F.F000 | 0x400F.FFFF | μDMA | 279 |
| 0x4010.0000 | 0x41FF.FFFF | Reserved | - |
| 0x4200.0000 | 0x43FF.FFFF | Bit-banded alias of 0x4000.0000 through 0x400F.FFFF | - |

Table 3-1. Memory Map (continued)

| Start | End | Description | For details, see page ... |
|-------------------------------|-------------|---|--|
| 0x4400.0000 | 0x5FFF.FFFF | Reserved | - |
| 0x6000.0000 | 0xCFFF.FFFF | EPI0 mapped peripheral and RAM | - |
| 0xD000.0000 | 0xDFFF.FFFF | Reserved | - |
| Private Peripheral Bus | | | |
| 0xE000.0000 | 0xE000.0FFF | Instrumentation Trace Macrocell (ITM) | ARM® Cortex™-M3 Technical Reference Manual |
| 0xE000.1000 | 0xE000.1FFF | Data Watchpoint and Trace (DWT) | ARM® Cortex™-M3 Technical Reference Manual |
| 0xE000.2000 | 0xE000.2FFF | Flash Patch and Breakpoint (FPB) | ARM® Cortex™-M3 Technical Reference Manual |
| 0xE000.3000 | 0xE000.DFFF | Reserved | - |
| 0xE000.E000 | 0xE000.EFFF | Nested Vectored Interrupt Controller (NVIC) | ARM® Cortex™-M3 Technical Reference Manual |
| 0xE000.F000 | 0xE003.FFFF | Reserved | - |
| 0xE004.0000 | 0xE004.0FFF | Trace Port Interface Unit (TPIU) | ARM® Cortex™-M3 Technical Reference Manual |
| 0xE004.1000 | 0xFFFF.FFFF | Reserved | - |

4 Interrupts

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 4-1 on page 80 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 47 interrupts (listed in Table 4-2 on page 81).

Priorities on the system handlers are set with the NVIC System Handler Priority registers. Interrupts are enabled through the NVIC Interrupt Set Enable register and prioritized with the NVIC Interrupt Priority registers. Priorities can be grouped by splitting priority levels into pre-emption priorities and subpriorities. All of the interrupt registers are described in Chapter 8, “Nested Vectored Interrupt Controller” in the *ARM® Cortex™-M3 Technical Reference Manual*.

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.

If you assign the same priority level to two or more interrupts, their hardware priority (the lower position number) determines the order in which the processor activates them. For example, if both GPIO Port A and GPIO Port B are priority level 1, then GPIO Port A has higher priority.

Important: It may take several processor cycles after a write to clear an interrupt source 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 Chapter 5, “Exceptions” and Chapter 8, “Nested Vectored Interrupt Controller” in the *ARM® Cortex™-M3 Technical Reference Manual* for more information on exceptions and interrupts.

Table 4-1. Exception Types

| Exception Type | Vector Number | Priority ^a | Description |
|------------------------------|---------------|-----------------------|--|
| - | 0 | - | Stack top is loaded from the first entry of the vector table on reset. |
| Reset | 1 | -3 (highest) | This exception is invoked on power up and warm reset. On the first instruction, Reset drops to the lowest priority (and then is called the base level of activation). This exception is asynchronous. |
| Non-Maskable Interrupt (NMI) | 2 | -2 | This exception is caused by the assertion of the NMI signal or by using the NVIC Interrupt Control State register and cannot be stopped or preempted by any exception but Reset. This exception is asynchronous. |
| Hard Fault | 3 | -1 | This exception is caused by all classes of Fault, when the fault cannot activate due to priority or the configurable fault handler has been disabled. This exception is synchronous. |
| Memory Management | 4 | programmable | This exception is caused by an MPU mismatch, including access violation and no match. This exception is synchronous. |

Table 4-1. Exception Types (continued)

| Exception Type | Vector Number | Priority ^a | Description |
|----------------|---------------|-----------------------|--|
| Bus Fault | 5 | programmable | This exception is caused by a pre-fetch fault, memory access fault, and other address/memory related faults. This exception is synchronous when precise and asynchronous when imprecise. This fault can be enabled or disabled. |
| Usage Fault | 6 | programmable | This exception is caused by a usage fault, such as undefined instruction executed or illegal state transition attempt. This exception is synchronous. |
| - | 7-10 | - | Reserved. |
| SVCall | 11 | programmable | This exception is caused by a system service call with an SVC instruction. This exception is synchronous. |
| Debug Monitor | 12 | programmable | This exception is caused by the debug monitor (when not halting). This exception is synchronous, but only active when enabled. This exception does not activate if it is a lower priority than the current activation. |
| - | 13 | - | Reserved. |
| PendSV | 14 | programmable | This exception is caused by a penable request for system service. This exception is asynchronous and only pended by software. |
| SysTick | 15 | programmable | This exception is caused by the SysTick timer reaching 0, when it is enabled to generate an interrupt. This exception is asynchronous. |
| Interrupts | 16 and above | programmable | This exception is caused by interrupts asserted from outside the ARM Cortex-M3 core and fed through the NVIC (prioritized). These exceptions are all asynchronous. Table 4-2 on page 81 lists the interrupts on the LM3S9B90 controller. |

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

Table 4-2. Interrupts

| Vector Number | Interrupt Number (Bit in Interrupt Registers) | Description |
|---------------|---|-------------------------|
| 0-15 | - | Processor exceptions |
| 16 | 0 | GPIO Port A |
| 17 | 1 | GPIO Port B |
| 18 | 2 | GPIO Port C |
| 19 | 3 | GPIO Port D |
| 20 | 4 | GPIO Port E |
| 21 | 5 | UART0 |
| 22 | 6 | UART1 |
| 23 | 7 | SSI0 |
| 24 | 8 | I ² C0 |
| 25-29 | 9-13 | Reserved |
| 30 | 14 | ADC0 Sequence 0 |
| 31 | 15 | ADC0 Sequence 1 |
| 32 | 16 | ADC0 Sequence 2 |
| 33 | 17 | ADC0 Sequence 3 |
| 34 | 18 | Watchdog Timers 0 and 1 |
| 35 | 19 | Timer 0A |
| 36 | 20 | Timer 0B |

Table 4-2. Interrupts (continued)

| Vector Number | Interrupt Number (Bit in Interrupt Registers) | Description |
|---------------|---|----------------------|
| 37 | 21 | Timer 1A |
| 38 | 22 | Timer 1B |
| 39 | 23 | Timer 2A |
| 40 | 24 | Timer 2B |
| 41 | 25 | Analog Comparator 0 |
| 42 | 26 | Analog Comparator 1 |
| 43 | 27 | Analog Comparator 2 |
| 44 | 28 | System Control |
| 45 | 29 | Flash Memory Control |
| 46 | 30 | GPIO Port F |
| 47 | 31 | GPIO Port G |
| 48 | 32 | GPIO Port H |
| 49 | 33 | UART2 |
| 50 | 34 | SSI1 |
| 51 | 35 | Timer 3A |
| 52 | 36 | Timer 3B |
| 53 | 37 | I ² C1 |
| 54 | 38 | Reserved |
| 55 | 39 | CAN0 |
| 56 | 40 | CAN1 |
| 57 | 41 | Reserved |
| 58 | 42 | Ethernet Controller |
| 59 | 43 | Hibernation Module |
| 60 | 44 | USB |
| 61 | 45 | Reserved |
| 62 | 46 | μDMA Software |
| 63 | 47 | μDMA Error |
| 64 | 48 | ADC1 Sequence 0 |
| 65 | 49 | ADC1 Sequence 1 |
| 66 | 50 | ADC1 Sequence 2 |
| 67 | 51 | ADC1 Sequence 3 |
| 68 | 52 | I ² S0 |
| 69 | 53 | EPI |
| 70 | 54 | GPIO Port J |

5 JTAG Interface

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

The JTAG port is comprised of four pins: TCK, TMS, TDI, and TDO. Data is transmitted serially into the controller on TDI and out of the controller on TDO. The interpretation of this data is dependent on the current state of the TAP controller. For detailed information on the operation of the JTAG port and TAP controller, please refer to the *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*.

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

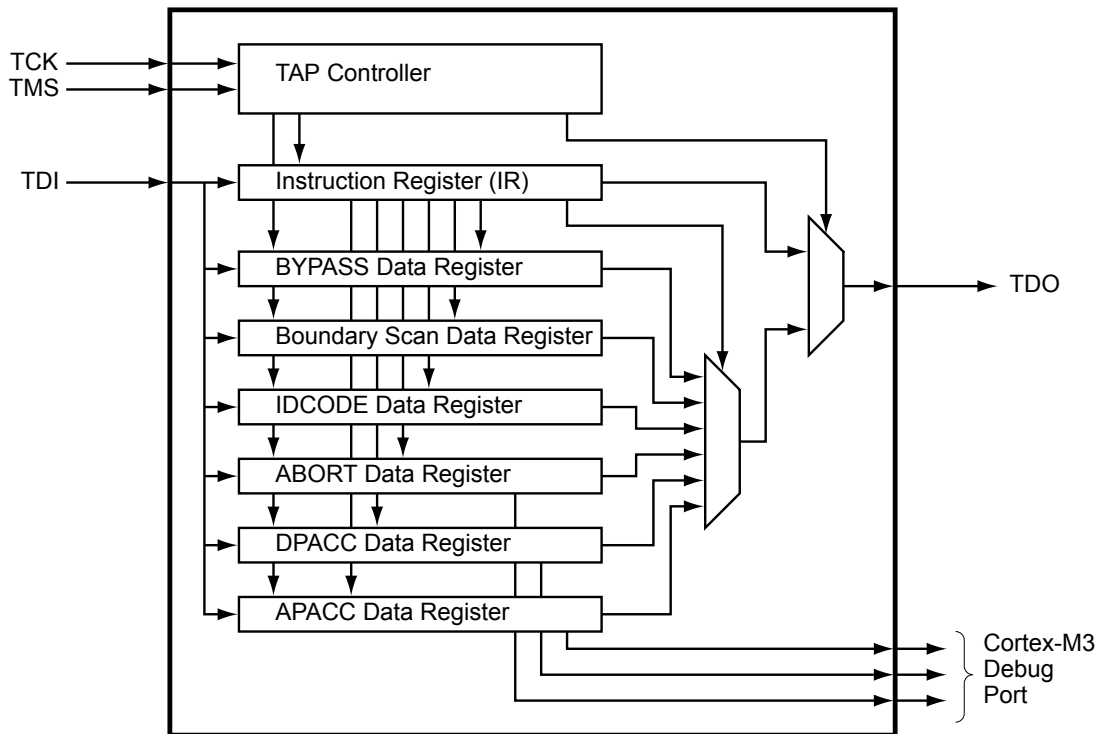
The Stellaris® JTAG module has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
 - Serial Wire JTAG Debug Port (SWJ-DP)
 - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
 - Data Watchpoint and 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

See the *ARM® Cortex™-M3 Technical Reference Manual* for more information on the ARM JTAG controller.

5.1 Block Diagram

Figure 5-1. JTAG Module Block Diagram



5.2 Signal Description

Table 5-1 on page 84 lists the external signals of the JTAG/SWD controller and describes the function of each. The JTAG/SWD controller signals are alternate functions for some GPIO signals, however note that the reset state of the pins is for the JTAG/SWD function. The JTAG/SWD controller signals are under commit protection and require a special process to be configured as GPIOs, see “Commit Control” on page 324. The column in the table below titled “Pin Mux/Pin Assignment” lists the GPIO pin placement for the JTAG/SWD controller signals. The **AFSEL** bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) is set to choose the JTAG/SWD function. The number in parentheses is the encoding that must be programmed into the **PMCn** field in the **GPIO Port Control (GPIOPCTL)** register (page 358) to assign the JTAG/SWD controller signals to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 318.

Table 5-1. Signals for JTAG_SWD_SWO

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

Table 5-1. Signals for JTAG_SWD_SWO (continued)

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

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

5.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 5-1 on page 84. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the TCK and TMS inputs. The current state of the TAP controller depends on the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI towards TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

The serial shift chains with parallel load registers are comprised of a single Instruction Register (IR) chain and multiple Data Register (DR) chains. The current instruction loaded in the parallel load register determines which DR chain is captured, shifted, or updated during the sequencing of the TAP controller.

Some instructions, like EXTEST and INTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 5-3 on page 91 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 1045 for JTAG timing diagrams.

Note: Of all the possible reset sources, only Power-On reset (POR) and the assertion of the $\overline{\text{RST}}$ input have any effect on the JTAG module. The pin configurations are reset by both the $\overline{\text{RST}}$ input and POR, whereas the internal JTAG logic is only reset with POR. See “Reset Sources” on page 96 for more information on reset.

5.3.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. These pins and their associated state after a power-on reset or reset caused by the $\overline{\text{RST}}$ input are given in Table 5-2. Detailed information on each pin follows. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 318 for information on how to reprogram the configuration of these pins.

Table 5-2. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion

| Pin Name | Data Direction | Internal Pull-Up | Internal Pull-Down | Drive Strength | Drive Value |
|----------|----------------|------------------|--------------------|----------------|-------------|
| TCK | Input | Enabled | Disabled | N/A | N/A |
| TMS | Input | Enabled | Disabled | N/A | N/A |
| TDI | Input | Enabled | Disabled | N/A | N/A |
| TDO | Output | Enabled | Disabled | 2-mA driver | High-Z |

5.3.1.1 Test Clock Input (TCK)

The TCK pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks and to ensure that multiple JTAG TAP controllers that are daisy-chained together can synchronously communicate serial test data between components.

During normal operation, TCK is driven by a free-running clock with a nominal 50% duty cycle. When necessary, TCK can be stopped at 0 or 1 for extended periods of time. While TCK is stopped at 0 or 1, the state of the TAP controller does not change and data in the JTAG Instruction and Data Registers is not lost.

By default, the internal pull-up resistor on the TCK pin is enabled after reset, assuring that no clocking occurs if the pin is not driven from an external source. The internal pull-up and pull-down resistors can be turned off to save internal power as long as the TCK pin is constantly being driven by an external source (see page 346 and page 348).

5.3.1.2 Test Mode Select (TMS)

The TMS pin selects the next state of the JTAG TAP controller. TMS is sampled on the rising edge of TCK. Depending on the current TAP state and the sampled value of TMS, the next state may be entered. Because the TMS pin is sampled on the rising edge of TCK, the *IEEE Standard 1149.1* expects the value on TMS to change on the falling edge of TCK.

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 5-2 on page 87.

By default, the internal pull-up resistor on the TMS pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC1/TMS; otherwise JTAG communication could be lost (see page 346).

5.3.1.3 Test Data Input (TDI)

The TDI pin provides a stream of serial information to the IR chain and the DR chains. TDI is sampled on the rising edge of TCK and, depending on the current TAP state and the current instruction, may present this data to the proper shift register chain. Because the TDI pin is sampled on the rising edge of TCK, the *IEEE Standard 1149.1* expects the value on TDI to change on the falling edge of TCK.

By default, the internal pull-up resistor on the TDI pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC2/TDI; otherwise JTAG communication could be lost (see page 346).

5.3.1.4 Test Data Output (TDO)

The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an inactive drive state when not actively shifting out data. Because TDO can be connected to the TDI of another controller in a daisy-chain configuration, the *IEEE Standard 1149.1* expects the value on TDO to change on the falling edge of TCK.

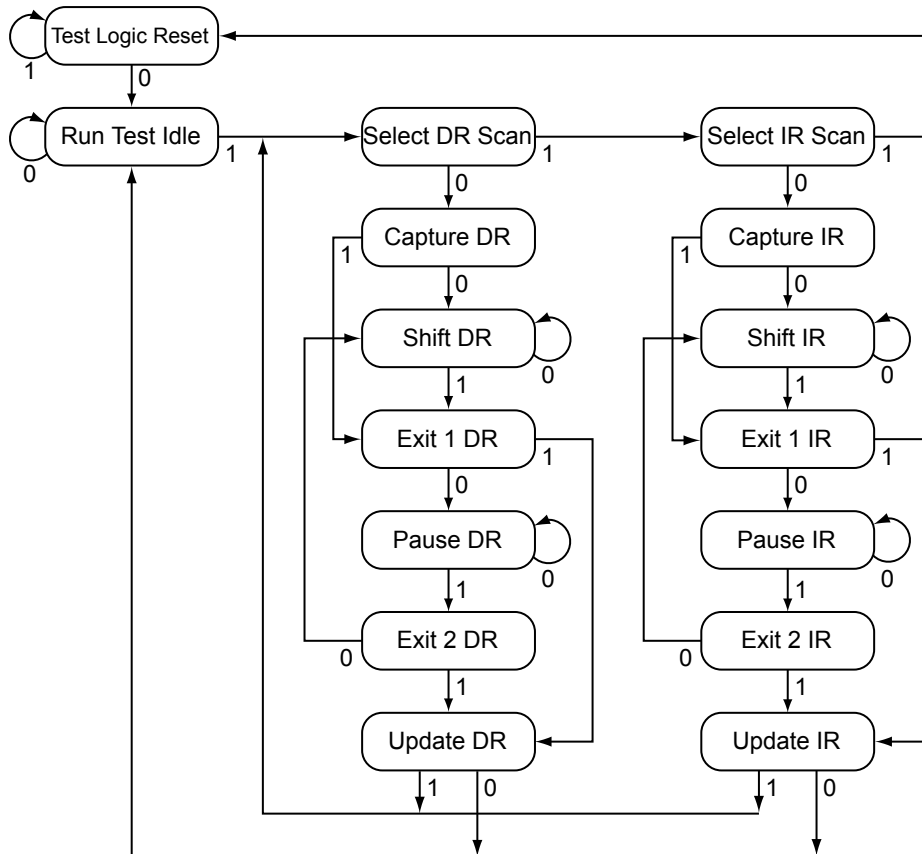
By default, the internal pull-up resistor on the TDO pin is enabled after reset, assuring that the pin remains at a constant logic level when the JTAG port is not being used. The internal pull-up and pull-down resistors can be turned off to save internal power if a High-Z output value is acceptable during certain TAP controller states (see page 346 and page 348).

5.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). In order to reset

the JTAG module after the microcontroller has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 5-2. Test Access Port State Machine



5.3.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out on TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Register Descriptions" on page 90.

5.3.4 Operational Considerations

Certain operational parameters must be considered when using the JTAG module. Because the JTAG pins can be programmed to be GPIOs, board configuration and reset conditions on these pins must be considered. In addition, because the JTAG module has integrated ARM Serial Wire Debug, the method for switching between these two operational modes is described below.

5.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or $\overline{\text{RST}}$, the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality ($\text{DEN}[3:0]$ set in the **Port C GPIO Digital Enable (GPIODEN)** register), enabling the pull-up resistors ($\text{PUE}[3:0]$ set in the **Port C GPIO Pull-Up Select (GPIOPUR)** register), disabling the pull-down resistors ($\text{PDE}[3:0]$ cleared in the **Port C GPIO Pull-Down Select (GPIOPDR)** register) and enabling the alternate hardware function ($\text{AFSEL}[3:0]$ set in the **Port C GPIO Alternate Function Select (GPIOAFSEL)** register) on the JTAG/SWD pins. See page 340, page 346, page 348, and page 351.

It is possible for software to configure these pins as GPIOs after reset by clearing $\text{AFSEL}[3:0]$ in the **Port C GPIOAFSEL** register. If the user does not require the JTAG/SWD port for debugging or board-level testing, this provides four more GPIOs for use in the design.

Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris[®] microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the NMI pin (PB7) and the four JTAG/SWD pins ($\text{PC}[3:0]$). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 340), **GPIO Pull Up Select (GPIOPUR)** register (see page 346), **GPIO Pull-Down Select (GPIOPDR)** register (see page 348), and **GPIO Digital Enable (GPIODEN)** register (see page 351) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 353) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 354) have been set.

5.3.4.2 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.

5.3.4.3 Recovering a "Locked" Microcontroller

Note: Performing the sequence below restores the nonvolatile registers discussed in “Nonvolatile Register Programming” on page 230 to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the nonvolatile registers being restored.

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

1. Assert and hold the $\overline{\text{RST}}$ signal.

2. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence on the section called “JTAG-to-SWD Switching” on page 89.
3. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on the section called “SWD-to-JTAG Switching” on page 90.
4. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
5. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
6. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
7. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
8. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
9. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
10. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
11. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
12. Release the $\overline{\text{RST}}$ signal.
13. Wait 400 ms.
14. Power-cycle the microcontroller.

5.3.4.4 ARM Serial Wire Debug (SWD)

In order to seamlessly integrate the ARM Serial Wire Debug (SWD) functionality, a serial-wire debugger must be able to connect to the Cortex-M3 core without having to perform, or have any knowledge of, JTAG cycles. This integration is accomplished with a SWD preamble that is issued before the SWD session begins.

The switching preamble used to enable the SWD interface of the SWJ-DP module starts with the TAP controller in the Test-Logic-Reset state. From here, the preamble sequences the TAP controller through the following states: Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, and Test Logic Reset states.

Stepping through this sequence of the TAP state machine enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the *ARM® Cortex™-M3 Technical Reference Manual* and the *ARM® CoreSight Technical Reference Manual*.

Because this sequence is a valid series of JTAG operations that could be issued, the ARM JTAG TAP controller is not fully compliant to the *IEEE Standard 1149.1*. This instance is the only one where the ARM JTAG TAP controller does not meet full compliance with the specification. Due to the low probability of this sequence occurring during normal operation of the TAP controller, it should not affect normal performance of the JTAG interface.

JTAG-to-SWD Switching

To switch the operating mode of the Debug Access Port (DAP) from JTAG to SWD mode, the external debug hardware must send the switching preamble to the microcontroller. The 16-bit TMS command for switching to SWD mode is defined as b1110.0111.1001.1110, transmitted LSB first.

This command can also be represented as 0xE79E when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

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

SWD-to-JTAG Switching

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

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

5.4 Initialization and Configuration

After a Power-On-Reset or an external reset ($\overline{\text{RST}}$), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. To return the pins to their JTAG functions, enable the four JTAG pins (PC[3:0]) for their alternate function using the **GPIOAFSEL** register. In addition to enabling the alternate functions, any other changes to the GPIO pad configurations on the four JTAG pins (PC[3:0]) should be returned to their default settings.

5.5 Register Descriptions

The registers in the JTAG TAP Controller or Shift Register chains are not memory mapped and are not accessible through the on-chip Advanced Peripheral Bus (APB). Instead, the registers within the JTAG controller are all accessed serially through the TAP Controller. These registers include the Instruction Register and the six Data Registers.

5.5.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG TDI and TDO pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the IR. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the IR bits is shown in Table 5-3. A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 5-3. JTAG Instruction Register Commands

| IR[3:0] | Instruction | Description |
|------------|------------------|--|
| 0x0 | EXTEST | Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction onto the pads. |
| 0x1 | INTEST | Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction into the controller. |
| 0x2 | SAMPLE / PRELOAD | Captures the current I/O values and shifts the sampled values out of the Boundary Scan Chain while new preload data is shifted in. |
| 0x8 | ABORT | Shifts data into the ARM Debug Port Abort Register. |
| 0xA | DPACC | Shifts data into and out of the ARM DP Access Register. |
| 0xB | APACC | Shifts data into and out of the ARM AC Access Register. |
| 0xE | IDCODE | Loads manufacturing information defined by the <i>IEEE Standard 1149.1</i> into the IDCODE chain and shifts it out. |
| 0xF | BYPASS | Connects TDI to TDO through a single Shift Register chain. |
| All Others | Reserved | Defaults to the BYPASS instruction to ensure that TDI is always connected to TDO. |

5.5.1.1 EXTEST Instruction

The EXTEST instruction is not associated with its own Data Register chain. Instead, the EXTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the EXTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the outputs and output enables are used to drive the GPIO pads rather than the signals coming from the core. With tests that drive known values out of the controller, this instruction can be used to verify connectivity. While the EXTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

5.5.1.2 INTEST Instruction

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

5.5.1.3 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction connects the Boundary Scan Data Register chain between TDI and TDO. This instruction samples the current state of the pad pins for observation and preloads new test data. Each GPIO pad has an associated input, output, and output enable signal. When the TAP controller enters the Capture DR state during this instruction, the input, output, and output-enable signals to each of the GPIO pads are captured. These samples are serially shifted out on TDO while the TAP controller is in the Shift DR state and can be used for observation or comparison in various tests.

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST and INTEST instructions to drive data into or out of the controller. See “Boundary Scan Data Register” on page 93 for more information.

5.5.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. See the “ABORT Data Register” on page 94 for more information.

5.5.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. See “DPACC Data Register” on page 94 for more information.

5.5.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. See “APACC Data Register” on page 94 for more information.

5.5.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure input and output data streams. IDCODE is the default instruction loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, or the Test-Logic-Reset state is entered. See “IDCODE Data Register” on page 93 for more information.

5.5.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. See “BYPASS Data Register” on page 93 for more information.

5.5.2 Data Registers

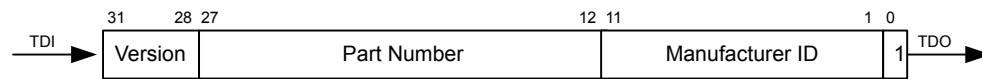
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

5.5.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-3. The standard requires that every JTAG-compliant microcontroller implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

The major uses of the JTAG port are for manufacturer testing of component assembly and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x412F.C230. This value indicates an ARM Cortex-M3, Version 2 processor and allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

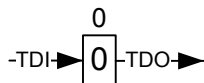
Figure 5-3. IDCODE Register Format



5.5.2.2 BYPASS Data Register

The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-4. The standard requires that every JTAG-compliant microcontroller implement either the BYPASS instruction or the IDCODE instruction as the default instruction. The LSB of the BYPASS Data Register is defined to be a 0 to distinguish it from the IDCODE instruction, which has an LSB of 1. This definition allows auto-configuration test tools to determine which instruction is the default instruction.

Figure 5-4. BYPASS Register Format

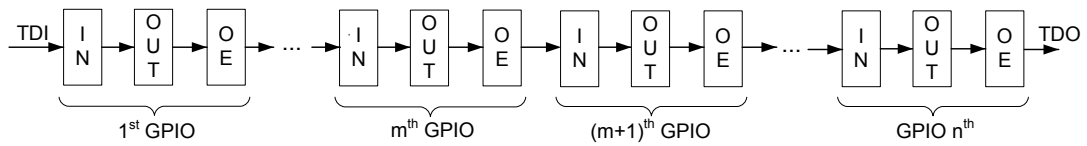


5.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 5-5. Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as shown in the figure.

When the Boundary Scan Data Register is accessed with the SAMPLE/PRELOAD instruction, the input, output, and output enable from each digital pad are sampled and then shifted out of the chain to be verified. The sampling of these values occurs on the rising edge of TCK in the Capture DR state of the TAP controller. While the sampled data is being shifted out of the Boundary Scan chain in the Shift DR state of the TAP controller, new data can be preloaded into the chain for use with the EXTEST and INTEST instructions. The EXTEST instruction forces data out of the controller, and the INTEST instruction forces data into the controller.

Figure 5-5. Boundary Scan Register Format



5.5.2.4 APACC Data Register

The format for the 35-bit APACC Data Register defined by ARM is described in the *ARM® Cortex™-M3 Technical Reference Manual*.

5.5.2.5 DPACC Data Register

The format for the 35-bit DPACC Data Register defined by ARM is described in the *ARM® Cortex™-M3 Technical Reference Manual*.

5.5.2.6 ABORT Data Register

The format for the 35-bit ABORT Data Register defined by ARM is described in the *ARM® Cortex™-M3 Technical Reference Manual*.

6 System Control

System control configures the overall operation of the device and provides information about the device. Configurable features include reset control, NMI operation, power control, clock control, and low-power modes.

6.1 Signal Description

Table 6-1 on page 95 lists the external signals of the System Control module and describes the function of each. The NMI signal is the alternate function for the GPIO PB7 signal and functions as a GPIO after reset. PB7 is under commit protection and requires a special process to be configured as the NMI signal or to subsequently return to the GPIO function, see “Commit Control” on page 324. The column in the table below titled “Pin Mux/Pin Assignment” lists the GPIO pin placement for the NMI signal. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the NMI function. The number in parentheses is the encoding that must be programmed into the PMCN field in the **GPIO Port Control (GPIOPCTL)** register (page 358) to assign the NMI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 318. The remaining signals (with the word “fixed” in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 6-1. Signals for System Control & Clocks

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|---|
| NMI | 89 | PB7 (4) | I | TTL | Non-maskable interrupt. |
| OSC0 | 48 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | 49 | fixed | O | Analog | Main oscillator crystal output. |
| RST | 64 | fixed | I | TTL | System reset input. |

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

6.2 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 95
- Local control, such as reset (see “Reset Control” on page 95), power (see “Power Control” on page 100) and clock control (see “Clock Control” on page 101)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 106

6.2.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, Flash memory size, and other features. See the **DID0** (page 110), **DID1** (page 141), **DC0-DC9** (page 143) and **NVMSTAT** (page 165) registers.

6.2.2 Reset Control

This section discusses aspects of hardware functions during reset as well as system software requirements following the reset sequence.

6.2.2.1 Reset Sources

The LM3S9B90 microcontroller has six sources of reset:

1. Power-on reset (POR) (see page 96).
2. External reset input pin ($\overline{\text{RST}}$) assertion (see page 97).
3. Internal brown-out (BOR) detector (see page 98).
4. Software-initiated reset (with the software reset registers) (see page 98).
5. A watchdog timer reset condition violation (see page 99).
6. MOSC failure (see page 100).

Table 6-2 provides a summary of results of the various reset operations.

Table 6-2. 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 Reset | Yes ^a | No | Yes ^b |
| Watchdog Reset | Yes | No | Yes |
| MOSC Failure Reset | Yes | No | Yes |

a. By using the SYSRESETREQ bit in the ARM Cortex-M3 Application Interrupt and Reset Control 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 is the cause, in which case, all the bits in the **RESC** register are cleared except for the POR indicator. A bit in the **RESC** register can be cleared by writing a 0.

6.2.2.2 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 $\overline{\text{RST}}$ input may be used as discussed in “External $\overline{\text{RST}}$ Pin” on page 97.

The Power-On Reset sequence is as follows:

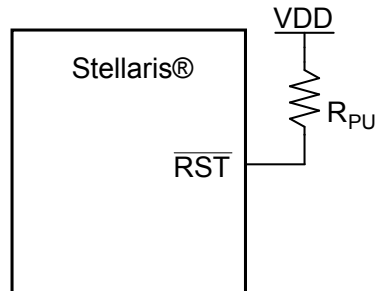
1. The microcontroller waits for internal POR to go inactive.
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in Figure 26-5 on page 1047.

6.2.2.3 External $\overline{\text{RST}}$ Pin

If the application only uses the internal POR circuit, the $\overline{\text{RST}}$ input must be connected to the power supply (V_{DD}) through an optional pull-up resistor (0 to 10K Ω) as shown in Figure 6-1 on page 97.

Figure 6-1. Basic $\overline{\text{RST}}$ Configuration



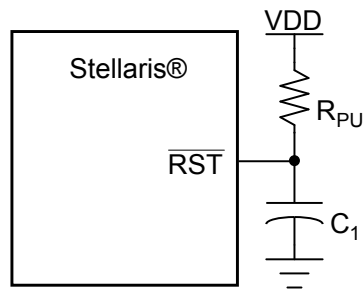
$R_{\text{PU}} = 0$ to 100 k Ω

The external reset pin ($\overline{\text{RST}}$) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see “JTAG Interface” on page 83). The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted for the duration specified by T_{MIN} and then de-asserted (see “Reset” on page 1046).
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

To improve noise immunity and/or to delay reset at power up, the $\overline{\text{RST}}$ input may be connected to an RC network as shown in Figure 6-2 on page 97.

Figure 6-2. External Circuitry to Extend Power-On Reset

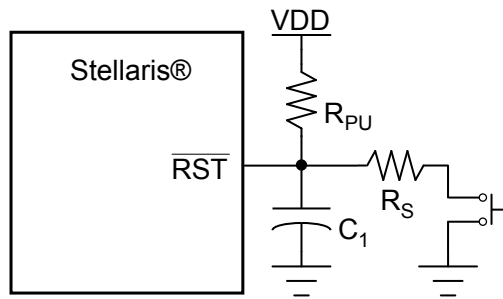


$R_{\text{PU}} = 1$ k Ω to 100 k Ω

$C_1 = 1$ nF to 10 μF

If the application requires the use of an external reset switch, Figure 6-3 on page 98 shows the proper circuitry to use.

Figure 6-3. Reset Circuit Controlled by Switch



Typical $R_{PU} = 10\text{ k}\Omega$

Typical $R_S = 470\ \Omega$

$C_1 = 10\text{ nF}$

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

The external reset timing is shown in Figure 26-4 on page 1046.

6.2.2.4 Brown-Out Reset (BOR)

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

The result of a brown-out reset is equivalent to that of an assertion of the external \overline{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 26-6 on page 1047.

6.2.2.5 Software Reset

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

Peripherals can be individually reset by software via three registers that control reset signals to each on-chip peripheral (see the **SRCRn** registers, page 193). 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 106).

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

1. A software microcontroller reset is initiated by setting 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 microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The software-initiated system reset timing is shown in Figure 26-7 on page 1047.

6.2.2.6 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The LM3S9B90 microcontroller has two Watchdog Timer modules in case one watchdog clock source fails. One watchdog is run off the system clock and the other is run off the Precision Internal Oscillator (PIOSC). Each module operates in the same manner except that because the PIOSC watchdog timer module is in a different clock domain, register accesses must have a time delay between them. The watchdog timer can be configured to generate an interrupt to the microcontroller on its first time-out and to generate a reset on its second time-out.

After the watchdog's first time-out event, the 32-bit watchdog counter is reloaded with the value of the **Watchdog Timer Load (WDTLOAD)** register and resumes counting down from that value. If the timer counts down to zero again before the first time-out interrupt is cleared, and the reset signal has been enabled, the watchdog timer asserts its reset signal to the microcontroller. The watchdog timer reset sequence is as follows:

1. The watchdog timer times out for the second time without being serviced.
2. An internal reset is asserted.
3. The internal reset is released and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

For more information on the Watchdog Timer module, see "Watchdog Timers" on page 476.

The watchdog reset timing is shown in Figure 26-8 on page 1047.

6.2.3 Non-Maskable Interrupt

The microcontroller has two sources of non-maskable interrupt (NMI):

- The assertion of the NMI signal
- A main oscillator verification error

If both sources of NMI are enabled, software must check that the main oscillator verification is the cause of the interrupt in order to distinguish between the two sources.

6.2.3.1 NMI Pin

The alternate function to GPIO port pin B7 is an NMI signal. The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in "General-Purpose Input/Outputs (GPIOs)" on page 318. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function just like the GPIO port pins associated with JTAG/SWD functionality, see page 354. The active sense of the NMI signal is High; asserting the enabled NMI signal above V_{IH} initiates the NMI interrupt sequence.

6.2.3.2 Main Oscillator Verification Failure

The LM3S9B90 microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or too slow. The main oscillator verification circuit can be programmed to generate a reset event, at which time a Power-on Reset is generated and control is transferred to the NMI handler. The NMI handler is used to address the main oscillator verification failure because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled by setting the `CVAL` bit in the **Main Oscillator Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status (`MOSCFAIL`) bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in “Main Oscillator Verification Circuit” on page 105.

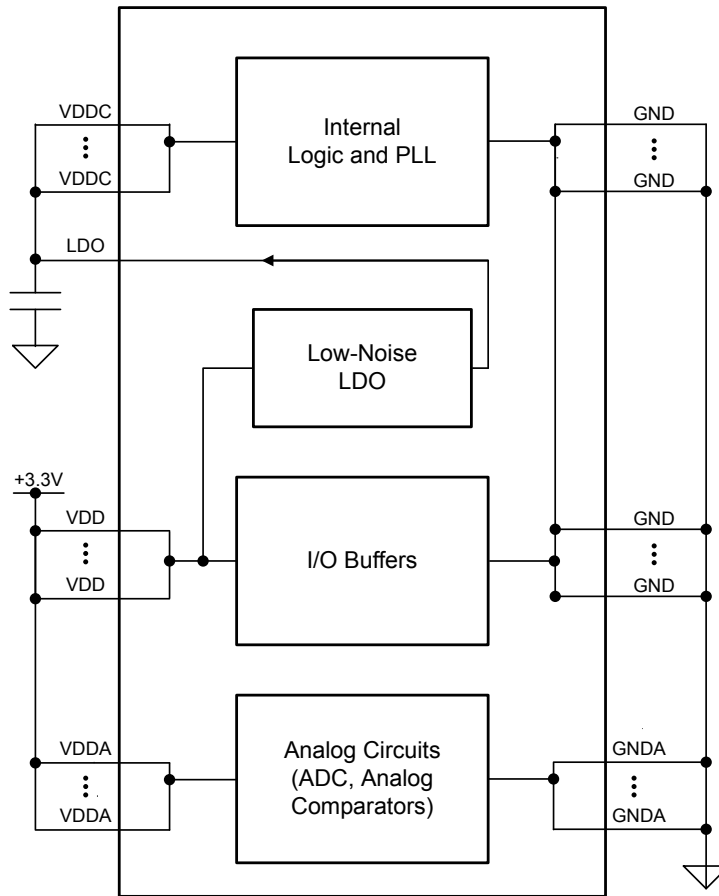
6.2.4 Power Control

The Stellaris[®] microcontroller provides an integrated LDO regulator that may be used to provide power to the majority of the microcontroller's internal logic. For power reduction, a non-programmable LDO may be used to scale the microcontroller's 3.3 V input voltage to 1.2V. The voltage output has a minimum voltage of 1.08 V and a maximum of 1.35 V. The LDO delivers up to 60 ma.

Figure 6-4 shows the power architecture.

Note: On the printed circuit board, use the `LDO` output as the source of `VDDC` input. In addition, the LDO requires decoupling capacitors. See “On-Chip Low Drop-Out (LDO) Regulator Characteristics” on page 1040.

Figure 6-4. Power Architecture



6.2.5 Clock Control

System control determines the control of clocks in this part.

6.2.5.1 Fundamental Clock Sources

There are multiple clock sources for use in the microcontroller:

- **Precision Internal Oscillator (PIOSC).** The precision internal oscillator is an on-chip clock source that is the clock source the microcontroller uses during and following POR. It does not require the use of any external components and provides a clock that is 16 MHz \pm 1% at room temperature and \pm 3% across temperature. The PIOSC allows for a reduced system cost in applications that require an accurate clock source. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference. If the Hibernation Module clock source is a 32.768-kHz oscillator, the precision internal oscillator can be trimmed by software based on a reference clock for increased accuracy.
- **Main Oscillator (MOSC).** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 3.579545 MHz through

16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 16.384 MHz. The single-ended clock source range is from DC through the specified speed of the microcontroller. The supported crystals are listed in the `XTAL` bit field in the **RCC** register (see page 121). Note that the MOSC must have a clock source for the USB PLL.

- **Internal 30-kHz Oscillator.** The internal 30-kHz oscillator provides an operational frequency of 30 kHz \pm 50%. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the MOSC and PIOSC to be powered down.
- **Hibernation Module Clock Source.** The Hibernation module can be clocked in one of two ways. The first way is a 4.194304-MHz crystal connected to the `XOSC0` and `XOSC1` pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. The second way is a 32.768-kHz oscillator connected to the `XOSC0` pin. The clock source for the Hibernation module can be used for the system clock, thus eliminating the need for an additional crystal or oscillator. In addition, a 4.194304-MHz crystal can also be a source for the PLL. The Hibernation module clock source is intended to provide the system with a real-time clock source and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL and the precision internal oscillator divided by four (4 MHz \pm 1%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 16.384 MHz (inclusive). Table 6-3 on page 102 shows how the various clock sources can be used in a system.

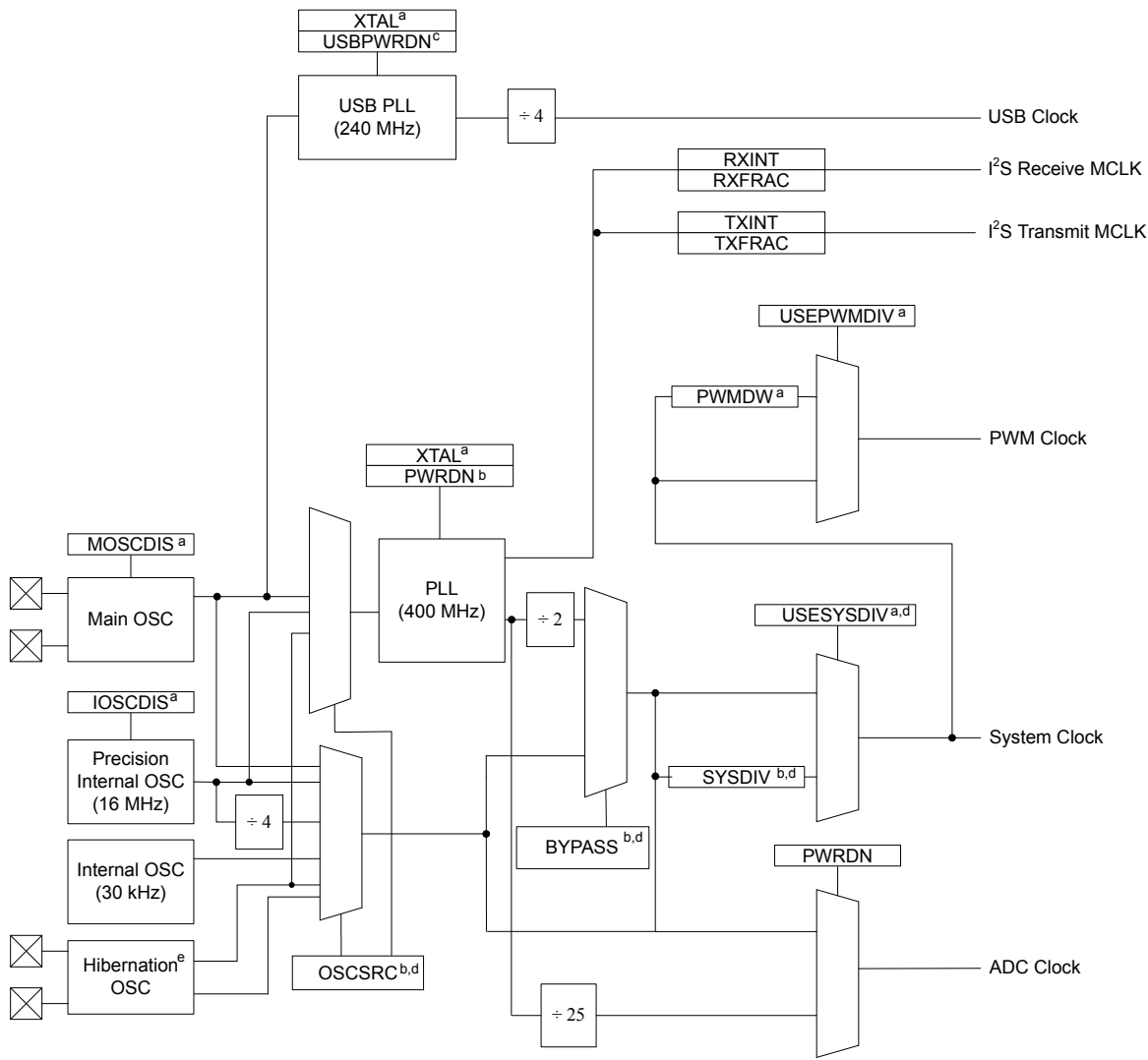
Table 6-3. Clock Source Options

| Clock Source | Drive PLL? | | Used as SysClk? | |
|--|------------|---------------------------|-----------------|---------------------------|
| | Yes | No | Yes | No |
| Precision Internal Oscillator | Yes | BYPASS = 0, OSCSRC = 0x1 | Yes | BYPASS = 1, OSCSRC = 0x1 |
| Precision Internal Oscillator divide by 4 (4 MHz \pm 1%) | 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 |
| Hibernation Module 4.194304-MHz Crystal | Yes | BYPASS = 0, OSCSRC2 = 0x7 | Yes | BYPASS = 1, OSCSRC2 = 0x6 |
| Hibernation Module 32.768-kHz Oscillator | No | BYPASS = 1 | Yes | BYPASS = 1, OSCSRC2 = 0x7 |

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.

Figure 6-5 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled. The ADC clock signal is automatically divided down to 16 MHz for proper ADC operation.

Figure 6-5. Main Clock Tree



- a. Control provided by RCC register bit/field.
 b. Control provided by RCC register bit/field or RCC2 register bit/field, if overridden with RCC2 register bit USERCC2.
 c. Control provided by RCC2 register bit/field.
 d. Also may be controlled by DSLPCLKCFG when in deep sleep mode.
 e. Only a 4.194304-Mhz crystal can be used to drive the PLL.

Note: The figure above shows all features available on all Stellaris® Tempest-class microcontrollers.

6.2.5.2 Precision Internal Oscillator Operation (PIOSC)

The microcontroller powers up with the PIOSC running. If another clock source is desired, the PIOSC can be powered down by setting the `IOSCDIS` bit in the `RCC` register.

The PIOSC generates a 16 MHz clock with a $\pm 1\%$ accuracy at room temperatures. Across the extended temperature range, the accuracy is $\pm 3\%$. At the factory, the PIOSC is set to 16 MHz at room temperature, however, the frequency can be trimmed for other voltage or temperature conditions using software in one of three ways:

- Default calibration: clear the `UTEN` bit and set the `UPDATE` bit in the **Precision Internal Oscillator Calibration (PIOSCCAL)** register.
- User-defined calibration: The user can program the `UT` value to adjust the PIOSC frequency. As the `UT` value increases, the generated period increases. To commit a new `UT` value, first set the `UTEN` bit, then program the `UT` field, and then set the `UPDATE` bit. The adjustment finishes within a few clock periods and is glitch free.
- Automatic calibration using the enable 32-kHz oscillator from the Hibernation module: set the `CAL` bit; the results of the calibration are shown in the `RESULT` field in the **Precision Internal Oscillator Statistic (PIOSCSTAT)** register. After calibration is complete, the PIOSC is trimmed using trimmed value returned in the `CT` field.

6.2.5.3 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 16.384 MHz, otherwise, the range of supported crystals is 1 to 16.384 MHz.

The `XTAL` bit in the **RCC** register (see page 121) 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.

6.2.5.4 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. Table 26-11 on page 1043 shows the actual PLL frequency and error for a given crystal choice.

To configure the PIOSC to be the clock source for the main PLL, program the `OSCR2` field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x1.

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 126). The internal translation provides a translation within $\pm 1\%$ of the targeted PLL VCO frequency.

To configure the Hibernation module clock source as the PLL input reference, program the `OSCR2` field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x6 for a 4.194304-MHz crystal or 0x7 for an external 32.768-kHz oscillator.

The Crystal Value field (`XTAL`) in the **Run-Mode Clock Configuration (RCC)** register (see page 121) 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.

6.2.5.5 USB PLL Frequency Configuration

The USB PLL is disabled by default during power-on reset and is enabled later by software. The USB PLL must be enabled and running for proper USB function. The main oscillator is the only clock reference for the USB PLL. The USB PLL is enabled by clearing the `USBPWRDN` bit of the **RCC2** register. The `XTAL` bit field (Crystal Value) of the **RCC** register describes the available crystal choices. The main oscillator must be connected to one of the following crystal values in order to correctly

generate the USB clock: 4, 5, 6, 8, 10, 12, or 16 MHz. Only these crystals provide the necessary USB PLL VCO frequency to conform with the USB timing specifications.

6.2.5.6 PLL Modes

Both PLLs have 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 121 and page 129).

6.2.5.7 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 26-10 on page 1042). During the relock time, the affected PLL is not usable as a clock reference.

Either 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). When the XTAL value is greater than 0x0F, the down counter is set to 0x2400 to maintain the required lock time on higher frequency crystal inputs. Hardware is provided to keep the PLL from being used as a system clock until the T_{READY} condition is met after one of the two changes above. It is the user's responsibility to have a stable clock source (like the main oscillator) before the **RCC/RCC2** register is switched to use the PLL.

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

The USB PLL is not protected during the lock time (T_{READY}), and software should ensure that the USB PLL has locked before using the interface. Software can use many methods to ensure the T_{READY} period has passed, including periodically polling the **USBPLLLRIS** bit in the **Raw Interrupt Status (RIS)** register, and enabling the USB PLL Lock interrupt.

6.2.5.8 Main Oscillator Verification Circuit

The clock control includes circuitry to ensure that the main oscillator is running at the appropriate frequency. The circuit monitors the main oscillator frequency and signals if the frequency is outside of the allowable band of attached crystals.

The detection circuit is enabled using the **CVAL** bit in the **Main Oscillator Control (MOSCCTL)** register. If this circuit is enabled and detects an error, the following sequence is performed by the hardware:

1. The **MOSCFAIL** bit in the **Reset Cause (RESC)** register is set.

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

6.2.6 System Control

For power-savings purposes, the **RCGCn**, **SCGCn**, and **DCGCn** registers control the clock gating logic for each peripheral or block in the system while the microcontroller is in Run, Sleep, and Deep-Sleep mode, respectively. The **DC1**, **DC2** and **DC4** registers act as a write mask for the **RCGCn**, **SCGCn**, and **DCGCn** registers.

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

- **Run Mode.** In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.
- **Sleep Mode.** In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M3 core executing a WFI (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* 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 microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first 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 brings the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* 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 specified in the **DSLCLKCFG** register. When the **DSLCLKCFG** register is used, the internal oscillator source is powered up, if necessary, and other clocks are powered down. If the PLL is running at the time of the WFI instruction, hardware powers the PLL down and overrides the **SYSDIV** field of the active **RCC/RCC2** register, to be determined by the **DSDIVORIDE** setting in the **DSLCLKCFG** register, up to /16 or /64 respectively. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration. If the PIOSC or the 4.194304-MHz Hibernation module clock source is used as the PLL reference clock source, it may continue to provide the clock during Deep-Sleep. See page 133.

- **Hibernate Mode.** In this mode, the power supplies are turned off to the main part of the microcontroller and only the Hibernation module's circuitry is active. An external wake event or RTC event is required to bring the microcontroller back to Run mode. The Cortex-M3 processor and peripherals outside of the Hibernation module see a normal "power on" sequence and the processor starts running code. Software can determine if the microcontroller has been restarted from Hibernate mode by inspecting the Hibernation module registers.

6.3 Initialization and Configuration

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

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

6.4 Register Map

Table 6-4 on page 107 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

Note: Spaces in the System Control register space that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

Additional Flash and ROM registers defined in the System Control register space are described in the "Internal Memory" on page 226.

Table 6-4. System Control Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|------|------|-------------|-------------------------|----------|
| 0x000 | DID0 | RO | - | Device Identification 0 | 110 |
| 0x004 | DID1 | RO | - | Device Identification 1 | 141 |
| 0x008 | DC0 | RO | 0x017F.007F | Device Capabilities 0 | 143 |
| 0x010 | DC1 | RO | - | Device Capabilities 1 | 144 |
| 0x014 | DC2 | RO | 0x570F.5037 | Device Capabilities 2 | 147 |
| 0x018 | DC3 | RO | 0xBFFF.7FC0 | Device Capabilities 3 | 149 |

Table 6-4. System Control Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|------------|-------|-------------|---|----------|
| 0x01C | DC4 | RO | 0x5004.F1FF | Device Capabilities 4 | 152 |
| 0x020 | DC5 | RO | 0x0000.0000 | Device Capabilities 5 | 154 |
| 0x024 | DC6 | RO | 0x0000.0013 | Device Capabilities 6 | 155 |
| 0x028 | DC7 | RO | 0xFFFF.FFFF | Device Capabilities 7 | 156 |
| 0x02C | DC8 | RO | 0xFFFF.FFFF | Device Capabilities 8 ADC Channels | 160 |
| 0x030 | PBORCTL | R/W | 0x0000.7FFD | Brown-Out Reset Control | 112 |
| 0x040 | SRCR0 | R/W | 0x00000000 | Software Reset Control 0 | 193 |
| 0x044 | SRCR1 | R/W | 0x00000000 | Software Reset Control 1 | 195 |
| 0x048 | SRCR2 | R/W | 0x00000000 | Software Reset Control 2 | 198 |
| 0x050 | RIS | RO | 0x0000.0000 | Raw Interrupt Status | 113 |
| 0x054 | IMC | R/W | 0x0000.0000 | Interrupt Mask Control | 115 |
| 0x058 | MISC | R/W1C | 0x0000.0000 | Masked Interrupt Status and Clear | 117 |
| 0x05C | RESC | R/W | - | Reset Cause | 119 |
| 0x060 | RCC | R/W | 0x0780.3AD1 | Run-Mode Clock Configuration | 121 |
| 0x064 | PLLCFG | RO | - | XTAL to PLL Translation | 126 |
| 0x06C | GPIOHBCTL | R/W | 0x0000.0000 | GPIO Host-Bus Control | 127 |
| 0x070 | RCC2 | R/W | 0x0780.6810 | Run-Mode Clock Configuration 2 | 129 |
| 0x07C | MOSCCTL | R/W | 0x0000.0000 | Main Oscillator Control | 132 |
| 0x100 | RCGC0 | R/W | 0x00000040 | Run Mode Clock Gating Control Register 0 | 166 |
| 0x104 | RCGC1 | R/W | 0x00000000 | Run Mode Clock Gating Control Register 1 | 175 |
| 0x108 | RCGC2 | R/W | 0x00000000 | Run Mode Clock Gating Control Register 2 | 184 |
| 0x110 | SCGC0 | R/W | 0x00000040 | Sleep Mode Clock Gating Control Register 0 | 169 |
| 0x114 | SCGC1 | R/W | 0x00000000 | Sleep Mode Clock Gating Control Register 1 | 178 |
| 0x118 | SCGC2 | R/W | 0x00000000 | Sleep Mode Clock Gating Control Register 2 | 187 |
| 0x120 | DCGC0 | R/W | 0x00000040 | Deep Sleep Mode Clock Gating Control Register 0 | 172 |
| 0x124 | DCGC1 | R/W | 0x00000000 | Deep-Sleep Mode Clock Gating Control Register 1 | 181 |
| 0x128 | DCGC2 | R/W | 0x00000000 | Deep Sleep Mode Clock Gating Control Register 2 | 190 |
| 0x144 | DSLCLKCFG | R/W | 0x0780.0000 | Deep Sleep Clock Configuration | 133 |
| 0x14C | DSFLASHCFG | R/W | 0x0000.0000 | Deep Sleep Flash Configuration | 135 |
| 0x150 | PIOSCCAL | R/W | 0x0000.0000 | Precision Internal Oscillator Calibration | 136 |
| 0x154 | PIOSCSTAT | RO | 0x0000.0040 | Precision Internal Oscillator Statistics | 138 |
| 0x170 | I2SMCLKCFG | R/W | 0x0000.0000 | I2S MCLK Configuration | 139 |

Table 6-4. System Control Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|---------|------|-------------|---|----------|
| 0x190 | DC9 | RO | 0x00FF.00FF | Device Capabilities 9 ADC Digital Comparators | 163 |
| 0x1A0 | NVMSTAT | RO | 0x0000.0001 | Non-Volatile Memory Information | 165 |

6.5 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.

Device Identification 0 (DID0)

Base 0x400F.E000

Offset 0x000

Type RO, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|----|----|----------|----|----|----|-------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | VER | | | reserved | | | | CLASS | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | MAJOR | | | | | | | | MINOR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | 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 | <p>DID0 Version</p> <p>This field defines the DID0 register format version. The version number is numeric. The value of the <code>VER</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>Second version of the DID0 register format.</td> </tr> </tbody> </table> | Value | Description | 0x1 | Second version of the DID0 register format. |
| 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 | 0x04 | <p>Device Class</p> <p>The <code>CLASS</code> field value identifies the internal design from which all mask sets are generated for all microcontrollers in a particular product line. The <code>CLASS</code> field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the <code>MAJOR</code> or <code>MINOR</code> fields require differentiation from prior microcontrollers. The value of the <code>CLASS</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x04</td> <td>Stellaris® Tempest-class microcontrollers</td> </tr> </tbody> </table> | Value | Description | 0x04 | Stellaris® Tempest-class microcontrollers |
| Value | Description | | | | | | | |
| 0x04 | Stellaris® Tempest-class microcontrollers | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | |
|-----------|---|------|-------|---|-------|-------------|-----|---|-----|--|-----|---|
| 15:8 | MAJOR | RO | - | <p>Major Revision</p> <p>This field specifies the major revision number of the microcontroller. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, and so on). This field is encoded as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Revision A (initial device)</td> </tr> <tr> <td>0x1</td> <td>Revision B (first base layer revision)</td> </tr> <tr> <td>0x2</td> <td>Revision C (second base layer revision)</td> </tr> </tbody> </table> <p>and so on.</p> | Value | Description | 0x0 | Revision A (initial device) | 0x1 | Revision B (first base layer revision) | 0x2 | Revision C (second base layer revision) |
| Value | Description | | | | | | | | | | | |
| 0x0 | Revision A (initial device) | | | | | | | | | | | |
| 0x1 | Revision B (first base layer revision) | | | | | | | | | | | |
| 0x2 | Revision C (second base layer revision) | | | | | | | | | | | |
| 7:0 | MINOR | RO | - | <p>Minor Revision</p> <p>This field specifies the minor revision number of the microcontroller. The minor revision reflects changes to the metal layers of the design. The MINOR field value is reset when the MAJOR field is changed. This field is numeric and is encoded as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Initial device, or a major revision update.</td> </tr> <tr> <td>0x1</td> <td>First metal layer change.</td> </tr> <tr> <td>0x2</td> <td>Second metal layer change.</td> </tr> </tbody> </table> <p>and so on.</p> | Value | Description | 0x0 | Initial device, or a major revision update. | 0x1 | First metal layer change. | 0x2 | Second metal layer change. |
| Value | Description | | | | | | | | | | | |
| 0x0 | Initial device, or a major revision update. | | | | | | | | | | | |
| 0x1 | First metal layer change. | | | | | | | | | | | |
| 0x2 | Second metal layer change. | | | | | | | | | | | |

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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|----------|
| | 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 | | | | | | | | | | | | | | | BORIOR | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | BORIOR | R/W | 0 | BOR Interrupt or Reset |
| | | | | Value Description |
| | | | 0 | A Brown Out Event causes an interrupt to be generated to the interrupt controller. |
| | | | 1 | A Brown Out Event causes a reset of the microcontroller. |
| 0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 3: Raw Interrupt Status (RIS), offset 0x050

This register indicates the status for system control raw interrupts. An interrupt is sent to the interrupt controller if the corresponding bit in the **Interrupt Mask Control (IMC)** register is set. Writing a 1 to the corresponding bit in the **Masked Interrupt Status and Clear (MISC)** register clears an interrupt status bit.

Raw Interrupt Status (RIS)

Base 0x400F.E000

Offset 0x050

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|------------|----|----|----|------------|---------|----------|----|----|----|--------|----------|
| | 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 | | | | MOSCPUPRIS | | | | USBPLLRLIS | PLLLRIS | reserved | | | | BORRIS | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------------|------|-----------|--|
| 31:9 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | MOSCPUPRIS | RO | 0 | MOSC Power Up Raw Interrupt Status Value Description 1 Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by T_{MOSC_SETTLE} . 0 Sufficient time has not passed for the MOSC to reach the expected frequency. This bit is cleared by writing a 1 to the MOSCPUPMIS bit in the MISC register. |
| 7 | USBPLLRLIS | RO | 0 | USB PLL Lock Raw Interrupt Status Value Description 1 The USB PLL timer has reached T_{READY} indicating that sufficient time has passed for the USB PLL to lock. 0 The USB PLL timer has not reached T_{READY} . This bit is cleared by writing a 1 to the USBPLLRLMIS bit in the MISC register. |
| 6 | PLLLRIS | RO | 0 | PLL Lock Raw Interrupt Status Value Description 1 The PLL timer has reached T_{READY} indicating that sufficient time has passed for the PLL to lock. 0 The PLL timer has not reached T_{READY} . This bit is cleared by writing a 1 to the PLLRLMIS bit in the MISC register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 5:2 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | BORRIS | RO | 0 | <p>Brown-Out Reset Raw Interrupt Status</p> <p>Value Description</p> <p>1 A brown-out condition is currently active.</p> <p>0 A brown-out condition is not currently active.</p> <p>Note the BORIOR bit in the PBORCTL register must be cleared to cause an interrupt due to a Brown Out Event.</p> <p>This bit is cleared by writing a 1 to the BORMIS bit in the MISC register.</p> |
| 0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 4: Interrupt Mask Control (IMC), offset 0x054

This register contains the mask bits for system control raw interrupts. A raw interrupt, indicated by a bit being set in the **Raw Interrupt Status (RIS)** register, is sent to the interrupt controller if the corresponding bit in this register is set.

Interrupt Mask Control (IMC)

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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|-----------|----------|--------|----------|----|----|----|-------|----------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | MOSCPUPIM | USBPLLIM | PLLLIM | reserved | | | | BORIM | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | RO | RO | RO | RO | R/W | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------|------|-----------|--|
| 31:9 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | MOSCPUPIM | R/W | 0 | MOSC Power Up Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the RIS register is set. 0 The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller. |
| 7 | USBPLLIM | R/W | 0 | USB PLL Lock Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the USBPLLRRIS bit in the RIS register is set. 0 The USBPLLRRIS interrupt is suppressed and not sent to the interrupt controller. |
| 6 | PLLLIM | R/W | 0 | PLL Lock Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the PLLRRIS bit in the RIS register is set. 0 The PLLRRIS interrupt is suppressed and not sent to the interrupt controller. |
| 5:2 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 1 | BORIM | R/W | 0 | Brown-Out Reset Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the BORRIS bit in the RIS register is set. 0 The BORRIS interrupt is suppressed and not sent to the interrupt controller. |
| 0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 5: Masked Interrupt Status and Clear (MISC), offset 0x058

On a read, this register gives the current masked status value of the corresponding interrupt in the **Raw Interrupt Status (RIS)** register. All of the bits are R/W1C, thus writing a 1 to a bit clears the corresponding raw interrupt bit in the **RIS** register (see page 113).

Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000

Offset 0x058

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 | | | | | | | MOSCPUPMIS | USBPLLLMIS | PLLLMIS | reserved | | | | BORMIS | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | RO | RO | RO | RO | R/W1C | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------------|-------|-----------|---|
| 31:9 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | MOSCPUPMIS | R/W1C | 0 | MOSC Power Up Masked Interrupt Status Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the MOSC PLL to lock. Writing a 1 to this bit clears it and also the <code>MOSCPUPRIS</code> bit in the RIS register. 0 When read, a 0 indicates that sufficient time has not passed for the MOSC PLL to lock. A write of 0 has no effect on the state of this bit. |
| 7 | USBPLLLMIS | R/W1C | 0 | USB PLL Lock Masked Interrupt Status Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the USB PLL to lock. Writing a 1 to this bit clears it and also the <code>USBPLLLRIS</code> bit in the RIS register. 0 When read, a 0 indicates that sufficient time has not passed for the USB PLL to lock. A write of 0 has no effect on the state of this bit. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|-------|---|
| 6 | PLLLMIS | R/W1C | 0 | <p>PLL Lock Masked Interrupt Status</p> <p>Value Description</p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the PLL to lock.</p> <p>Writing a 1 to this bit clears it and also the PLLLRIS bit in the RIS register.</p> <p>0 When read, a 0 indicates that sufficient time has not passed for the PLL to lock.</p> <p>A write of 0 has no effect on the state of this bit.</p> |
| 5:2 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | BORMIS | R/W1C | 0 | <p>BOR Masked Interrupt Status</p> <p>Value Description</p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because of a brown-out condition.</p> <p>Writing a 1 to this bit clears it and also the BORRIS bit in the RIS register.</p> <p>0 When read, a 0 indicates that a brown-out condition has not occurred.</p> <p>A write of 0 has no effect on the state of this bit.</p> |
| 0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 6: Reset Cause (RESC), offset 0x05C

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

Reset Cause (RESC)

Base 0x400F.E000

Offset 0x05C

Type R/W, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|------|-----|------|-----|-----|----------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | MOSCFAIL |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | 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 | | | | | | | | | | WDT1 | SW | WDT0 | BOR | POR | EXT |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:17 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 16 | MOSCFAIL | R/W | - | MOSC Failure Reset |
| | | | | Value Description |
| | | | | 1 When read, this bit indicates that the MOSC circuit was enabled for clock validation and failed, generating a reset event. |
| | | | | 0 When read, this bit indicates that a MOSC failure has not generated a reset since the previous power-on reset. |
| | | | | Writing a 0 to this bit clears it. |
| 15:6 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | WDT1 | R/W | - | Watchdog Timer 1 Reset |
| | | | | Value Description |
| | | | | 1 When read, this bit indicates that Watchdog Timer 1 timed out and generated a reset. |
| | | | | 0 When read, this bit indicates that Watchdog Timer 1 has not generated a reset since the previous power-on reset. |
| | | | | Writing a 0 to this bit clears it. |

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

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

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

Run-Mode Clock Configuration (RCC)

Base 0x400F.E000

Offset 0x060

Type R/W, reset 0x0780.3AD1

| | | | | | | | | | | | | | | | | |
|-------|----------|----|-------|----------|--------|--------|-----|-----|-----|-----------|----------|----------|----|---------|---------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | ACG | SYSDIV | | | | USESYSDIV | reserved | | | | | |
| Type | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | PWRDN | reserved | BYPASS | XTAL | | | | OSCSRC | | reserved | | IOSCDIS | MOSCDIS | |
| Type | RO | RO | R/W | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |

| 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 microcontroller enters a Sleep or Deep-Sleep mode (respectively).

Value Description

- | | |
|---|--|
| 1 | The SCGCn or DCGCn registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The SCGCn and DCGCn registers allows unused peripherals to consume less power when the microcontroller is in a sleep mode. |
| 0 | The Run-Mode Clock Gating Control (RCGCn) registers are used when the microcontroller enters a sleep mode. |

The **RCGCn** registers are always used to control the clocks in Run mode.

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|--|----------------------|-------|--|-------|--------------------|----------------------|--|----------|-------------------------------------|-----|----|----------|-----|----|--------|-----|----|--------|-----|----|--------|-----|----|-----------|-----|----|-----------|-----|----|--------|-----|----|-----------|-----|-----|--------|-----|-----|-----------|-----|-----|-----------|-----|-----|-----------|-----|-----|-----------|-----|-----|-----------|-----|-----|--------------------|
| 26:23 | SYSDIV | R/W | 0xF | <p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from the PLL output.</p> <p>Although the PLL VCO frequency is 400 MHz, it is predivided by 2 before the divisor is applied.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Divisor (BYPASS=1)</th> <th>Frequency (BYPASS=0)</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>reserved</td><td>reserved</td></tr> <tr><td>0x1</td><td>/2</td><td>reserved</td></tr> <tr><td>0x2</td><td>/3</td><td>80 MHz</td></tr> <tr><td>0x3</td><td>/4</td><td>50 MHz</td></tr> <tr><td>0x4</td><td>/5</td><td>40 MHz</td></tr> <tr><td>0x5</td><td>/6</td><td>33.33 MHz</td></tr> <tr><td>0x6</td><td>/7</td><td>28.57 MHz</td></tr> <tr><td>0x7</td><td>/8</td><td>25 MHz</td></tr> <tr><td>0x8</td><td>/9</td><td>22.22 MHz</td></tr> <tr><td>0x9</td><td>/10</td><td>20 MHz</td></tr> <tr><td>0xA</td><td>/11</td><td>18.18 MHz</td></tr> <tr><td>0xB</td><td>/12</td><td>16.67 MHz</td></tr> <tr><td>0xC</td><td>/13</td><td>15.38 MHz</td></tr> <tr><td>0xD</td><td>/14</td><td>14.29 MHz</td></tr> <tr><td>0xE</td><td>/15</td><td>13.33 MHz</td></tr> <tr><td>0xF</td><td>/16</td><td>12.5 MHz (default)</td></tr> </tbody> </table> <p>If the SYSDIV value is less than MINSYSDIV (see page 144), and the PLL is being used, then the MINSYSDIV value is used as the divisor.</p> <p>If the PLL is not being used, the SYSDIV value can be less than MINSYSDIV.</p> | Value | Divisor (BYPASS=1) | Frequency (BYPASS=0) | 0x0 | reserved | reserved | 0x1 | /2 | reserved | 0x2 | /3 | 80 MHz | 0x3 | /4 | 50 MHz | 0x4 | /5 | 40 MHz | 0x5 | /6 | 33.33 MHz | 0x6 | /7 | 28.57 MHz | 0x7 | /8 | 25 MHz | 0x8 | /9 | 22.22 MHz | 0x9 | /10 | 20 MHz | 0xA | /11 | 18.18 MHz | 0xB | /12 | 16.67 MHz | 0xC | /13 | 15.38 MHz | 0xD | /14 | 14.29 MHz | 0xE | /15 | 13.33 MHz | 0xF | /16 | 12.5 MHz (default) |
| Value | Divisor (BYPASS=1) | Frequency (BYPASS=0) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | reserved | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | /2 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | /3 | 80 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | /4 | 50 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | /5 | 40 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | /6 | 33.33 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | /7 | 28.57 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | /8 | 25 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | /9 | 22.22 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9 | /10 | 20 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA | /11 | 18.18 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB | /12 | 16.67 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xC | /13 | 15.38 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD | /14 | 14.29 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xE | /15 | 13.33 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF | /16 | 12.5 MHz (default) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | USESYSCLK | R/W | 0 | <p>Enable System Clock Divider</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.</td> </tr> <tr> <td>0</td> <td>The system clock is used undivided.</td> </tr> </tbody> </table> | Value | Description | 1 | The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source. | 0 | The system clock is used undivided. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | The system clock is used undivided. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21:14 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | PWRDN | R/W | 1 | <p>PLL Power Down</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.</td> </tr> <tr> <td>0</td> <td>The PLL is operating normally.</td> </tr> </tbody> </table> | Value | Description | 1 | The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit. | 0 | The PLL is operating normally. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | The PLL is operating normally. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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 Value Description 1 The system clock is derived from the OSC source. 0 The system clock is the PLL output clock divided by the system divider. 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---------------|
| 10:6 | XTAL | R/W | 0x0B | 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 26-11 on page 1043 for more information.

Frequencies that may be used with the USB interface are indicated in the table. To function within the clocking requirements of the USB specification, a crystal of 4, 5, 6, 8, 10, 12, or 16 MHz must be used.

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

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

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

This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the Run-Mode Clock Configuration (RCC) register (see page 121).

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 -

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 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 | | F | | | | | | | | | | R | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:14 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 13:5 | F | RO | - | PLL F Value 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 9: GPIO Host-Bus Control (GPIOHBCTL), offset 0x06C

This register controls which internal bus is used to access each GPIO port. When a bit is clear, the corresponding GPIO port is accessed across the legacy Advanced Peripheral Bus (APB) bus and through the APB memory aperture. When a bit is set, the corresponding port is accessed across the Advanced Host Bus (AHB) bus and through the AHB memory aperture. Each GPIO port can be individually configured to use AHB or APB, but may be accessed only through one aperture. The AHB bus provides better back-to-back access performance than the APB bus. The address aperture in the memory map changes for the ports that are enabled for AHB access (see Table 10-6 on page 328).

GPIO Host-Bus Control (GPIOHBCTL)

Base 0x400F.E000
Offset 0x06C
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 | | | | | | | PORTJ | PORTH | PORTG | PORTF | PORTE | PORTD | PORTC | PORTB | PORTA |
| Type | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|--|
| 31:9 | 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. |
| 8 | PORTJ | R/W | 0 | Port J Advanced Host Bus This bit defines the memory aperture for Port J. Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 7 | PORTH | R/W | 0 | Port H Advanced Host Bus This bit defines the memory aperture for Port H. Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |
| 6 | PORTG | R/W | 0 | Port G Advanced Host Bus This bit defines the memory aperture for Port G. Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|---|
| 5 | PORTF | R/W | 0 | <p>Port F Advanced Host Bus</p> <p>This bit defines the memory aperture for Port F.</p> <p>Value Description</p> <p>1 Advanced Host Bus (AHB)</p> <p>0 Advanced Peripheral Bus (APB). This bus is the legacy bus.</p> |
| 4 | PORTE | R/W | 0 | <p>Port E Advanced Host Bus</p> <p>This bit defines the memory aperture for Port E.</p> <p>Value Description</p> <p>1 Advanced Host Bus (AHB)</p> <p>0 Advanced Peripheral Bus (APB). This bus is the legacy bus.</p> |
| 3 | PORTD | R/W | 0 | <p>Port D Advanced Host Bus</p> <p>This bit defines the memory aperture for Port D.</p> <p>Value Description</p> <p>1 Advanced Host Bus (AHB)</p> <p>0 Advanced Peripheral Bus (APB). This bus is the legacy bus.</p> |
| 2 | PORTC | R/W | 0 | <p>Port C Advanced Host Bus</p> <p>This bit defines the memory aperture for Port C.</p> <p>Value Description</p> <p>1 Advanced Host Bus (AHB)</p> <p>0 Advanced Peripheral Bus (APB). This bus is the legacy bus.</p> |
| 1 | PORTB | R/W | 0 | <p>Port B Advanced Host Bus</p> <p>This bit defines the memory aperture for Port B.</p> <p>Value Description</p> <p>1 Advanced Host Bus (AHB)</p> <p>0 Advanced Peripheral Bus (APB). This bus is the legacy bus.</p> |
| 0 | PORTA | R/W | 0 | <p>Port A Advanced Host Bus</p> <p>This bit defines the memory aperture for Port A.</p> <p>Value Description</p> <p>1 Advanced Host Bus (AHB)</p> <p>0 Advanced Peripheral Bus (APB). This bus is the legacy bus.</p> |

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

This register overrides the **RCC** equivalent register fields 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. The fields within the **RCC2** register occupy the same bit positions as they do within the **RCC** register as LSB-justified.

The **SYSDIV2** field is 2 bits wider than the **SYSDIV** field in the **RCC** register so that additional larger divisors are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. An additional bit, **FRACT**, has been added as an available LSB for **SYSDIV2** to provide additional frequency choices. **FRACT** can be modified when **USEFRACT** is set. The following table provides some examples of frequency choices using the **SYSDIV2**, **USEFRACT** and **FRACT** bits. The PLL VCO frequency is 400 MHz.

Table 6-5. Examples of Possible System Clock Frequencies

| System Clock | SYSDIV2 | USEFRACT | FRACT |
|--------------|---------|----------|------------|
| 20 MHz | 0x09 | 0 | don't care |
| 20 MHz | 0x09 | 1 | 1 |
| 25 MHz | 0x07 | 0 | don't care |
| 40 MHz | 0x04 | 0 | don't care |
| 44.4 MHz | 0x04 | 1 | 0 |
| 50 MHz | 0x03 | 0 | don't care |
| 80 MHz | 0x02 | 1 | 0 |

Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000

Offset 0x070

Type R/W, reset 0x0780.6810

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|----------|----------|----------|----------|---------|----------|-----|-----|-----|---------|----------|-----|----------|----|----|----|
| | USERCC2 | USEFRACT | reserved | SYSDIV2 | | | | | | FRACT | reserved | | | | | |
| Type | R/W | R/W | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | USBPWRDN | PWRDN2 | reserved | BYPASS2 | reserved | | | | OSCSRC2 | | | reserved | | | |
| Type | RO | R/W | R/W | RO | R/W | RO | RO | RO | RO | R/W | R/W | R/W | RO | RO | RO | RO |
| Reset | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|-----------------|
| 31 | USERCC2 | R/W | 0 | Use RCC2 |

Value Description

| Value | Description |
|-------|---|
| 1 | The RCC2 register fields override the RCC register fields. |
| 0 | The RCC register fields are used, and the fields in RCC2 are ignored. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 30 | USEFRACT | R/W | 0 | <p>Use FRACT</p> <p>The FRACT bit adds an additional bit as the LSB to the SYSDIV2 field allowing additional frequency choices.</p> <p>Value Description</p> <p>1 The FRACT bit can be set or cleared by the software.</p> <p>0 The FRACT bit is forced to be set.</p> |
| 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 | <p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from the PLL output.</p> <p>Although the PLL VCO frequency is 400 MHz, it is predivided by 2 before the divisor is applied.</p> <p>This field is wider than the RCC register SYSDIV field in order to provide additional divisor values. These additional values permit the system clock to be run at much lower frequencies during Deep Sleep mode. For example, where the RCC register SYSDIV encoding of 1111 provides /16, the RCC2 register SYSDIV2 encoding of 111111 provides /64.</p> |
| 22 | FRACT | R/W | 0 | <p>Fractional Divider</p> <p>The FRACT bit adds an additional bit as the LSB to the SYSDIV2 field allowing additional frequency choices.</p> <p>This bit can only be set or cleared when USEFRACT is set.</p> |
| 21:15 | reserved | RO | 0x0 | Software should not rely on the value of 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 | USBPWRDN | R/W | 1 | <p>Power-Down USB PLL</p> <p>Value Description</p> <p>1 The USB PLL is powered down.</p> <p>0 The USB PLL operates normally.</p> |
| 13 | PWRDN2 | R/W | 1 | <p>Power-Down PLL</p> <p>Value Description</p> <p>1 The PLL is powered down.</p> <p>0 The PLL operates normally.</p> |
| 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 | <p>PLL Bypass</p> <p>Value Description</p> <p>1 The system clock is derived from the OSC source.</p> <p>0 The system clock is the PLL output clock divided by the system divider.</p> <p>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.</p> |
| 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 | <p>Oscillator Source</p> <p>Selects the input source for the OSC. The values are:</p> <p>Value Description</p> <p>0x0 MOSC Main oscillator</p> <p>0x1 PIOSC Precision internal oscillator</p> <p>0x2 PIOSC/4 Precision internal oscillator / 4</p> <p>0x3 30 kHz 30-kHz internal oscillator</p> <p>0x4-0x5 Reserved</p> <p>0x6 4.19 MHz 4.194304-MHz external oscillator</p> <p>0x7 32 kHz 32.768-kHz external oscillator</p> |
| 3:0 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 11: Main Oscillator Control (MOSCCTL), offset 0x07C

This register provides the ability to enable the MOSC clock verification circuit. When enabled, this circuit monitors the frequency of the MOSC to verify that the oscillator is operating within specified limits. If the clock goes invalid after being enabled, the microcontroller issues a power-on reset and reboots to the NMI handler.

Main Oscillator Control (MOSCCTL)

Base 0x400F.E000
 Offset 0x07C
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------|
| | 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 | | | | | | | | | | | | | | | CVAL |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | CVAL | R/W | 0 | Clock Validation for MOSC |
| | | | | Value Description |
| | | | | 1 The MOSC monitor circuit is enabled. |
| | | | | 0 The MOSC monitor circuit is disabled. |

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

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

Deep Sleep Clock Configuration (DSLPCCLKCFG)

Base 0x400F.E000

Offset 0x144

Type R/W, reset 0x0780.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|------------|-----|-----|----------|-----|-----|----------|-----|----------|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | DSDIVORIDE | | | | | | reserved | | | | | | |
| Type | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | DSOSCSRC | | | | | reserved | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|-------------|------|-------|---|-------|-------------|-----|----------|-----|----|-----|----|-----|----|-----|----|-----|----|-----|----|-----|----|-----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 31:29 | 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 | <p>Divider Field Override</p> <p>If Deep-Sleep mode is enabled when the PLL is running, the PLL is disabled. This 6-bit field contains a system divider field that overrides the SYSDIV field in the RCC register or the SYSDIV2 field in the RCC2 register during Deep Sleep. This divider is applied to the source selected by the DSOSCSRC field.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>reserved</td></tr> <tr><td>0x1</td><td>/2</td></tr> <tr><td>0x2</td><td>/3</td></tr> <tr><td>0x3</td><td>/4</td></tr> <tr><td>0x4</td><td>/5</td></tr> <tr><td>0x5</td><td>/6</td></tr> <tr><td>0x6</td><td>/7</td></tr> <tr><td>0x7</td><td>/8</td></tr> <tr><td>0x8</td><td>/9</td></tr> <tr><td>0x9</td><td>/10</td></tr> <tr><td>0xA</td><td>/11</td></tr> <tr><td>0xB</td><td>/12</td></tr> <tr><td>0xC</td><td>/13</td></tr> <tr><td>0xD</td><td>/14</td></tr> <tr><td>0xE</td><td>/15</td></tr> <tr><td>0xF</td><td>/16</td></tr> </tbody> </table> | Value | Description | 0x0 | reserved | 0x1 | /2 | 0x2 | /3 | 0x3 | /4 | 0x4 | /5 | 0x5 | /6 | 0x6 | /7 | 0x7 | /8 | 0x8 | /9 | 0x9 | /10 | 0xA | /11 | 0xB | /12 | 0xC | /13 | 0xD | /14 | 0xE | /15 | 0xF | /16 |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | /2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | /3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | /4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | /5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | /6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | /7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | /8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | /9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9 | /10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA | /11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB | /12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xC | /13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD | /14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xE | /15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF | /16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22:7 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | |
|-----------|---|------|-------|--|-------|-------------|-----|---|-----|--|-----|----------|-----|--|---------|----------|-----|--|-----|---|
| 6:4 | DSOSCSRC | R/W | 0x0 | <p>Clock Source</p> <p>Specifies the clock source during Deep-Sleep mode.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td> <p>MOSC</p> <p>Use the main oscillator as the source.</p> <p>Note: If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode.</p> <p>If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of MOSC in Deep-Sleep mode.</p> </td> </tr> <tr> <td>0x1</td> <td> <p>PIOSC</p> <p>Use the precision internal 16-MHz oscillator as the source.</p> <p>Note: If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of PIOSC in Deep-Sleep mode.</p> </td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td> <p>30 kHz</p> <p>Use the 30-kHz internal oscillator as the source.</p> </td> </tr> <tr> <td>0x4-0x5</td> <td>Reserved</td> </tr> <tr> <td>0x6</td> <td> <p>4.194304 MHz</p> <p>Use the Hibernation module 4.194304-MHz external crystal clock as the source.</p> </td> </tr> <tr> <td>0x7</td> <td> <p>32 kHz</p> <p>Use the Hibernation module 32.768-kHz external oscillator as the source.</p> </td> </tr> </tbody> </table> | Value | Description | 0x0 | <p>MOSC</p> <p>Use the main oscillator as the source.</p> <p>Note: If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode.</p> <p>If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of MOSC in Deep-Sleep mode.</p> | 0x1 | <p>PIOSC</p> <p>Use the precision internal 16-MHz oscillator as the source.</p> <p>Note: If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of PIOSC in Deep-Sleep mode.</p> | 0x2 | Reserved | 0x3 | <p>30 kHz</p> <p>Use the 30-kHz internal oscillator as the source.</p> | 0x4-0x5 | Reserved | 0x6 | <p>4.194304 MHz</p> <p>Use the Hibernation module 4.194304-MHz external crystal clock as the source.</p> | 0x7 | <p>32 kHz</p> <p>Use the Hibernation module 32.768-kHz external oscillator as the source.</p> |
| Value | Description | | | | | | | | | | | | | | | | | | | |
| 0x0 | <p>MOSC</p> <p>Use the main oscillator as the source.</p> <p>Note: If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode.</p> <p>If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of MOSC in Deep-Sleep mode.</p> | | | | | | | | | | | | | | | | | | | |
| 0x1 | <p>PIOSC</p> <p>Use the precision internal 16-MHz oscillator as the source.</p> <p>Note: If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of PIOSC in Deep-Sleep mode.</p> | | | | | | | | | | | | | | | | | | | |
| 0x2 | Reserved | | | | | | | | | | | | | | | | | | | |
| 0x3 | <p>30 kHz</p> <p>Use the 30-kHz internal oscillator as the source.</p> | | | | | | | | | | | | | | | | | | | |
| 0x4-0x5 | Reserved | | | | | | | | | | | | | | | | | | | |
| 0x6 | <p>4.194304 MHz</p> <p>Use the Hibernation module 4.194304-MHz external crystal clock as the source.</p> | | | | | | | | | | | | | | | | | | | |
| 0x7 | <p>32 kHz</p> <p>Use the Hibernation module 32.768-kHz external oscillator as the source.</p> | | | | | | | | | | | | | | | | | | | |
| 3:0 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | |

Register 13: Deep Sleep Flash Configuration (DSFLASHCFG), offset 0x14C

This register allows the user to force the shutdown of the Flash subsystem during all Deep-Sleep periods. For deep-sleep periods that do not require a MOSC startup time or a PLL lock time, the microcontroller has a lockout period of 30-120 μ s for the Flash to start up after the event to exit deep sleep has occurred.

Deep Sleep Flash Configuration (DSFLASHCFG)

Base 0x400F.E000

Offset 0x14C

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 | | | | | | | | | | | | | | | SHDWN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | SHDWN | R/W | 0 | Flash Shutdown |
| | | | | Value Description |
| | | | | 1 The Flash subsystem is shutdown during all deep-sleep operations. |
| | | | | 0 The Flash subsystem is powered up during deep-sleep operations |

Register 14: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150

This register provides the ability to update or recalibrate the precision internal oscillator. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

Precision Internal Oscillator Calibration (PIOSCCAL)

Base 0x400F.E000
 Offset 0x150
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----|----|-----|--------|----------|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | UTEN | reserved | | | | | | | | | | | | | | |
| Type | R/W | 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 | | | | | | CAL | UPDATE | reserved | UT | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | R/W | R/W | 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 | UTEN | R/W | 0 | Use User Trim Value |
| | | | | Value Description |
| | | | | 1 The trim value in bits[6:0] of this register are used for any update trim operation. |
| | | | | 0 The factory calibration value is used for an update trim operation. |
| 30:10 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9 | CAL | R/W | 0 | Start Calibration |
| | | | | Value Description |
| | | | | 1 Starts a new calibration of the PIOSC. Results are in the PIOSCSTAT register. The resulting trim value from the operation is active in the PIOSC after the calibration completes. The result overrides any previous update trim operation whether the calibration passes or fails. |
| | | | | 0 No action. |
| | | | | This bit is auto-cleared when the calibration finishes. |
| 8 | UPDATE | R/W | 0 | Update Trim |
| | | | | Value Description |
| | | | | 1 Updates the PIOSC trim value with the UT bit or the DT bit in the PIOSCSTAT register. Used with UTEN . |
| | | | | 0 No action. |
| | | | | This bit is auto-cleared after the update. |
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 6:0 | UT | R/W | 0x0 | User Trim Value User trim value that can be loaded into the PIOSC. Refer to "Main PLL Frequency Configuration" on page 104 for more information on calibrating the PIOSC. |

Register 15: Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154

This register provides the user information on the PIOSC calibration. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

Precision Internal Oscillator Statistics (PIOSCSTAT)

Base 0x400F.E000

Offset 0x154

Type RO, reset 0x0000.0040

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|--------|----|----------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | DT | | | | | | |
| 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 | - | - | - | - | - | - | - |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | RESULT | | reserved | CT | | | | | | |
| 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 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:23 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 22:16 | DT | RO | - | Default Trim Value This field contains the default trim value. This value is loaded into the PIOSC after every full power-up. |
| 15:10 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9:8 | RESULT | RO | 0 | Calibration Result Value Description 0x0 Calibration has not been attempted. 0x1 The last calibration operation completed to meet 1% accuracy. 0x2 The last calibration operation failed to meet 1% accuracy. 0x3 Reserved |
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | CT | RO | 0x40 | Calibration Trim Value This field contains the trim value from the last calibration operation. After factory calibration CT and DT are the same. |

Register 16: I²S MCLK Configuration (I2SMCLKCFG), offset 0x170

This register configures the receive and transmit fractional clock dividers for the for the I²S master transmit and receive clocks (I2S0TXMCLK and I2S0RXMCLK) . Varying the integer and fractional inputs for the clocks allows greater accuracy in hitting the target I²S clock frequencies. Refer to “Clock Control” on page 718 for combinations of the TXI and TXF bits and the RXI and RXF bits that provide MCLK frequencies within acceptable error limits.

I2S MCLK Configuration (I2SMCLKCFG)

Base 0x400F.E000
Offset 0x170
Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | RXEN | reserved | | | | | | | | | | | | | | |
| Type | R/W | RO | 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 |
| | TXEN | reserved | | | | | | | | | | | | | | |
| Type | R/W | RO | 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 | RXEN | R/W | 0 | <p>RX Clock Enable</p> <p>Value Description</p> <p>1 The I²S receive clock generator is enabled.</p> <p>0 The I²S receive clock generator is disabled.</p> <p>If the RXSLV bit in the I²S Module Configuration (I2SCFG) register is set, then the I2S0RXMCLK must be externally generated.</p> |
| 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. |
| 29:20 | RXI | R/W | 0x0 | <p>RX Clock Integer Input</p> <p>This field contains the integer input for the receive clock generator.</p> |
| 19:16 | RXF | R/W | 0x0 | <p>RX Clock Fractional Input</p> <p>This field contains the fractional input for the receive clock generator.</p> |
| 15 | TXEN | R/W | 0 | <p>TX Clock Enable</p> <p>Value Description</p> <p>1 The I²S transmit clock generator is enabled.</p> <p>0 The I²S transmit clock generator is disabled.</p> <p>If the TXSLV bit in the I²S Module Configuration (I2SCFG) register is set, then the I2S0TXMCLK must be externally generated.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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. |
| 13:4 | TXI | R/W | 0x00 | TX Clock Integer Input This field contains the integer input for the transmit clock generator. |
| 3:0 | TXF | R/W | 0x0 | TX Clock Fractional Input This field contains the fractional input for the transmit clock generator. |

Register 17: Device Identification 1 (DID1), offset 0x004

This register identifies the device family, part number, temperature range, and package type.

Device Identification 1 (DID1)

Base 0x400F.E000

Offset 0x004

Type RO, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----------|-----|----|----|------|--------|----|-----|----|------|------|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | VER | | | | FAM | | | | PARTNO | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PINCOUNT | | | reserved | | | | TEMP | | | PKG | | ROHS | QUAL | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | - | - | - | - | - | 1 | - | - |

| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|---|------|-------|---|-------|-------------|------|---|
| 31:28 | VER | RO | 0x1 | <p>DID1 Version</p> <p>This field defines the DID1 register format version. The version number is numeric. The value of the <code>VER</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>Second version of the DID1 register format.</td> </tr> </tbody> </table> | Value | Description | 0x1 | Second version of the DID1 register format. |
| Value | Description | | | | | | | |
| 0x1 | Second version of the DID1 register format. | | | | | | | |
| 27:24 | FAM | RO | 0x0 | <p>Family</p> <p>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):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.</td> </tr> </tbody> </table> | Value | Description | 0x0 | Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S. |
| Value | Description | | | | | | | |
| 0x0 | Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S. | | | | | | | |
| 23:16 | PARTNO | RO | 0x66 | <p>Part Number</p> <p>This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x66</td> <td>LM3S9B90</td> </tr> </tbody> </table> | Value | Description | 0x66 | LM3S9B90 |
| Value | Description | | | | | | | |
| 0x66 | LM3S9B90 | | | | | | | |
| 15:13 | PINCOUNT | RO | 0x2 | <p>Package Pin Count</p> <p>This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x2</td> <td>100-pin package</td> </tr> </tbody> </table> | Value | Description | 0x2 | 100-pin package |
| Value | Description | | | | | | | |
| 0x2 | 100-pin package | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 12:8 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:5 | TEMP | RO | - | Temperature Range 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 18: 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 0x017F.007F

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | SRAMSZ | | | | | | | | | | | | | | | |
| 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 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FLASHSZ | | | | | | | | | | | | | | | |
| 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 |
|-----------|---------|------|--------|--|
| 31:16 | SRAMSZ | RO | 0x017F | SRAM Size Indicates the size of the on-chip SRAM memory. Value Description 0x017F 96 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 19: Device Capabilities 1 (DC1), offset 0x010

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

Device Capabilities 1 (DC1)

Base 0x400F.E000

Offset 0x010

Type RO, reset -

| | | | | | | | | | | | | | | | | | | |
|-------|-----------|----|----|------|------------|----|------------|------|----------|-----|---------|-----|------|-----|-----|------|------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| | reserved | | | WDT1 | reserved | | CAN1 | CAN0 | reserved | | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| | MINSYSDIV | | | | MAXADC1SPD | | MAXADC0SPD | | MPU | HIB | TEMPSNS | PLL | WDT0 | SWO | SWD | JTAG | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | | |
| Reset | - | - | - | - | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | WDT1 | RO | 1 | Watchdog Timer1 Present When set, indicates that watchdog timer 1 is present. |
| 27:26 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 25 | CAN1 | RO | 1 | CAN Module 1 Present When set, indicates that CAN unit 1 is present. |
| 24 | CAN0 | RO | 1 | CAN Module 0 Present When set, indicates that CAN unit 0 is present. |
| 23:18 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 17 | ADC1 | RO | 1 | ADC Module 1 Present When set, indicates that ADC module 1 is present. |
| 16 | ADC0 | RO | 1 | ADC Module 0 Present When set, indicates that ADC module 0 is present |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------------|------|-------|---|
| 15:12 | MINSYSDIV | RO | - | <p>System Clock Divider</p> <p>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.</p> <p>Value Description</p> <p>0x1 Divide VCO (400MHZ) by 5 minimum</p> <p>0x2 Divide VCO (400MHZ) by $2^2 + 2 = 6$ minimum</p> <p>0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4.</p> <p>0x7 Specifies a 25-MHz clock with a PLL divider of 8.</p> <p>0x9 Specifies a 20-MHz clock with a PLL divider of 10.</p> |
| 11:10 | MAXADC1SPD | RO | 0x3 | <p>Max ADC1 Speed</p> <p>This field indicates the maximum rate at which the ADC samples data.</p> <p>Value Description</p> <p>0x3 1M samples/second</p> |
| 9:8 | MAXADC0SPD | RO | 0x3 | <p>Max ADC0 Speed</p> <p>This field indicates the maximum rate at which the ADC samples data.</p> <p>Value Description</p> <p>0x3 1M samples/second</p> |
| 7 | MPU | RO | 1 | <p>MPU Present</p> <p>When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the ARM Cortex-M3 Technical Reference Manual for details on the MPU.</p> |
| 6 | HIB | RO | 1 | <p>Hibernation Module Present</p> <p>When set, indicates that the Hibernation module is present.</p> |
| 5 | TEMPSNS | RO | 1 | <p>Temp Sensor Present</p> <p>When set, indicates that the on-chip temperature sensor is present.</p> |
| 4 | PLL | RO | 1 | <p>PLL Present</p> <p>When set, indicates that the on-chip Phase Locked Loop (PLL) is present.</p> |
| 3 | WDT0 | RO | 1 | <p>Watchdog Timer 0 Present</p> <p>When set, indicates that watchdog timer 0 is present.</p> |
| 2 | SWO | RO | 1 | <p>SWO Trace Port Present</p> <p>When set, indicates that the Serial Wire Output (SWO) trace port is present.</p> |
| 1 | SWD | RO | 1 | <p>SWD Present</p> <p>When set, indicates that the Serial Wire Debugger (SWD) is present.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 0 | JTAG | RO | 1 | JTAG Present When set, indicates that the JTAG debugger interface is present. |

Register 20: Device Capabilities 2 (DC2), offset 0x014

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

Device Capabilities 2 (DC2)

Base 0x400F.E000

Offset 0x014

Type RO, reset 0x570F.5037

| | | | | | | | | | | | | | | | | |
|-------|----------|------|----------|------|----------|-------|-------|-------|----------|----|------|--------|----------|--------|--------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | EPI0 | reserved | I2S0 | 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 | 1 | 0 | 1 | 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 | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 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 | EPI0 | RO | 1 | EPI Module 0 Present When set, indicates that EPI 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 | I2S0 | RO | 1 | I2S Module 0 Present When set, indicates that I2S module 0 is present. |
| 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 Module 3 Present When set, indicates that General-Purpose Timer module 3 is present. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 18 | TIMER2 | RO | 1 | Timer Module 2 Present When set, indicates that General-Purpose Timer module 2 is present. |
| 17 | TIMER1 | RO | 1 | Timer Module 1 Present When set, indicates that General-Purpose Timer module 1 is present. |
| 16 | TIMER0 | RO | 1 | Timer Module 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. |
| 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. |
| 11:6 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | SSI1 | RO | 1 | SSI Module 1 Present When set, indicates that SSI module 1 is present. |
| 4 | SSI0 | RO | 1 | SSI Module 0 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 | UART Module 2 Present When set, indicates that UART module 2 is present. |
| 1 | UART1 | RO | 1 | UART Module 1 Present When set, indicates that UART module 1 is present. |
| 0 | UART0 | RO | 1 | UART Module 0 Present When set, indicates that UART module 0 is present. |

Register 21: Device Capabilities 3 (DC3), offset 0x018

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

Device Capabilities 3 (DC3)

Base 0x400F.E000

Offset 0x018

Type RO, reset 0xBFFF.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 | ADC0AIN7 | ADC0AIN6 | ADC0AIN5 | ADC0AIN4 | ADC0AIN3 | ADC0AIN2 | ADC0AIN1 | ADC0AIN0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | 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 | ADC0AIN7 | RO | 1 | ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present. |
| 22 | ADC0AIN6 | RO | 1 | ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 21 | ADC0AIN5 | RO | 1 | ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present. |
| 20 | ADC0AIN4 | RO | 1 | ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present. |
| 19 | ADC0AIN3 | RO | 1 | ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present. |
| 18 | ADC0AIN2 | RO | 1 | ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present. |
| 17 | ADC0AIN1 | RO | 1 | ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present. |
| 16 | ADC0AIN0 | RO | 1 | ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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 22: Device Capabilities 4 (DC4), offset 0x01C

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

Device Capabilities 4 (DC4)

Base 0x400F.E000
 Offset 0x01C
 Type RO, reset 0x5004.F1FF

| | | | | | | | | | | | | | | | | |
|-------|----------|-------|----------|-------|----------|----|----|-------|-------|-------|-------|-------|----------|------|------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | EPHY0 | reserved | EMAC0 | reserved | | | | | | | PICAL | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | CCP7 | CCP6 | UDMA | ROM | reserved | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | PIOD | PIOC | PIOB | PIOA |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 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 PHY Layer 0 Present When set, indicates that Ethernet PHY layer 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 MAC Layer 0 Present When set, indicates that Ethernet MAC layer 0 is present. |
| 27:19 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 18 | PICAL | RO | 1 | PIOSC Calibrate When set, indicates that the PIOSC can be calibrated by software. |
| 17:16 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15 | CCP7 | RO | 1 | CCP7 Pin Present When set, indicates that Capture/Compare/PWM pin 7 is present. |
| 14 | CCP6 | RO | 1 | CCP6 Pin Present When set, indicates that Capture/Compare/PWM pin 6 is present. |
| 13 | UDMA | RO | 1 | Micro-DMA Module Present When set, indicates that the micro-DMA module present. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 12 | ROM | RO | 1 | Internal Code ROM Present When set, indicates that internal code ROM is present. |
| 11:9 | 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 | GPIOJ | RO | 1 | GPIO Port J Present When set, indicates that GPIO Port J is present. |
| 7 | GPIOH | RO | 1 | GPIO Port H Present When set, indicates that GPIO Port H is present. |
| 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. |
| 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 23: Device Capabilities 5 (DC5), offset 0x020

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

Device Capabilities 5 (DC5)

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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 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 | | | | | | | | | | | | | | | |
| 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 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 24: Device Capabilities 6 (DC6), offset 0x024

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

Device Capabilities 6 (DC6)

Base 0x400F.E000

Offset 0x024

Type RO, reset 0x0000.0013

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|---------|----------|----|------|----|
| | 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 | | | | | | | | | | | USB0PHY | reserved | | USB0 | |
| 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 | 1 | 0 | 0 | 1 | 1 |

| 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 | USB0PHY | RO | 1 | USB Module 0 PHY Present When set, indicates that the USB module 0 PHY is present. |
| 3:2 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1:0 | USB0 | RO | 0x3 | USB Module 0 Present This field indicates that USB module 0 is present and specifies its capability. Value Description 0x3 USB0 is OTG. |

Register 25: Device Capabilities 7 (DC7), offset 0x028

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

Device Capabilities 7 (DC7)

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

| | | | | | | | | | | | | | | | | |
|-------|----------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | DMACH30 | DMACH29 | DMACH28 | DMACH27 | DMACH26 | DMACH25 | DMACH24 | DMACH23 | DMACH22 | DMACH21 | DMACH20 | DMACH19 | DMACH18 | DMACH17 | DMACH16 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DMACH15 | DMACH14 | DMACH13 | DMACH12 | DMACH11 | DMACH10 | DMACH9 | DMACH8 | DMACH7 | DMACH6 | DMACH5 | DMACH4 | DMACH3 | DMACH2 | DMACH1 | DMACH0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | reserved | RO | 1 | Reserved Reserved for uDMA channel 31. |
| 30 | DMACH30 | RO | 1 | SW When set, indicates uDMA channel 30 is available for software transfers. |
| 29 | DMACH29 | RO | 1 | I2S0_TX / CAN1_TX When set, indicates uDMA channel 29 is available and connected to the transmit path of I2S module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of CAN module 1 transmit. |
| 28 | DMACH28 | RO | 1 | I2S0_RX / CAN1_RX When set, indicates uDMA channel 28 is available and connected to the receive path of I2S module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of CAN module 1 receive. |
| 27 | DMACH27 | RO | 1 | CAN1_TX / ADC1_SS3 When set, indicates uDMA channel 27 is available and connected to the transmit path of CAN module 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 3. |
| 26 | DMACH26 | RO | 1 | CAN1_RX / ADC1_SS2 When set, indicates uDMA channel 26 is available and connected to the receive path of CAN module 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 2. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 25 | DMACH25 | RO | 1 | SSI1_TX / ADC1_SS1 When set, indicates uDMA channel 25 is available and connected to the transmit path of SSI module 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 1. |
| 24 | DMACH24 | RO | 1 | SSI1_RX / ADC1_SS0 When set, indicates uDMA channel 24 is available and connected to the receive path of SSI module 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 0. |
| 23 | DMACH23 | RO | 1 | UART1_TX / CAN2_TX When set, indicates uDMA channel 23 is available and connected to the transmit path of UART module 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of CAN module 2 transmit. |
| 22 | DMACH22 | RO | 1 | UART1_RX / CAN2_RX When set, indicates uDMA channel 22 is available and connected to the receive path of UART module 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of CAN module 2 receive. |
| 21 | DMACH21 | RO | 1 | Timer1B / EPI0_TX When set, indicates uDMA channel 21 is available and connected to Timer 1B. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of EPI module 0 transmit. |
| 20 | DMACH20 | RO | 1 | Timer1A / EPI0_RX When set, indicates uDMA channel 20 is available and connected to Timer 1A. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of EPI module 0 receive. |
| 19 | DMACH19 | RO | 1 | Timer0B / Timer1B When set, indicates uDMA channel 19 is available and connected to Timer 0B. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 1B. |
| 18 | DMACH18 | RO | 1 | Timer0A / Timer1A When set, indicates uDMA channel 18 is available and connected to Timer 0A. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 1A. |
| 17 | DMACH17 | RO | 1 | ADC0_SS3 When set, indicates uDMA channel 17 is available and connected to ADC module 0 Sample Sequencer 3. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 16 | DMACH16 | RO | 1 | ADC0_SS2 When set, indicates uDMA channel 16 is available and connected to ADC module 0 Sample Sequencer 2. |
| 15 | DMACH15 | RO | 1 | ADC0_SS1 / Timer2B When set, indicates uDMA channel 15 is available and connected to ADC module 0 Sample Sequencer 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 2B. |
| 14 | DMACH14 | RO | 1 | ADC0_SS0 / Timer2A When set, indicates uDMA channel 14 is available and connected to ADC module 0 Sample Sequencer 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 2A. |
| 13 | DMACH13 | RO | 1 | CAN0_TX / UART2_TX When set, indicates uDMA channel 13 is available and connected to the transmit path of CAN module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of UART module 2 transmit. |
| 12 | DMACH12 | RO | 1 | CAN0_RX / UART2_RX When set, indicates uDMA channel 12 is available and connected to the receive path of CAN module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of UART module 2 receive. |
| 11 | DMACH11 | RO | 1 | SSI0_TX / UART1_TX When set, indicates uDMA channel 11 is available and connected to the transmit path of SSI module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of UART module 1 transmit. |
| 10 | DMACH10 | RO | 1 | SSI0_RX / UART1_RX When set, indicates uDMA channel 10 is available and connected to the receive path of SSI module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of UART module 1 receive. |
| 9 | DMACH9 | RO | 1 | UART0_TX / SSI1_TX When set, indicates uDMA channel 9 is available and connected to the transmit path of UART module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of SSI module 1 transmit. |
| 8 | DMACH8 | RO | 1 | UART0_RX / SSI1_RX When set, indicates uDMA channel 8 is available and connected to the receive path of UART module 0. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of SSI module 1 receive. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 7 | DMACH7 | RO | 1 | ETH_TX / Timer2B When set, indicates uDMA channel 7 is available and connected to the transmit path of the Ethernet module. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 2B. |
| 6 | DMACH6 | RO | 1 | ETH_RX / Timer2A When set, indicates uDMA channel 6 is available and connected to the receive path of the Ethernet module. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 2A. |
| 5 | DMACH5 | RO | 1 | USB_EP3_TX / Timer2B When set, indicates uDMA channel 5 is available and connected to the transmit path of USB endpoint 3. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 2B. |
| 4 | DMACH4 | RO | 1 | USB_EP3_RX / Timer2A When set, indicates uDMA channel 4 is available and connected to the receive path of USB endpoint 3. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 2A. |
| 3 | DMACH3 | RO | 1 | USB_EP2_TX / Timer3B When set, indicates uDMA channel 3 is available and connected to the transmit path of USB endpoint 2. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 3B. |
| 2 | DMACH2 | RO | 1 | USB_EP2_RX / Timer3A When set, indicates uDMA channel 2 is available and connected to the receive path of USB endpoint 2. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of Timer 3A. |
| 1 | DMACH1 | RO | 1 | USB_EP1_TX / UART2_TX When set, indicates uDMA channel 1 is available and connected to the transmit path of USB endpoint 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of UART module 2 transmit. |
| 0 | DMACH0 | RO | 1 | USB_EP1_RX / UART2_RX When set, indicates uDMA channel 0 is available and connected to the receive path of USB endpoint 1. If the corresponding bit in the DMACHALT register is set, the channel is connected instead to the alternate channel assignment of UART module 2 receive. |

Register 26: Device Capabilities 8 ADC Channels (DC8), offset 0x02C

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

Device Capabilities 8 ADC Channels (DC8)

Base 0x400F.E000

Offset 0x02C

Type RO, reset 0xFFFF.FFFF

| | | | | | | | | | | | | | | | | |
|-------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | ADC1AIN15 | ADC1AIN14 | ADC1AIN13 | ADC1AIN12 | ADC1AIN11 | ADC1AIN10 | ADC1AIN9 | ADC1AIN8 | ADC1AIN7 | ADC1AIN6 | ADC1AIN5 | ADC1AIN4 | ADC1AIN3 | ADC1AIN2 | ADC1AIN1 | ADC1AIN0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ADC0AIN15 | ADC0AIN14 | ADC0AIN13 | ADC0AIN12 | ADC0AIN11 | ADC0AIN10 | ADC0AIN9 | ADC0AIN8 | ADC0AIN7 | ADC0AIN6 | ADC0AIN5 | ADC0AIN4 | ADC0AIN3 | ADC0AIN2 | ADC0AIN1 | ADC0AIN0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------|------|-------|--|
| 31 | ADC1AIN15 | RO | 1 | ADC Module 1 AIN15 Pin Present When set, indicates that ADC module 1 input pin 15 is present. |
| 30 | ADC1AIN14 | RO | 1 | ADC Module 1 AIN14 Pin Present When set, indicates that ADC module 1 input pin 14 is present. |
| 29 | ADC1AIN13 | RO | 1 | ADC Module 1 AIN13 Pin Present When set, indicates that ADC module 1 input pin 13 is present. |
| 28 | ADC1AIN12 | RO | 1 | ADC Module 1 AIN12 Pin Present When set, indicates that ADC module 1 input pin 12 is present. |
| 27 | ADC1AIN11 | RO | 1 | ADC Module 1 AIN11 Pin Present When set, indicates that ADC module 1 input pin 11 is present. |
| 26 | ADC1AIN10 | RO | 1 | ADC Module 1 AIN10 Pin Present When set, indicates that ADC module 1 input pin 10 is present. |
| 25 | ADC1AIN9 | RO | 1 | ADC Module 1 AIN9 Pin Present When set, indicates that ADC module 1 input pin 9 is present. |
| 24 | ADC1AIN8 | RO | 1 | ADC Module 1 AIN8 Pin Present When set, indicates that ADC module 1 input pin 8 is present. |
| 23 | ADC1AIN7 | RO | 1 | ADC Module 1 AIN7 Pin Present When set, indicates that ADC module 1 input pin 7 is present. |
| 22 | ADC1AIN6 | RO | 1 | ADC Module 1 AIN6 Pin Present When set, indicates that ADC module 1 input pin 6 is present. |
| 21 | ADC1AIN5 | RO | 1 | ADC Module 1 AIN5 Pin Present When set, indicates that ADC module 1 input pin 5 is present. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------|------|-------|--|
| 20 | ADC1AIN4 | RO | 1 | ADC Module 1 AIN4 Pin Present When set, indicates that ADC module 1 input pin 4 is present. |
| 19 | ADC1AIN3 | RO | 1 | ADC Module 1 AIN3 Pin Present When set, indicates that ADC module 1 input pin 3 is present. |
| 18 | ADC1AIN2 | RO | 1 | ADC Module 1 AIN2 Pin Present When set, indicates that ADC module 1 input pin 2 is present. |
| 17 | ADC1AIN1 | RO | 1 | ADC Module 1 AIN1 Pin Present When set, indicates that ADC module 1 input pin 1 is present. |
| 16 | ADC1AIN0 | RO | 1 | ADC Module 1 AIN0 Pin Present When set, indicates that ADC module 1 input pin 0 is present. |
| 15 | ADC0AIN15 | RO | 1 | ADC Module 0 AIN15 Pin Present When set, indicates that ADC module 0 input pin 15 is present. |
| 14 | ADC0AIN14 | RO | 1 | ADC Module 0 AIN14 Pin Present When set, indicates that ADC module 0 input pin 14 is present. |
| 13 | ADC0AIN13 | RO | 1 | ADC Module 0 AIN13 Pin Present When set, indicates that ADC module 0 input pin 13 is present. |
| 12 | ADC0AIN12 | RO | 1 | ADC Module 0 AIN12 Pin Present When set, indicates that ADC module 0 input pin 12 is present. |
| 11 | ADC0AIN11 | RO | 1 | ADC Module 0 AIN11 Pin Present When set, indicates that ADC module 0 input pin 11 is present. |
| 10 | ADC0AIN10 | RO | 1 | ADC Module 0 AIN10 Pin Present When set, indicates that ADC module 0 input pin 10 is present. |
| 9 | ADC0AIN9 | RO | 1 | ADC Module 0 AIN9 Pin Present When set, indicates that ADC module 0 input pin 9 is present. |
| 8 | ADC0AIN8 | RO | 1 | ADC Module 0 AIN8 Pin Present When set, indicates that ADC module 0 input pin 8 is present. |
| 7 | ADC0AIN7 | RO | 1 | ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present. |
| 6 | ADC0AIN6 | RO | 1 | ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present. |
| 5 | ADC0AIN5 | RO | 1 | ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present. |
| 4 | ADC0AIN4 | RO | 1 | ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 3 | ADC0AIN3 | RO | 1 | ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present. |
| 2 | ADC0AIN2 | RO | 1 | ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present. |
| 1 | ADC0AIN1 | RO | 1 | ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present. |
| 0 | ADC0AIN0 | RO | 1 | ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present. |

Register 27: Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190

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

Device Capabilities 9 ADC Digital Comparators (DC9)

Base 0x400F.E000

Offset 0x190

Type RO, reset 0x00FF.00FF

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|---------|---------|---------|---------|---------|---------|---------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | ADC1DC7 | ADC1DC6 | ADC1DC5 | ADC1DC4 | ADC1DC3 | ADC1DC2 | ADC1DC1 | ADC1DC0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | ADC0DC7 | ADC0DC6 | ADC0DC5 | ADC0DC4 | ADC0DC3 | ADC0DC2 | ADC0DC1 | ADC0DC0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:24 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 23 | ADC1DC7 | RO | 1 | ADC1 DC7 Present When set, indicates that ADC module 1 Digital Comparator 7 is present. |
| 22 | ADC1DC6 | RO | 1 | ADC1 DC6 Present When set, indicates that ADC module 1 Digital Comparator 6 is present. |
| 21 | ADC1DC5 | RO | 1 | ADC1 DC5 Present When set, indicates that ADC module 1 Digital Comparator 5 is present. |
| 20 | ADC1DC4 | RO | 1 | ADC1 DC4 Present When set, indicates that ADC module 1 Digital Comparator 4 is present. |
| 19 | ADC1DC3 | RO | 1 | ADC1 DC3 Present When set, indicates that ADC module 1 Digital Comparator 3 is present. |
| 18 | ADC1DC2 | RO | 1 | ADC1 DC2 Present When set, indicates that ADC module 1 Digital Comparator 2 is present. |
| 17 | ADC1DC1 | RO | 1 | ADC1 DC1 Present When set, indicates that ADC module 1 Digital Comparator 1 is present. |
| 16 | ADC1DC0 | RO | 1 | ADC1 DC0 Present When set, indicates that ADC module 1 Digital Comparator 0 is present. |
| 15:8 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7 | ADC0DC7 | RO | 1 | ADC0 DC7 Present When set, indicates that ADC module 0 Digital Comparator 7 is present. |

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

Register 28: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0

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

Non-Volatile Memory Information (NVMSTAT)

Base 0x400F.E000

Offset 0x1A0

Type RO, reset 0x0000.0001

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| | 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 | | | | | | | | | | | | | | | FWB |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:1 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | FWB | RO | 1 | 32 Word Flash Write Buffer Active When set, indicates that the 32 word Flash memory write buffer feature is active. |

Register 29: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

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

Run Mode Clock Gating Control Register 0 (RCGC0)

Base 0x400F.E000

Offset 0x100

Type R/W, reset 0x00000040

| | | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|------|------------|-----|------------|------|----------|-----|----------|----|------|----------|-----|-----|------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| | reserved | | | WDT1 | reserved | | CAN1 | CAN0 | reserved | | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | R/W | RO | RO | R/W | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| | reserved | | | | MAXADC1SPD | | MAXADC0SPD | | reserved | HIB | reserved | | WDT0 | reserved | | | | |
| Type | RO | RO | RO | RO | R/W | R/W | R/W | R/W | RO | R/W | RO | RO | R/W | RO | RO | RO | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | WDT1 | R/W | 0 | WDT1 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 27:26 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 25 | CAN1 | R/W | 0 | CAN1 Clock Gating Control This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 24 | CAN0 | R/W | 0 | CAN0 Clock Gating Control This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---------------------|------|-------|--|-------|-------------|-----|-------------------|-----|---------------------|-----|---------------------|-----|---------------------|
| 23:18 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 17 | ADC1 | R/W | 0 | ADC1 Clock Gating Control This bit controls the clock gating for SAR ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. | | | | | | | | | | |
| 16 | ADC0 | R/W | 0 | ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. | | | | | | | | | | |
| 15:12 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 11:10 | MAXADC1SPD | R/W | 0 | ADC1 Sample Speed This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved): <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table> | Value | Description | 0x3 | 1M samples/second | 0x2 | 500K samples/second | 0x1 | 250K samples/second | 0x0 | 125K samples/second |
| Value | Description | | | | | | | | | | | | | |
| 0x3 | 1M samples/second | | | | | | | | | | | | | |
| 0x2 | 500K samples/second | | | | | | | | | | | | | |
| 0x1 | 250K samples/second | | | | | | | | | | | | | |
| 0x0 | 125K samples/second | | | | | | | | | | | | | |
| 9:8 | MAXADC0SPD | R/W | 0 | ADC0 Sample Speed This field sets the rate at which ADC0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved): <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table> | Value | Description | 0x3 | 1M samples/second | 0x2 | 500K samples/second | 0x1 | 250K samples/second | 0x0 | 125K samples/second |
| 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. | | | | | | | | | | |

| 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 | WDT0 | R/W | 0 | WDT0 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 2:0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 30: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

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

Sleep Mode Clock Gating Control Register 0 (SCGC0)

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

| | | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|------|------------|-----|------------|------|----------|-----|----------|----|------|----------|-----|-----|------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| | reserved | | | WDT1 | reserved | | CAN1 | CAN0 | reserved | | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | R/W | RO | RO | R/W | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| | reserved | | | | MAXADC1SPD | | MAXADC0SPD | | reserved | HIB | reserved | | WDT0 | reserved | | | | |
| Type | RO | RO | RO | RO | R/W | R/W | R/W | R/W | RO | R/W | RO | RO | R/W | RO | RO | RO | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | WDT1 | R/W | 0 | WDT1 Clock Gating Control This bit controls the clock gating for Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 27:26 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 25 | CAN1 | R/W | 0 | CAN1 Clock Gating Control This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 24 | CAN0 | R/W | 0 | CAN0 Clock Gating Control This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---------------------|------|-------|--|-------|-------------|-----|-------------------|-----|---------------------|-----|---------------------|-----|---------------------|
| 23:18 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 17 | ADC1 | R/W | 0 | <p>ADC1 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> | | | | | | | | | | |
| 16 | ADC0 | R/W | 0 | <p>ADC0 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> | | | | | | | | | | |
| 15:12 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 11:10 | MAXADC1SPD | R/W | 0 | <p>ADC1 Sample Speed</p> <p>This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table> | Value | Description | 0x3 | 1M samples/second | 0x2 | 500K samples/second | 0x1 | 250K samples/second | 0x0 | 125K samples/second |
| Value | Description | | | | | | | | | | | | | |
| 0x3 | 1M samples/second | | | | | | | | | | | | | |
| 0x2 | 500K samples/second | | | | | | | | | | | | | |
| 0x1 | 250K samples/second | | | | | | | | | | | | | |
| 0x0 | 125K samples/second | | | | | | | | | | | | | |
| 9:8 | MAXADC0SPD | R/W | 0 | <p>ADC0 Sample Speed</p> <p>This field sets the rate at which ADC module 0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table> | Value | Description | 0x3 | 1M samples/second | 0x2 | 500K samples/second | 0x1 | 250K samples/second | 0x0 | 125K samples/second |
| 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. | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 6 | HIB | R/W | 1 | <p>HIB Clock Gating Control</p> <p>This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 5:4 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 3 | WDT0 | R/W | 0 | <p>WDT0 Clock Gating Control</p> <p>This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 2:0 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |

Register 31: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

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

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

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

| | | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|------|------------|-----|------------|------|----------|-----|----------|----|------|----------|-----|-----|------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| | reserved | | | WDT1 | reserved | | CAN1 | CAN0 | reserved | | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | R/W | RO | RO | R/W | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| | reserved | | | | MAXADC1SPD | | MAXADC0SPD | | reserved | HIB | reserved | | WDT0 | reserved | | | | |
| Type | RO | RO | RO | RO | R/W | R/W | R/W | R/W | RO | R/W | RO | RO | R/W | RO | RO | RO | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | WDT1 | R/W | 0 | WDT1 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 27:26 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 25 | CAN1 | R/W | 0 | CAN1 Clock Gating Control This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 24 | CAN0 | R/W | 0 | CAN0 Clock Gating Control This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---------------------|------|-------|--|-------|-------------|-----|-------------------|-----|---------------------|-----|---------------------|-----|---------------------|
| 23:18 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 17 | ADC1 | R/W | 0 | ADC1 Clock Gating Control This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. | | | | | | | | | | |
| 16 | ADC0 | R/W | 0 | ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. | | | | | | | | | | |
| 15:12 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 11:10 | MAXADC1SPD | R/W | 0 | ADC1 Sample Speed This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved): <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table> | Value | Description | 0x3 | 1M samples/second | 0x2 | 500K samples/second | 0x1 | 250K samples/second | 0x0 | 125K samples/second |
| Value | Description | | | | | | | | | | | | | |
| 0x3 | 1M samples/second | | | | | | | | | | | | | |
| 0x2 | 500K samples/second | | | | | | | | | | | | | |
| 0x1 | 250K samples/second | | | | | | | | | | | | | |
| 0x0 | 125K samples/second | | | | | | | | | | | | | |
| 9:8 | MAXADC0SPD | R/W | 0 | ADC0 Sample Speed This field sets the rate at which ADC module 0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved): <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table> | Value | Description | 0x3 | 1M samples/second | 0x2 | 500K samples/second | 0x1 | 250K samples/second | 0x0 | 125K samples/second |
| 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. | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 6 | HIB | R/W | 1 | HIB Clock Gating Control This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 5:4 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | WDT0 | R/W | 0 | WDT0 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 2:0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 32: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

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

Run Mode Clock Gating Control Register 1 (RCGC1)

Base 0x400F.E000

Offset 0x104

Type R/W, reset 0x00000000

| | | | | | | | | | | | | | | | | |
|-------|----------|------|----------|------|----------|-------|-------|-------|----------|----|------|--------|----------|--------|--------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | EPI0 | reserved | I2S0 | reserved | COMP2 | COMP1 | COMP0 | reserved | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 | |
| Type | RO | R/W | RO | R/W | 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 | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | reserved | RO | 0 | Software should not rely on the value of 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 | EPI0 | R/W | 0 | EPI0 Clock Gating This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 | I2S0 | R/W | 0 | I2S0 Clock Gating This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 26 | COMP2 | R/W | 0 | <p>Analog Comparator 2 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 25 | COMP1 | R/W | 0 | <p>Analog Comparator 1 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 24 | COMP0 | R/W | 0 | <p>Analog Comparator 0 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 23:20 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 19 | TIMER3 | R/W | 0 | <p>Timer 3 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 18 | TIMER2 | R/W | 0 | <p>Timer 2 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 17 | TIMER1 | R/W | 0 | <p>Timer 1 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 16 | TIMER0 | R/W | 0 | <p>Timer 0 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 15 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 14 | I2C1 | R/W | 0 | <p>I2C1 Clock Gating Control</p> <p>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |

| 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 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 11:6 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | SSI1 | R/W | 0 | SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 4 | SSI0 | R/W | 0 | SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | UART2 | R/W | 0 | UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

Register 33: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

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

Sleep Mode Clock Gating Control Register 1 (SCGC1)

Base 0x400F.E000

Offset 0x114

Type R/W, reset 0x00000000

| | | | | | | | | | | | | | | | | |
|-------|----------|------|----------|------|----------|-------|-------|-------|----------|----|------|--------|----------|--------|--------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | EPI0 | reserved | I2S0 | reserved | COMP2 | COMP1 | COMP0 | reserved | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 | |
| Type | RO | R/W | RO | R/W | 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 | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | reserved | RO | 0 | Software should not rely on the value of 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 | EPI0 | R/W | 0 | EPI0 Clock Gating This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | I2S0 | R/W | 0 | I2S0 Clock Gating This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 26 | COMP2 | R/W | 0 | <p>Analog Comparator 2 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 25 | COMP1 | R/W | 0 | <p>Analog Comparator 1 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 24 | COMP0 | R/W | 0 | <p>Analog Comparator 0 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 23:20 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 19 | TIMER3 | R/W | 0 | <p>Timer 3 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 18 | TIMER2 | R/W | 0 | <p>Timer 2 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 17 | TIMER1 | R/W | 0 | <p>Timer 1 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 16 | TIMER0 | R/W | 0 | <p>Timer 0 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 15 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 14 | I2C1 | R/W | 0 | <p>I2C1 Clock Gating Control</p> <p>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |

| 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 | <p>I2C0 Clock Gating Control</p> <p>This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 11:6 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | SSI1 | R/W | 0 | <p>SSI1 Clock Gating Control</p> <p>This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 4 | SSI0 | R/W | 0 | <p>SSI0 Clock Gating Control</p> <p>This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 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 | <p>UART2 Clock Gating Control</p> <p>This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 1 | UART1 | R/W | 0 | <p>UART1 Clock Gating Control</p> <p>This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 0 | UART0 | R/W | 0 | <p>UART0 Clock Gating Control</p> <p>This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |

Register 34: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

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

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

Base 0x400F.E000

Offset 0x124

Type R/W, reset 0x00000000

| | | | | | | | | | | | | | | | | |
|-------|----------|------|----------|------|----------|-------|-------|-------|----------|----|------|--------|----------|--------|--------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | EPI0 | reserved | I2S0 | reserved | COMP2 | COMP1 | COMP0 | reserved | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 | |
| Type | RO | R/W | RO | R/W | 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 | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | reserved | RO | 0 | Software should not rely on the value of 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 | EPI0 | R/W | 0 | EPI0 Clock Gating This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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 | I2S0 | R/W | 0 | I2S0 Clock Gating This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault. |
| 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 26 | COMP2 | R/W | 0 | <p>Analog Comparator 2 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 25 | COMP1 | R/W | 0 | <p>Analog Comparator 1 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 24 | COMP0 | R/W | 0 | <p>Analog Comparator 0 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 23:20 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 19 | TIMER3 | R/W | 0 | <p>Timer 3 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 18 | TIMER2 | R/W | 0 | <p>Timer 2 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 17 | TIMER1 | R/W | 0 | <p>Timer 1 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 16 | TIMER0 | R/W | 0 | <p>Timer 0 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 15 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 14 | I2C1 | R/W | 0 | <p>I2C1 Clock Gating Control</p> <p>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |

| 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 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 11:6 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | SSI1 | R/W | 0 | SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 4 | SSI0 | R/W | 0 | SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | UART2 | R/W | 0 | UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

Register 35: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

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

Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000

Offset 0x108

Type R/W, reset 0x00000000

| | | | | | | | | | | | | | | | | |
|-------|----------|-------|----------|----------|----------|----|----|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | EPHY0 | reserved | EMAC0 | reserved | | | | | | | | | | | USB0 |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | UDMA | reserved | | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | GIPOD | GPIOC | GPIOB | GPIOA |
| Type | RO | RO | R/W | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | reserved | RO | 0 | Software should not rely on the value of 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 layer 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 layer 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 27:17 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 16 | USB0 | R/W | 0 | <p>USB0 Clock Gating Control</p> <p>This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 15:14 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 13 | UDMA | R/W | 0 | <p>Micro-DMA Clock Gating Control</p> <p>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 12:9 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 8 | GPIOJ | R/W | 0 | <p>Port J Clock Gating Control</p> <p>This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 7 | GPIOH | R/W | 0 | <p>Port H Clock Gating Control</p> <p>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 6 | GPIOG | R/W | 0 | <p>Port G Clock Gating Control</p> <p>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 5 | GPIOF | R/W | 0 | <p>Port F Clock Gating Control</p> <p>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 4 | GPIOE | R/W | 0 | <p>Port E Clock Gating Control</p> <p>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 3 | GPIOD | R/W | 0 | <p>Port D Clock Gating Control</p> <p>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|--|
| 2 | GPIOC | R/W | 0 | Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

Register 36: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

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

Sleep Mode Clock Gating Control Register 2 (SCGC2)

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

| | | | | | | | | | | | | | | | | | |
|-------|----------|-------|----------|----------|----------|----|----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | EPHY0 | reserved | EMAC0 | reserved | | | | | | | | | | | USB0 | |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | UDMA | reserved | | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA | |
| Type | RO | RO | R/W | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 layer 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 layer 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 27:17 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 16 | USB0 | R/W | 0 | <p>USB0 Clock Gating Control</p> <p>This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 15:14 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 13 | UDMA | R/W | 0 | <p>Micro-DMA Clock Gating Control</p> <p>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 12:9 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 8 | GPIOJ | R/W | 0 | <p>Port J Clock Gating Control</p> <p>This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 7 | GPIOH | R/W | 0 | <p>Port H Clock Gating Control</p> <p>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 6 | GPIOG | R/W | 0 | <p>Port G Clock Gating Control</p> <p>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 5 | GPIOF | R/W | 0 | <p>Port F Clock Gating Control</p> <p>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 4 | GPIOE | R/W | 0 | <p>Port E Clock Gating Control</p> <p>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 3 | GPIOD | R/W | 0 | <p>Port D Clock Gating Control</p> <p>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|--|
| 2 | GPIOC | R/W | 0 | Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

Register 37: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

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

Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

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

| | | | | | | | | | | | | | | | | | |
|-------|----------|-------|----------|----------|----------|----|----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | EPHY0 | reserved | EMAC0 | reserved | | | | | | | | | | | USB0 | |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | UDMA | reserved | | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA | |
| Type | RO | RO | R/W | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 layer 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 layer 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |
| 27:17 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 16 | USB0 | R/W | 0 | <p>USB0 Clock Gating Control</p> <p>This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 15:14 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 13 | UDMA | R/W | 0 | <p>Micro-DMA Clock Gating Control</p> <p>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 12:9 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 8 | GPIOJ | R/W | 0 | <p>Port J Clock Gating Control</p> <p>This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 7 | GPIOH | R/W | 0 | <p>Port H Clock Gating Control</p> <p>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 6 | GPIOG | R/W | 0 | <p>Port G Clock Gating Control</p> <p>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 5 | GPIOF | R/W | 0 | <p>Port F Clock Gating Control</p> <p>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 4 | GPIOE | R/W | 0 | <p>Port E Clock Gating Control</p> <p>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |
| 3 | GPIOD | R/W | 0 | <p>Port D Clock Gating Control</p> <p>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|--|
| 2 | GPIOC | R/W | 0 | Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault. |

Register 38: Software Reset Control 0 (SRCR0), offset 0x040

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

Software Reset Control 0 (SRCR0)

Base 0x400F.E000

Offset 0x040

Type R/W, reset 0x00000000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|------|----------|----|------|------|----------|-----|-----|----------|-----|------|----------|------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | WDT1 | reserved | | CAN1 | CAN0 | reserved | | | | | | | ADC1 | ADC0 |
| Type | RO | RO | RO | R/W | RO | RO | R/W | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | HIB | reserved | | WDT0 | reserved | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | RO | RO | R/W | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31:29 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | WDT1 | R/W | 0 | WDT1 Reset Control When this bit is set, Watchdog Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 27:26 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 25 | CAN1 | R/W | 0 | CAN1 Reset Control When this bit is set, CAN module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 24 | CAN0 | R/W | 0 | CAN0 Reset Control When this bit is set, CAN module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 23:18 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 17 | ADC1 | R/W | 0 | ADC1 Reset Control When this bit is set, ADC module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 16 | ADC0 | R/W | 0 | ADC0 Reset Control When this bit is set, ADC module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 15:7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6 | HIB | R/W | 0 | HIB Reset Control When this bit is set, the Hibernation module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 5:4 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | WDT0 | R/W | 0 | WDT0 Reset Control When this bit is set, Watchdog Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 2:0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 39: Software Reset Control 1 (SRCR1), offset 0x044

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

Software Reset Control 1 (SRCR1)

Base 0x400F.E000

Offset 0x044

Type R/W, reset 0x00000000

| | | | | | | | | | | | | | | | | |
|-------|----------|------|----------|------|----------|-------|-------|-------|----------|----|------|--------|----------|--------|--------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | EPI0 | reserved | I2S0 | reserved | COMP2 | COMP1 | COMP0 | reserved | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 | |
| Type | RO | R/W | RO | R/W | 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 | | | | | | SSI1 | SSI0 | reserved | UART2 | UART1 | UART0 |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | reserved | RO | 0 | Software should not rely on the value of 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 | EPI0 | R/W | 0 | EPI0 Reset Control When this bit is set, EPI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 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 | I2S0 | R/W | 0 | I2S0 Reset Control When this bit is set, I2S module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 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 When this bit is set, Analog Comparator module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 25 | COMP1 | R/W | 0 | Analog Comp 1 Reset Control When this bit is set, Analog Comparator module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 24 | COMP0 | R/W | 0 | Analog Comp 0 Reset Control When this bit is set, Analog Comparator module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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 Timer 3 Reset Control. When this bit is set, General-Purpose Timer module 3 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 18 | TIMER2 | R/W | 0 | Timer 2 Reset Control When this bit is set, General-Purpose Timer module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 17 | TIMER1 | R/W | 0 | Timer 1 Reset Control When this bit is set, General-Purpose Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 16 | TIMER0 | R/W | 0 | Timer 0 Reset Control When this bit is set, General-Purpose Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 15 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 14 | I2C1 | R/W | 0 | I2C1 Reset Control When this bit is set, I2C module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 13 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 12 | I2C0 | R/W | 0 | I2C0 Reset Control When this bit is set, I2C module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 11:6 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | SSI1 | R/W | 0 | SSI1 Reset Control When this bit is set, SSI module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 4 | SSI0 | R/W | 0 | SSI0 Reset Control When this bit is set, SSI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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 When this bit is set, UART module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 1 | UART1 | R/W | 0 | UART1 Reset Control When this bit is set, UART module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 0 | UART0 | R/W | 0 | UART0 Reset Control When this bit is set, UART module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

Register 40: Software Reset Control 2 (SRCR2), offset 0x048

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

Software Reset Control 2 (SRCR2)

Base 0x400F.E000

Offset 0x048

Type R/W, reset 0x00000000

| | | | | | | | | | | | | | | | | | |
|-------|----------|-------|----------|----------|----------|----|----|-------|-------|-------|-------|-------|-------|-------|-------|------|--|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | EPHY0 | reserved | EMAC0 | reserved | | | | | | | | | | | USB0 | |
| Type | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | UDMA | reserved | | | | GPIOD | GPIOC | GPIOB | GPIOA | GPIOD | GPIOC | GPIOB | GPIOA | | |
| Type | RO | RO | R/W | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31 | reserved | RO | 0 | Software should not rely on the value of 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 When this bit is set, Ethernet PHY layer 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 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 When this bit is set, Ethernet MAC layer 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 27:17 | 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 | USB0 | R/W | 0 | USB0 Reset Control When this bit is set, USB module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 15:14 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 13 | UDMA | R/W | 0 | Micro-DMA Reset Control When this bit is set, uDMA module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 12:9 | 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 | GPIOJ | R/W | 0 | Port J Reset Control When this bit is set, Port J module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 7 | GPIOH | R/W | 0 | Port H Reset Control When this bit is set, Port H module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 6 | GPIOG | R/W | 0 | Port G Reset Control When this bit is set, Port G module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 5 | GPIOF | R/W | 0 | Port F Reset Control When this bit is set, Port F module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 4 | GPIOE | R/W | 0 | Port E Reset Control When this bit is set, Port E module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 3 | GPIOD | R/W | 0 | Port D Reset Control When this bit is set, Port D module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 2 | GPIOC | R/W | 0 | Port C Reset Control When this bit is set, Port C module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 1 | GPIOB | R/W | 0 | Port B Reset Control When this bit is set, Port B module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |
| 0 | GPIOA | R/W | 0 | Port A Reset Control When this bit is set, Port A module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set. |

7 Hibernation Module

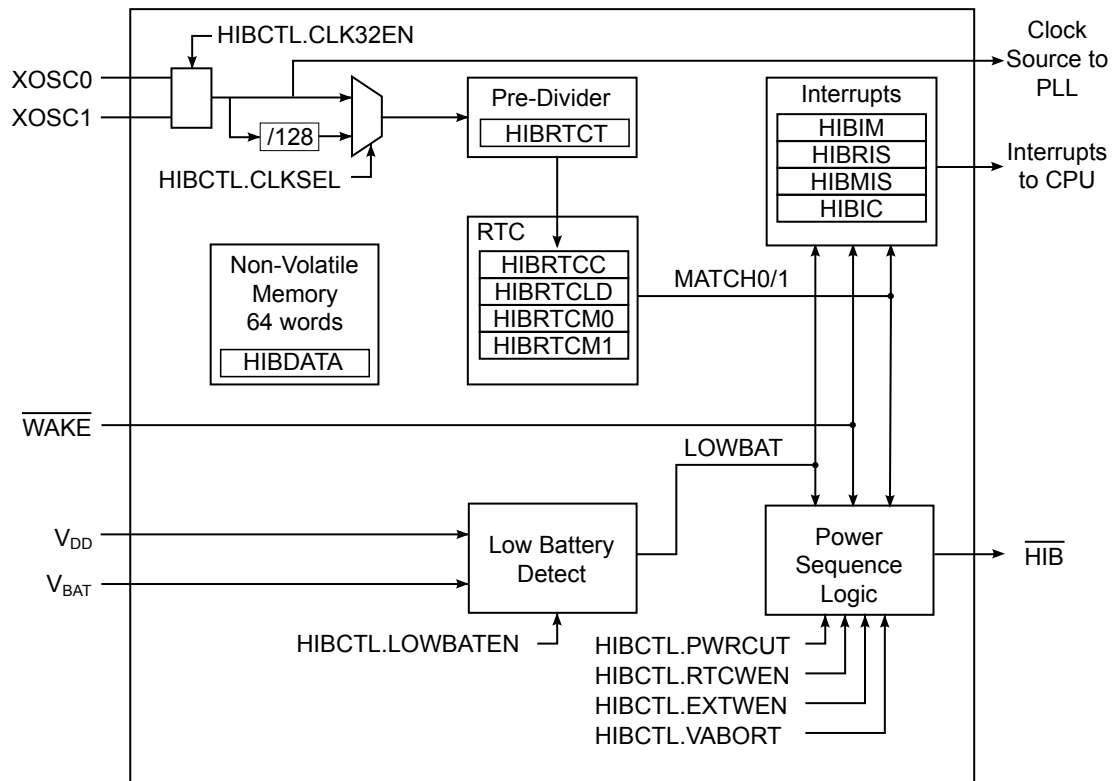
The Hibernation Module manages removal and restoration of power to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal or at a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

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

7.1 Block Diagram

Figure 7-1. Hibernation Module Block Diagram



7.2 Signal Description

Table 7-1 on page 201 lists the external signals of the Hibernation module and describes the function of each. These signals have dedicated functions and are not alternate functions for any GPIO signals.

Table 7-1. Signals for Hibernate

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|--------------------------|------------|--------------------------|----------|--------------------------|---|
| $\overline{\text{HIB}}$ | 51 | fixed | O | OD | An open-drain output that indicates the processor is in Hibernate mode. |
| VBAT | 55 | fixed | - | Power | Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply. |
| $\overline{\text{WAKE}}$ | 50 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| XOSC0 | 52 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the <code>CLKSEL</code> bit in the <code>HIBCTL</code> register. |
| XOSC1 | 53 | fixed | O | Analog | Hibernation module oscillator crystal output. |

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

7.3 Functional Description

Important: The Hibernate module must have either the RTC function or the External Wake function enabled to ensure proper operation of the microcontroller. See “Initialization” on page 206.

The Hibernation module provides two mechanisms for power control:

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

The Hibernation module power source is determined dynamically. The supply voltage of the Hibernation module is the larger of the main voltage source (V_{DD}) or the battery/auxiliary voltage source (V_{BAT}). Care must be taken that the voltage amplitude of the 32-kHz oscillator is less than V_{BAT} , otherwise, the Hibernation module draws power from the oscillator and not V_{BAT} . 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 ($\overline{\text{WAKE}}$) 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.

Power-up from a power cut to code execution is defined as the regulator turn-on time (specified at $t_{\text{HIB_TO_VDD}}$ maximum) plus the normal chip POR (see “Hibernation Module” on page 1048).

7.3.1 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 $t_{\text{HIB_REG_WRITE}}$, therefore software must guarantee that this delay is inserted between back-to-back writes to certain Hibernation registers or between a write followed by a read to those same registers. The timing for back-to-back reads from the Hibernation module has no restrictions. Software may make use of the WRC bit in the **Hibernation Control (HIBCTL)** register to ensure that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **HIBCTL** for $\text{WRC}=1$ prior to accessing any affected register. The following registers are subject to this timing restriction:

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

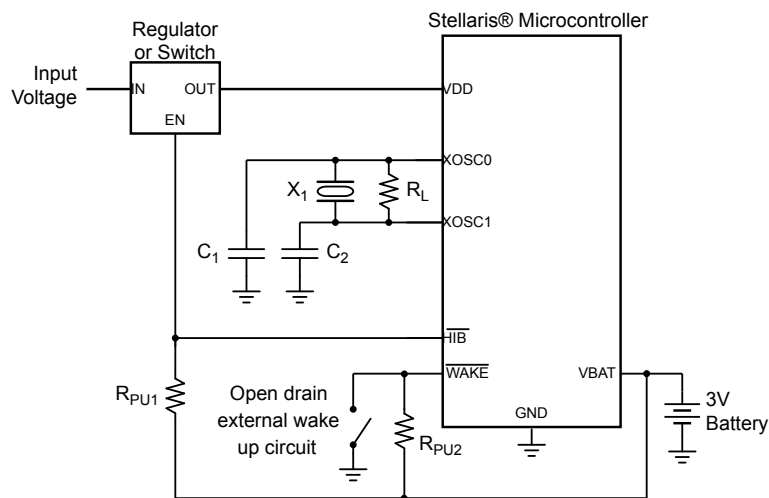
7.3.2 Clock Source

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. Care must be taken that the voltage amplitude of the 32-kHz oscillator is less than V_{BAT} , otherwise, the Hibernation module draws power from the oscillator and not V_{BAT} during hibernation. See Figure 7-2 on page 203 and Figure 7-3 on page 204. Note that these diagrams only show the connection to the Hibernation pins and not to the full system. See “Hibernation Module” on page 1048 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 clearing the `CLKSEL` bit for a 4.194304-MHz clock source and setting the `CLKSEL` bit for 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 writing to the `CLK32EN` bit and before any other accesses to the Hibernation module registers. The delay allows the crystal to power up and stabilize. If an oscillator is used for the clock source, no delay is needed.

Figure 7-2. Clock Source Using Crystal



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

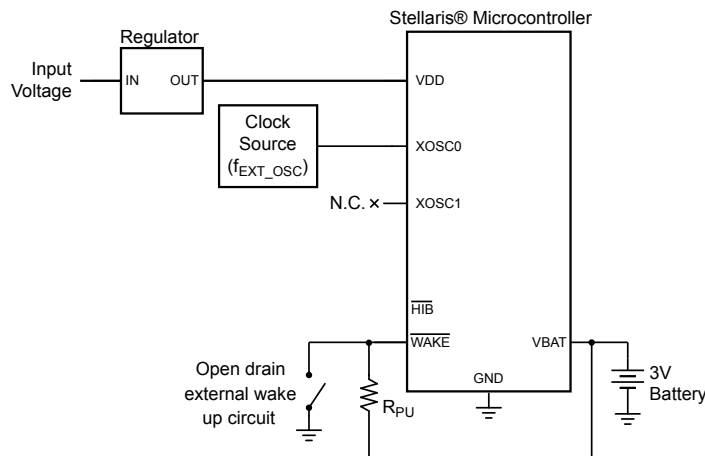
$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

R_L = Load resistor is R_{XOSC_LOAD} .

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

R_{PU2} = Pull-up resistor 2 is 1 M Ω

See “Hibernation Module” on page 1048 for specific parameter values.

Figure 7-3. Clock Source Using Dedicated Oscillator and VDD3ON Mode

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

If the application does not require the use of the Hibernation module, the XOSC0 and XOSC1 can remain unconnected. In this situation, the Hibernation module registers are not accessible.

7.3.3 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 does 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 than 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 Hibernation Raw Interrupt Status (HIBRIS) register is set when the battery level is low. If the VABORT bit in the HIBCTL register is also set, then the module is prevented from entering Hibernate mode when a low battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see “Interrupts and Status” on page 206).

7.3.4 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 202). The 32.768-kHz clock signal is fed into a predivider register that counts down the 32.768-kHz clock ticks to achieve a once per second clock rate for the RTC. The rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register, HIBRTCT. This register has a nominal value of 0x7FFF, and is used for one second out of every 64 seconds to divide the input clock. This configuration

allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from 0x7FFF. The predivider trim should be adjusted up from 0x7FFF in order to slow down the RTC rate and down from 0x7FFF in order to speed up the RTC rate.

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

The RTC must be enabled with the `RTCEN` bit of the `HIBCTL` register. The value of the RTC can be set at any time by writing to the `HIBRTCLD` register. The predivider trim can be adjusted by reading and writing the `HIBRTCT` register. The predivider uses this register once every 64 seconds to adjust the clock rate. The two match registers can be set by writing to the `HIBRTCM0` and `HIBRTCM1` registers. The RTC can be configured to generate interrupts by using the interrupt registers (see “Interrupts and Status” on page 206).

7.3.5 Non-Volatile Memory

The Hibernation module contains 64 32-bit words of memory that are powered from the battery or auxiliary power supply and therefore retained during hibernation. The processor software can save state information in this memory prior to hibernation and recover the state upon waking. The non-volatile memory can be accessed through the `HIBDATA` registers.

7.3.6 Power Control Using $\overline{\text{HIB}}$

Important: The Hibernation Module requires special system implementation considerations when using $\overline{\text{HIB}}$ to control power, as it is intended to power-down all other sections of the microcontroller. All system signals and power supplies that connect to the chip must be driven to 0 V_{DC} or powered down with the same regulator controlled by $\overline{\text{HIB}}$. See “Hibernation Module” on page 1048 for more details.

The Hibernation module controls power to the microcontroller through the use of the $\overline{\text{HIB}}$ pin which is intended to be connected to the enable signal of the external regulator(s) providing 3.3 V to the microcontroller and other circuits. When the $\overline{\text{HIB}}$ signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the microcontroller and any parts of the system that are powered by the regulator. The Hibernation module remains powered from the V_{BAT} supply (which could be a battery or an auxiliary power source) until a Wake event. Power to the microcontroller is restored by deasserting the $\overline{\text{HIB}}$ signal, which causes the external regulator to turn power back on to the chip.

7.3.7 Power Control Using VDD3ON Mode

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

7.3.8 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 $\overline{\text{WAKE}}$ pin, or by using an RTC match. If a Flash memory write operation is in progress, an interlock feature holds off the transition into Hibernation mode until the write has completed.

The Hibernation module is configured to wake from the external $\overline{\text{WAKE}}$ pin by setting the `PINWEN` bit of the `HIBCTL` register. It is configured to wake from RTC match by setting the `RTCWEN` bit. Either

one or both of these bits must be set prior to going into hibernation. Note that the $\overline{\text{WAKE}}$ pin uses the Hibernation module's internal power supply as the logic 1 reference.

Upon either external wake-up or RTC match, the Hibernation module delays coming out of hibernation until V_{DD} is above the minimum specified voltage, see Table 26-2 on page 1039.

When the Hibernation module wakes, the microcontroller performs a normal power-on reset. Software can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see "Interrupts and Status" on page 206) and by looking for state data in the non-volatile memory (see "Non-Volatile Memory" on page 205).

7.3.9 Interrupts and Status

The Hibernation module can generate interrupts when the following conditions occur:

- Assertion of $\overline{\text{WAKE}}$ pin
- RTC match
- Low battery detected

All of the interrupts are ORed together before being sent to the interrupt controller, so the Hibernation module can only generate a single interrupt request to the controller at any given time. The software interrupt handler can service multiple interrupt events by reading the **Hibernation Masked Interrupt Status (HIBMIS)** register. Software can also read the status of the Hibernation module at any time by reading the **HIBRIS** register which shows all of the pending events. This register can be used 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 **Hibernation Interrupt Mask (HIBIM)** register. Pending interrupts can be cleared by writing the corresponding bit in the **Hibernation Interrupt Clear (HIBIC)** register.

7.4 Initialization and Configuration

The Hibernation module has several different configurations. The following sections show the recommended programming sequence for various scenarios. The examples below assume that a 32.768-kHz oscillator is used, and thus always set the **CLKSEL** bit of the **HIBCTL** register. If a 4.194304-MHz crystal is used instead, then the **CLKSEL** bit remains cleared. Because the Hibernation module runs at 32.768 kHz and is asynchronous to the rest of the system, software must allow a delay of $t_{\text{HIB_REG_WRITE}}$ after writes to certain registers (see "Register Access Timing" on page 202). The registers that require a delay are listed in a note in "Register Map" on page 209 as well as in each register description.

7.4.1 Initialization

The Hibernation module comes out of reset with the clock enabled, but if the clock has been disabled, then the clock source must be re-enabled, even if the RTC feature is not used. See page 166.

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 $t_{\text{HIBOSC_SETTLE}}$ 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 steps are only necessary when the entire system is initialized for the first time. If the microcontroller has been in hibernation, then the Hibernation module has already been powered up and the above steps are not necessary. The software can detect that the Hibernation module and clock are already powered by examining the **CLK32EN** bit of the **HIBCTL** register.

Table 7-2 on page 207 illustrates how the clocks function with various bit setting both in normal operation and in hibernation.

Table 7-2. Hibernation Module Clock Operation

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

7.4.2 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 **RTCALTO** and **RTCALTI** 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.

7.4.3 RTC Match/Wake-Up from Hibernation

Use the following steps to implement the RTC match and wake-up functionality of the Hibernation module:

1. Write 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.

7.4.4 External Wake-Up from Hibernation

Use the following steps to implement the Hibernation module with the external $\overline{\text{WAKE}}$ pin as the wake-up source for the microcontroller:

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

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

7.4.5 RTC or External Wake-Up from Hibernation

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

7.4.6 Register Reset

The Hibernation module handles resets according to the following conditions:

- Cold Reset

When the hibernation module has no externally applied voltage and detects a change to either V_{DD} or V_{BAT} , it resets all hibernation module registers to the value in Table 7-3 on page 209.
- Reset During Hibernation Module Disable

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

While in Hibernation mode, or while transitioning from Hibernation mode to run mode (leaving the power cut), the reset generated by the POR circuitry of the microcontroller is suppressed, and the state of the Hibernation module's registers is unaffected.
- Reset While Hibernation Module is in Normal Mode

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

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

7.5 Register Map

Table 7-3 on page 209 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 166).

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 202.

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

Table 7-3. Hibernation Module Register Map

| Offset | Name | Type | Reset | Description | See page |
|-------------|----------|-------|-------------|-------------------------------------|----------|
| 0x000 | HIBRTCC | RO | 0x0000.0000 | Hibernation RTC Counter | 210 |
| 0x004 | HIBRTCM0 | R/W | 0xFFFF.FFFF | Hibernation RTC Match 0 | 211 |
| 0x008 | HIBRTCM1 | R/W | 0xFFFF.FFFF | Hibernation RTC Match 1 | 212 |
| 0x00C | HIBRTCLD | R/W | 0xFFFF.FFFF | Hibernation RTC Load | 213 |
| 0x010 | HIBCTL | R/W | 0x8000.0000 | Hibernation Control | 214 |
| 0x014 | HIBIM | R/W | 0x0000.0000 | Hibernation Interrupt Mask | 217 |
| 0x018 | HIBRIS | RO | 0x0000.0000 | Hibernation Raw Interrupt Status | 219 |
| 0x01C | HIBMIS | RO | 0x0000.0000 | Hibernation Masked Interrupt Status | 221 |
| 0x020 | HIBIC | R/W1C | 0x0000.0000 | Hibernation Interrupt Clear | 223 |
| 0x024 | HIBRTCT | R/W | 0x0000.7FFF | Hibernation RTC Trim | 224 |
| 0x030-0x12C | HIBDATA | R/W | - | Hibernation Data | 225 |

7.6 Register Descriptions

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

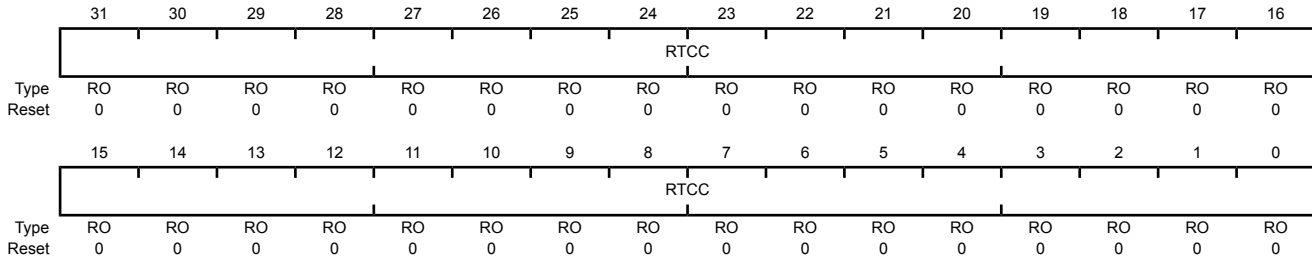
Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 202.

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.

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 202.

Hibernation RTC Match 0 (HIBRTCM0)

Base 0x400F.C000

Offset 0x004

Type R/W, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | RTCM0 | | | | | | | | | | | | | | | |
| 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 |
| | RTCM0 | | | | | | | | | | | | | | | |
| 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 | 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.

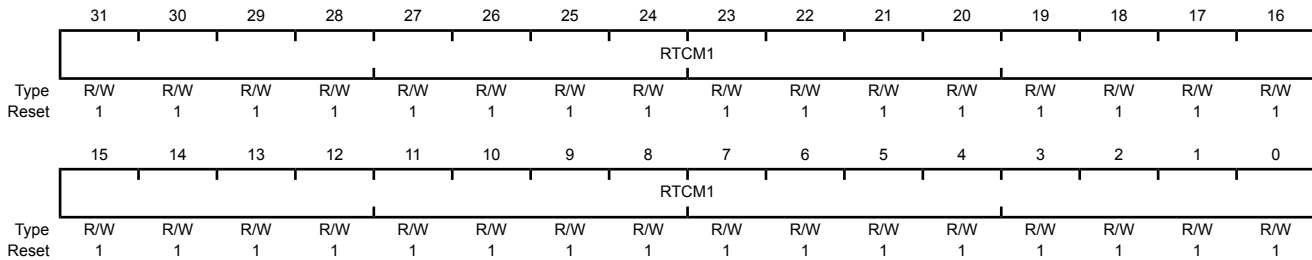
Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 202.

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 202.

Hibernation RTC Load (HIBRTCLD)

Base 0x400F.C000

Offset 0x00C

Type R/W, reset 0xFFFF.FFFF

| | | | | | | | | | | | | | | | | |
|-------|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | RTCLD | | | | | | | | | | | | | | | |
| 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 |
| | RTCLD | | | | | | | | | | | | | | | |
| 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 | 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----|----|----|--------|--------|---------|----------|--------|--------|--------|--------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | WRC | reserved | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | VDD3ON | VABORT | CLK32EN | LOWBATEN | PINWEN | RTCWEN | CLKSEL | HIBREQ | RTCEN |
| Type | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | WRC | RO | 1 | <p>Write Complete/Capable</p> <p>Value Description</p> <p>0 The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior.</p> <p>1 The interface is ready to accept a write.</p> <p>Software must poll this bit between write requests and defer writes until WRC=1 to ensure proper operation.</p> <p>This difference may be exploited by software at reset time to detect which method of programming is appropriate: 0 = software delay loops required; 1 = WRC paced available.</p> <p>The bit name WRC means "Write Complete," which is the normal use of the bit (between write accesses). However, because the bit is set out-of-reset, the name can also mean "Write Capable" which simply indicates that the interface may be written to by software.</p> |
| 30:9 | 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 | VDD3ON | R/W | 0 | <p>VDD Powered</p> <p>Value Description</p> <p>1 The internal switches control the power to the on-chip modules (VDD3ON mode).</p> <p>0 The internal switches are not used. The $\overline{\text{HIB}}$ signal should be used to control an external switch or regulator.</p> <p>Note that regardless of the status of the VDD3ON bit, the $\overline{\text{HIB}}$ signal is asserted during Hibernate mode. Thus, when VDD3ON is set, the $\overline{\text{HIB}}$ signal should not be connected to the 3.3V regulator, and the 3.3V power source should remain connected.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 7 | VABORT | R/W | 0 | Power Cut Abort Enable Value Description 1 Power cut is aborted. 0 A power cut occurs during a low-battery alert. |
| 6 | CLK32EN | R/W | 0 | Clocking Enable This bit must be enabled to use the Hibernation module. Value Description 1 The clock source to the Hibernation module is enabled. 0 The clock source to the Hibernation module is disabled. The CLKSEL bit is used to select between the 4.194304-MHz crystal source and the 32.768-kHz oscillator source. 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 1 Low battery voltage detection is enabled. If $V_{BAT} < V_{LOWBAT}$, the LOWBAT bit in the HIBRIS register is set. 0 Low battery monitoring is disabled. |
| 4 | PINWEN | R/W | 0 | External \overline{WAKE} Pin Enable Value Description 1 An assertion of the \overline{WAKE} pin takes the microcontroller out of hibernation. 0 The status of the \overline{WAKE} pin has no effect on hibernation. |
| 3 | RTCWEN | R/W | 0 | RTC Wake-up Enable Value Description 1 An RTC match event (the value the HIBRTCC register matches the value of the HIBRTCM0 or HIBRTCM1 register) takes the microcontroller out of hibernation. 0 An RTC match event has no effect on hibernation. |
| 2 | CLKSEL | R/W | 0 | Hibernation Module Clock Select Value Description 1 Use raw output. Use this value for a 32.768-kHz oscillator. 0 Use Divide-by-128 output. Use this value for a 4.194304-MHz crystal. |

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

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

This register is the interrupt mask register for the Hibernation module interrupt sources. Each bit in this register masks the corresponding bit in the **Hibernation Raw Interrupt Status (HIBRIS)** register. If a bit is unmasked, the interrupt is sent to the interrupt controller. If the bit is masked, the interrupt is not sent to the interrupt controller.

Hibernation Interrupt Mask (HIBIM)

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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|------|--------|--------|--------|-----|
| | 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 | | | | | | | | | | | | EXTW | LOWBAT | RTCAL1 | RTCAL0 | |
| Type | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | EXTW | R/W | 0 | External Wake-Up Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the EXTW bit in the HIBRIS register is set. 0 The EXTW interrupt is suppressed and not sent to the interrupt controller. |
| 2 | LOWBAT | R/W | 0 | Low Battery Voltage Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the LOWBAT bit in the HIBRIS register is set. 0 The LOWBAT interrupt is suppressed and not sent to the interrupt controller. |
| 1 | RTCAL1 | R/W | 0 | RTC Alert 1 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the RTCAL1 bit in the HIBRIS register is set. 0 The RTCAL1 interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 0 | RTCALTO | R/W | 0 | RTC Alert 0 Interrupt Mask |
| | | | | Value Description |
| | | | | 1 An interrupt is sent to the interrupt controller when the RTCALTO bit in the HIBRIS register is set. |
| | | | | 0 The RTCALTO interrupt is suppressed and not sent to the interrupt controller. |

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

This register is the raw interrupt status for the Hibernation module interrupt sources. Each bit can be masked by clearing the corresponding bit in the **HIBIM** register. When a bit is masked, the interrupt is not sent to the interrupt controller. Bits in this register are cleared by writing a 1 to the corresponding bit in the **Hibernation Interrupt Clear (HIBIC)** register.

Hibernation Raw Interrupt Status (HIBRIS)

Base 0x400F.C000

Offset 0x018

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|------|--------|--------|--------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | | | | EXTW | LOWBAT | RTCAL1 | RTCAL0 |
| 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 |
|-----------|----------|------|------------|--|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | EXTW | RO | 0 | External Wake-Up Raw Interrupt Status Value Description 1 The $\overline{\text{WAKE}}$ pin has been asserted. 0 The $\overline{\text{WAKE}}$ pin has not been asserted. This bit is cleared by writing a 1 to the EXTW bit in the HIBIC register. |
| 2 | LOWBAT | RO | 0 | Low Battery Voltage Raw Interrupt Status Value Description 1 The battery voltage dropped below V_{LOWBAT} . 0 The battery voltage has not dropped below V_{LOWBAT} . This bit is cleared by writing a 1 to the LOWBAT bit in the HIBIC register. |
| 1 | RTCAL1 | RO | 0 | RTC Alert 1 Raw Interrupt Status Value Description 1 The value of the HIBRTCC register matches the value in the HIBRTCM1 register. 0 No match This bit is cleared by writing a 1 to the RTCAL1 bit in the HIBIC register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 0 | RTCALTO | RO | 0 | RTC Alert 0 Raw Interrupt Status |
| | | | | Value Description |
| | | | | 1 The value of the HIBRTCC register matches the value in the HIBRTCM0 register. |
| | | | | 0 No match |
| | | | | This bit is cleared by writing a 1 to the RTCALTO bit in the HIBIC register. |

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

This register is the masked interrupt status for the Hibernation module interrupt sources. Bits in this register are the AND of the corresponding bits in the **HIBRIS** and **HIBIM** registers. When both corresponding bits are set, the bit in this register is set, and the interrupt is sent to the interrupt controller.

Hibernation Masked Interrupt Status (HIBMIS)

Base 0x400F.C000

Offset 0x01C

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|------|--------|---------|---------|
| | 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 | | | | | | | | | | | | EXTW | LOWBAT | RTCALT1 | RTCALT0 |
| 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 |
|-----------|----------|------|------------|--|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | EXTW | RO | 0 | External Wake-Up Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a $\overline{\text{WAKE}}$ pin assertion. 0 An external wake-up interrupt has not occurred. This bit is cleared by writing a 1 to the EXTW bit in the HIBIC register. |
| 2 | LOWBAT | RO | 0 | Low Battery Voltage Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a low battery voltage condition. 0 A low battery voltage interrupt has not occurred. This bit is cleared by writing a 1 to the LOWBAT bit in the HIBIC register. |
| 1 | RTCALT1 | RO | 0 | RTC Alert 1 Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a low battery voltage condition. 0 A low battery voltage interrupt has not occurred. When this bit is set, an RTC match 1 interrupt is sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 0 | RTCALTO | RO | 0 | RTC Alert 0 Masked Interrupt Status When this bit is set, an RTC match 0 interrupt is sent to the interrupt controller. |

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

This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources. Writing a 1 to a bit clears the corresponding interrupt in the **HIBRIS** register.

Hibernation Interrupt Clear (HIBIC)

Base 0x400F.C000

Offset 0x020

Type R/W1C, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-------|--------|--------|--------|
| | 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 | | | | | | | | | | | | EXTW | LOWBAT | RTCAL1 | RTCAL0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|--|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | EXTW | R/W1C | 0 | External Wake-Up Masked Interrupt Clear Writing a 1 to this bit clears the EXTW bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value. |
| 2 | LOWBAT | R/W1C | 0 | Low Battery Voltage Masked Interrupt Clear Writing a 1 to this bit clears the LOWBAT bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value. |
| 1 | RTCAL1 | R/W1C | 0 | RTC Alert1 Masked Interrupt Clear Writing a 1 to this bit clears the RTCAL1 bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value. |
| 0 | RTCAL0 | R/W1C | 0 | RTC Alert0 Masked Interrupt Clear Writing a 1 to this bit clears the RTCAL0 bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value. |

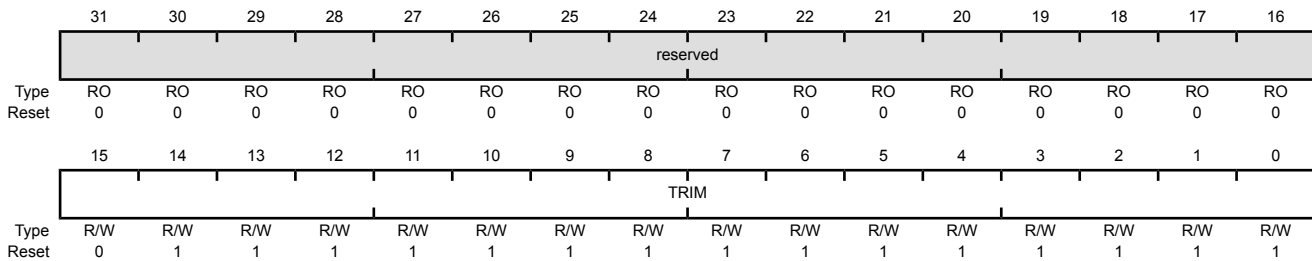
Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024

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 $0x7FFF \pm N$ clock cycles, where N is the number of clock cycles to add or subtract every 63 seconds.

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 202.

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. Compensation can be adjusted by software by moving the default value of 0x7FFF up or down. Moving the value up slows down the RTC and moving the value down speeds up the RTC. |

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

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 202.

Hibernation Data (HIBDATA)

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

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

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

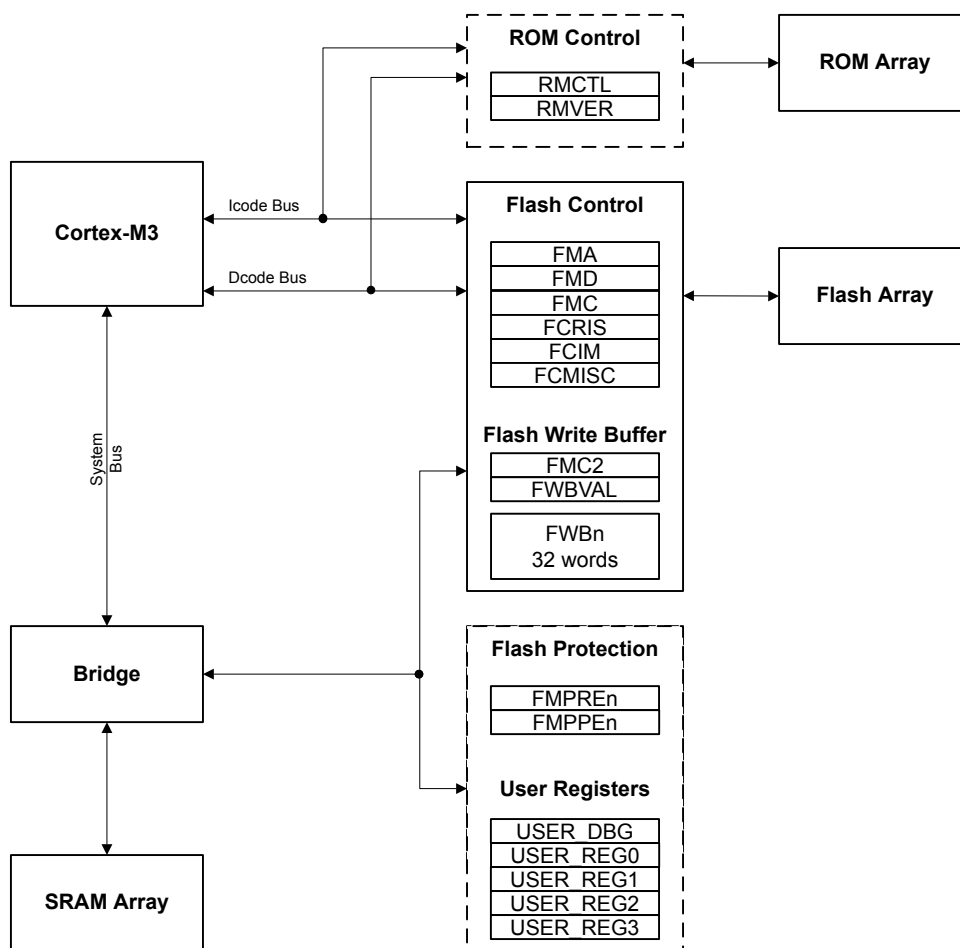
8 Internal Memory

The LM3S9B90 microcontroller comes with 96 KB of bit-banded SRAM, internal ROM, and 256 KB of Flash memory. The Flash memory controller provides a user-friendly interface, making Flash memory programming a simple task. Flash memory protection can be applied to the Flash memory on a 2-KB block basis.

8.1 Block Diagram

Figure 8-1 on page 226 illustrates the internal memory blocks and control logic. The dashed boxes in the figure indicate registers residing in the System Control module rather than the Internal Memory Control module.

Figure 8-1. Internal Memory Block Diagram



8.2 Functional Description

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

8.2.1 SRAM

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

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 base is located at address 0x2200.0000.

The bit-band alias is calculated by using the formula:

$$\text{bit-band alias} = \text{bit-band base} + (\text{byte offset} * 32) + (\text{bit number} * 4)$$

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, please refer to Chapter 4, “Memory Map” in the *ARM® Cortex™-M3 Technical Reference Manual*.

8.2.2 ROM

The internal ROM of the Stellaris® device is located at address 0x0100.0000 of the device memory map. The ROM contains the following components:

- Stellaris® Boot Loader and vector table (see “Boot Loader” on page 1061)
- Stellaris® Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces (see “ROM DriverLib Functions” on page 1066)
- Advanced Encryption Standard (AES) cryptography tables (see “Advance Encryption Standard and Cyclic Redundancy Check Software in ROM” on page 1088)
- Cyclic Redundancy Check (CRC) error detection functionality (see “Advance Encryption Standard and Cyclic Redundancy Check Software in ROM” on page 1088)

8.2.3 Flash Memory

The Flash memory is organized as a set of 1-KB blocks that can be individually erased. An individual 32-bit word can be programmed to change bits from 1 to 0. In addition, a write buffer provides the ability to concurrently program 32 continuous words in Flash memory. Erasing a block causes the entire contents of the block to be reset to all 1s. The 1-KB blocks are paired into sets of 2-KB blocks that can be individually protected. The protection allows blocks to be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

The Flash memory controller has a prefetch buffer that is automatically used when the CPU frequency is greater than 50 MHz. The prefetch buffer fetches two 32-bit words per clock allowing Flash memory to be read with no wait states while code is executing linearly. Branches incur a single wait state.

8.2.3.1 Flash Memory Protection

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

- **Flash Memory Protection Program Enable (FMPPEn)**: If a bit is set, the corresponding block may be programmed (written) or erased. If a bit is cleared, the corresponding block may not be changed.
- **Flash Memory Protection Read Enable (FMPREn)**: If a bit is set, the corresponding block may be executed or read by software or debuggers. If a bit is cleared, the corresponding block may only be executed, and contents of the memory block are prohibited from being accessed as data.

The policies may be combined as shown in Table 8-1 on page 228.

Table 8-1. Flash Memory Protection Policy Combinations

| FMPPEn | FMPREn | Protection |
|--------|--------|--|
| 0 | 0 | Execute-only protection. The block may only be executed and may not be written or erased. This mode is used to protect code. |
| 1 | 0 | The block may be written, erased or executed, but not read. This combination is unlikely to be used. |
| 0 | 1 | Read-only protection. The block may be read or executed but may not be written or erased. This mode is used to lock the block from further modification while allowing any read or execute access. |
| 1 | 1 | No protection. The block may be written, erased, executed or read. |

An access that attempts to program or erase a program-protected block is prohibited. An access that attempts to read a read-protected block is prohibited. Such accesses return data of all 0s. A controller interrupt may be optionally generated whenever an attempt is made to improperly access the Flash memory (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 230.

8.3 Flash Memory Initialization and Configuration

8.3.1 Flash Memory Programming

The Stellaris[®] devices provide a user-friendly interface for Flash memory programming. All erase/program operations are handled via three registers: **Flash Memory Address (FMA)**, **Flash Memory Data (FMD)**, and **Flash Memory Control (FMC)**. Note that if the debug capabilities of the microcontroller have been deactivated, resulting in a "locked" state, a recovery sequence must be

performed in order to reactivate the debug module. See “Recovering a “Locked” Microcontroller” on page 88.

Caution – The Flash memory is divided into sectors of electrically separated address ranges of 4 KB each, aligned on 4 KB boundaries. Erase/program operations on a 1-KB page have an electrical effect on the other three 1-KB pages within the sector. A specific 1-KB page must be erased after 6 total erase/program cycles occur to the other pages within it’s 4-KB sector. The following sequence of operations on a 4-KB sector of Flash memory (Page 0..3) provides an example:

- Page 3 is erase and programmed with values.
 - Page 0, Page 1, and Page 2 are erased and then programmed with values. At this point Page 3 has been affected by 3 erase/program cycles.
 - Page 0, Page 1, and Page 2 are again erased and then programmed with values. At this point Page 3 has been affected by 6 erase/program cycles.
 - If the contents of Page 3 must continue to be valid, Page 3 must be erased and reprogrammed before any other page in this sector has another erase or program operation.
-

8.3.1.1 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 memory write key and the `WRITE` bit (a value of 0xA442.0001) to the **FMC** register.
4. Poll the **FMC** register until the `WRITE` bit is cleared.

Important: To ensure proper operation, two writes to the same word must be separated by an `ERASE`. The following two sequences are allowed:

- `ERASE -> PROGRAM value -> PROGRAM 0x0000.0000`
- `ERASE -> PROGRAM value -> ERASE`

The following sequence is NOT allowed:

- `ERASE -> PROGRAM value -> PROGRAM value`
-

8.3.1.2 To perform an erase of a 1-KB page

1. Write the page address to the **FMA** register.
2. Write the Flash memory write key and the `ERASE` bit (a value of 0xA442.0002) to the **FMC** register.
3. Poll the **FMC** register until the `ERASE` bit is cleared.

8.3.1.3 To perform a mass erase of the Flash memory

1. Write the Flash memory write key and the `MERASE` bit (a value of `0xA442.0004`) to the **FMC** register.
2. Poll the **FMC** register until the `MERASE` bit is cleared.

8.3.2 32-Word Flash Memory Write Buffer

A 32-word write buffer provides the capability to perform faster write accesses to the Flash memory by concurrently programming 32 words with a single buffered Flash memory write operation. The buffered Flash memory write operation takes the same amount of time as the single word write operation controlled by bit 0 in the **FMC** register. The data for the buffered write is written to the **Flash Write Buffer (FWBn)** registers.

The registers are 32-word aligned with Flash memory, and therefore the register **FWB0** corresponds with the address in **FMA** where bits [6:0] of **FMA** are all 0. **FWB1** corresponds with the address in **FMA** + `0x4` and so on. Only the **FWBn** registers that have been updated since the previous buffered Flash memory write operation are written. The **Flash Write Buffer Valid (FWBVAL)** register shows which registers have been written since the last buffered Flash memory write operation. This register contains a bit for each of the 32 **FWBn** registers, where bit[n] of **FWBVAL** corresponds to **FWBn**. The **FWBn** register has been updated if the corresponding bit in the **FWBVAL** register is set.

8.3.2.1 To program 32 words with a single buffered Flash memory write operation

1. Write the source data to the **FWBn** registers.
2. Write the target address to the **FMA** register. This must be a 32-word aligned address (that is, bits [6:0] in **FMA** must be 0s).
3. Write the Flash memory write key and the `WRBUF` bit (a value of `0xA442.0001`) to the **FMC2** register.
4. Poll the **FMC2** register until the `WRBUF` bit is cleared.

8.3.3 Nonvolatile Register Programming

This section discusses how to update registers that are resident within the Flash memory itself. These registers exist in a separate space from the main Flash memory array and are not affected by an `ERASE` or `MASS ERASE` operation. The bits in these registers can be changed from 1 to 0 with a write operation. The register contents are unaffected by any reset condition except power-on reset, which returns the register contents to `0xFFFF.FFFF`. By committing the register values using the `COMT` bit in the **FMC** register, the register contents become nonvolatile and are therefore retained following power cycling. Once the register contents are committed, the only way to restore the factory default values is to perform the sequence described in “Recovering a “Locked” Microcontroller” on page 88.

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: These registers can only have bits changed from 1 to 0 by user programming and can only be committed once. After being committed, these registers can only be restored to their factory default values only by performing the sequence described in “Recovering

a "Locked" Microcontroller" on page 88. The mass erase of the main Flash memory array caused by the sequence is performed prior to restoring these registers.

In addition, the **USER_REG0**, **USER_REG1**, **USER_REG2**, **USER_REG3**, and **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. Table 8-2 on page 231 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 8-2. 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 |

8.4 Register Map

Table 8-3 on page 231 lists the ROM Controller register and the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, **FCMISC**, **FMC2**, **FWBVAL**, and **FWBn** register offsets are relative to the Flash memory control base address of 0x400F.D000. The ROM and Flash memory protection register offsets are relative to the System Control base address of 0x400F.E000.

Table 8-3. Flash Register Map

| Offset | Name | Type | Reset | Description | See page |
|--|--------|-------|-------------|--|----------|
| Flash Memory Registers (Flash Control Offset) | | | | | |
| 0x000 | FMA | R/W | 0x0000.0000 | Flash Memory Address | 233 |
| 0x004 | FMD | R/W | 0x0000.0000 | Flash Memory Data | 234 |
| 0x008 | FMC | R/W | 0x0000.0000 | Flash Memory Control | 235 |
| 0x00C | FCRIS | RO | 0x0000.0000 | Flash Controller Raw Interrupt Status | 237 |
| 0x010 | FCIM | R/W | 0x0000.0000 | Flash Controller Interrupt Mask | 238 |
| 0x014 | FCMISC | R/W1C | 0x0000.0000 | Flash Controller Masked Interrupt Status and Clear | 239 |
| 0x020 | FMC2 | R/W | 0x0000.0000 | Flash Memory Control 2 | 240 |

Table 8-3. Flash Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|---|-----------|-------|-------------|--|----------|
| 0x030 | FWBVAL | R/W | 0x0000.0000 | Flash Write Buffer Valid | 241 |
| 0x100 - 0x13C | FWBn | R/W | 0x0000.0000 | Flash Write Buffer n | 242 |
| Memory Registers (System Control Offset) | | | | | |
| 0x0F0 | RMCTL | R/W1C | - | ROM Control | 243 |
| 0x0F4 | RMVER | RO | 0x0202.5400 | ROM Version Register | 244 |
| 0x130 | FMPRE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 0 | 245 |
| 0x200 | FMPRE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 0 | 245 |
| 0x134 | FMPPE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 0 | 246 |
| 0x400 | FMPPE0 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 0 | 246 |
| 0x1D0 | USER_DBG | R/W | 0xFFFF.FFFE | User Debug | 247 |
| 0x1E0 | USER_REG0 | R/W | 0xFFFF.FFFF | User Register 0 | 248 |
| 0x1E4 | USER_REG1 | R/W | 0xFFFF.FFFF | User Register 1 | 249 |
| 0x1E8 | USER_REG2 | R/W | 0xFFFF.FFFF | User Register 2 | 250 |
| 0x1EC | USER_REG3 | R/W | 0xFFFF.FFFF | User Register 3 | 251 |
| 0x204 | FMPRE1 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 1 | 252 |
| 0x208 | FMPRE2 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 2 | 253 |
| 0x20C | FMPRE3 | R/W | 0xFFFF.FFFF | Flash Memory Protection Read Enable 3 | 254 |
| 0x404 | FMPPE1 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 1 | 255 |
| 0x408 | FMPPE2 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 2 | 256 |
| 0x40C | FMPPE3 | R/W | 0xFFFF.FFFF | Flash Memory Protection Program Enable 3 | 257 |

8.5 Flash Memory Register Descriptions (Flash Control Offset)

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

Register 1: Flash Memory Address (FMA), offset 0x000

During a write operation, this register contains a 4-byte-aligned address and specifies where the data is written. During erase operations, this register contains a 1 KB-aligned 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

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | OFFSET | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | OFFSET | | | | | | | | | | | | | | | |
| 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 |
|-----------|----------|------|-------|--|
| 31:18 | 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 memory where operation is performed, except for nonvolatile registers (see "Nonvolatile Register Programming" on page 230 for details on values for this field). |

Register 2: Flash Memory Data (FMD), offset 0x004

This register contains the data to be written during the programming cycle or read during the read cycle. Note that the contents of this register are undefined for a read access of an execute-only block. This register is not used during erase cycles.

Flash Memory Data (FMD)

Base 0x400F.D000

Offset 0x004

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | DATA | | | | | | | | | | | | | | | |
| 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 |
| | DATA | | | | | | | | | | | | | | | |
| 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 | DATA | R/W | 0x0000.0000 | Data Value Data value for write operation. |

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

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 233). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 234) is written to the specified address.

This register must be the final register written and initiates the memory operation. The four control bits in the lower byte of this register are used to initiate memory operations.

Care must be taken not to set multiple control bits as the results of such an operation are unpredictable.

Flash Memory Control (FMC)

Base 0x400F.D000
Offset 0x008
Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|------|--------|-------|-------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | WRKEY | | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | | | COMT | MERASE | ERASE | WRITE | |
| Type | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|---|------|--------|--|---|---|---|---|
| 31:16 | WRKEY | WO | 0x0000 | Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a Flash memory write to occur. Writes to the FMC register without this <code>WRKEY</code> value are ignored. A read of this field returns the value 0. | | | | |
| 15:4 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | |
| 3 | COMT | R/W | 0 | Commit Register Value This bit is used to commit writes to Flash-memory-resident registers and to monitor the progress of that process. Value Description <table border="0"> <tr> <td>1</td> <td>Set this bit to commit (write) the register value to a Flash-memory-resident register. When read, a 1 indicates that the previous commit access is not complete.</td> </tr> <tr> <td>0</td> <td>A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous commit access is complete.</td> </tr> </table> A commit can take up to 50 μ s. See "Nonvolatile Register Programming" on page 230 for more information on programming Flash-memory-resident registers. | 1 | Set this bit to commit (write) the register value to a Flash-memory-resident register. When read, a 1 indicates that the previous commit access is not complete. | 0 | A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous commit access is complete. |
| 1 | Set this bit to commit (write) the register value to a Flash-memory-resident register. When read, a 1 indicates that the previous commit access is not complete. | | | | | | | |
| 0 | A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous commit access is complete. | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 2 | MERASE | R/W | 0 | <p>Mass Erase Flash Memory</p> <p>This bit is used to mass erase the Flash main memory and to monitor the progress of that process.</p> <p>Value Description</p> <p>1 Set this bit to erase the Flash main memory.</p> <p>When read, a 1 indicates that the previous mass erase access is not complete.</p> <p>0 A write of 0 has no effect on the state of this bit.</p> <p>When read, a 0 indicates that the previous mass erase access is complete.</p> <p>A mass erase can take up to 250 ms.</p> |
| 1 | ERASE | R/W | 0 | <p>Erase a Page of Flash Memory</p> <p>This bit is used to erase a page of Flash memory and to monitor the progress of that process.</p> <p>Value Description</p> <p>1 Set this bit to erase the Flash memory page specified by the contents of the FMA register.</p> <p>When read, a 1 indicates that the previous page erase access is not complete.</p> <p>0 A write of 0 has no effect on the state of this bit.</p> <p>When read, a 0 indicates that the previous page erase access is complete.</p> <p>A page erase can take up to 25 ms.</p> |
| 0 | WRITE | R/W | 0 | <p>Write a Word into Flash Memory</p> <p>This bit is used to write a word into Flash memory and to monitor the progress of that process.</p> <p>Value Description</p> <p>1 Set this bit to write the data stored in the FMD register into the Flash memory location specified by the contents of the FMA register.</p> <p>When read, a 1 indicates that the write update access is not complete.</p> <p>0 A write of 0 has no effect on the state of this bit.</p> <p>When read, a 0 indicates that the previous write update access is complete.</p> <p>Writing a single word can take up to 50 μs.</p> |

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

This register indicates that the Flash memory controller has an interrupt condition. An interrupt is sent to the interrupt controller only if the corresponding **FCIM** register bit is set.

Flash Controller Raw Interrupt Status (FCRIS)

Base 0x400F.D000

Offset 0x00C

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------|------|
| | 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 | | | | | | | | | | | | | | | PRIS | ARIS |
| Type | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|--|------|------------|--|---|--|---|--|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | |
| 1 | PRIS | RO | 0 | <p>Programming Raw Interrupt Status</p> <p>This bit provides status on programming cycles which are write or erase actions generated through the FMC or FMC2 register bits (see page 235 and page 240).</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>The programming cycle has completed.</td> </tr> <tr> <td>0</td> <td>The programming cycle has not completed.</td> </tr> </table> <p>This status is sent to the interrupt controller when the PMASK bit in the FCIM register is set.</p> <p>This bit is cleared by writing a 1 to the PMISC bit in the FCMISC register.</p> | 1 | The programming cycle has completed. | 0 | The programming cycle has not completed. |
| 1 | The programming cycle has completed. | | | | | | | |
| 0 | The programming cycle has not completed. | | | | | | | |
| 0 | ARIS | RO | 0 | <p>Access Raw Interrupt Status</p> <p>This bit indicates if the Flash memory was improperly accessed.</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>The program tried to access the Flash memory counter to the policy set in the FMPREn and FMPPEn registers.</td> </tr> <tr> <td>0</td> <td>No access has tried to improperly access the Flash memory.</td> </tr> </table> <p>This status is sent to the interrupt controller when the AMASK bit in the FCIM register is set.</p> <p>This bit is cleared by writing a 1 to the AMISC bit in the FCMISC register.</p> | 1 | The program tried to access the Flash memory counter to the policy set in the FMPREn and FMPPEn registers. | 0 | No access has tried to improperly access the Flash memory. |
| 1 | The program tried to access the Flash memory counter to the policy set in the FMPREn and FMPPEn registers. | | | | | | | |
| 0 | No access has tried to improperly access the Flash memory. | | | | | | | |

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

This register controls whether the Flash memory controller generates interrupts to the controller.

Flash Controller Interrupt Mask (FCIM)

Base 0x400F.D000

Offset 0x010

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|-----|
| | 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 | | | | | | | | | | | | | | PMASK | AMASK | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|---|------|------------|--|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | |
| 1 | PMASK | R/W | 0 | <p>Programming Interrupt Mask</p> <p>This bit controls the reporting of the programming raw interrupt status to the interrupt controller.</p> <p>Value Description</p> <table border="1"> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the <code>PRIS</code> bit is set.</td> </tr> <tr> <td>0</td> <td>The <code>PRIS</code> interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table> | 1 | An interrupt is sent to the interrupt controller when the <code>PRIS</code> bit is set. | 0 | The <code>PRIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 1 | An interrupt is sent to the interrupt controller when the <code>PRIS</code> bit is set. | | | | | | | |
| 0 | The <code>PRIS</code> interrupt is suppressed and not sent to the interrupt controller. | | | | | | | |
| 0 | AMASK | R/W | 0 | <p>Access Interrupt Mask</p> <p>This bit controls the reporting of the access raw interrupt status to the interrupt controller.</p> <p>Value Description</p> <table border="1"> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the <code>ARIS</code> bit is set.</td> </tr> <tr> <td>0</td> <td>The <code>ARIS</code> interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table> | 1 | An interrupt is sent to the interrupt controller when the <code>ARIS</code> bit is set. | 0 | The <code>ARIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 1 | An interrupt is sent to the interrupt controller when the <code>ARIS</code> bit is set. | | | | | | | |
| 0 | The <code>ARIS</code> 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 | RO | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | PMISC | R/W1C | 0 | <p>Programming Masked Interrupt Status and Clear</p> <p>Value Description</p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed.</p> <p>Writing a 1 to this bit clears PMISC and also the PRIS bit in the FCRIS register (see page 237).</p> <p>0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred.</p> <p>A write of 0 has no effect on the state of this bit.</p> |
| 0 | AMISC | R/W1C | 0 | <p>Access Masked Interrupt Status and Clear</p> <p>Value Description</p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because an improper access to protected Flash memory was attempted.</p> <p>Writing a 1 to this bit clears AMISC and also the ARIS bit in the FCRIS register (see page 237).</p> <p>0 When read, a 0 indicates that no improper accesses have occurred.</p> <p>A write of 0 has no effect on the state of this bit.</p> |

Register 7: Flash Memory Control 2 (FMC2), offset 0x020

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 233). If the access is a write access, the data contained in the **Flash Write Buffer (FWB)** registers is written.

This register must be the final register written as it initiates the memory operation.

Flash Memory Control 2 (FMC2)

Base 0x400F.D000

Offset 0x020

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | WRKEY | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | WRBUF |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|--|
| 31:16 | WRKEY | WO | 0x0000 | Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a write to occur. Writes to the FMC2 register without this WRKEY value are ignored. A read of this field returns the value 0. |
| 15:1 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | WRBUF | R/W | 0 | Buffered Flash Memory Write |

This bit is used to start a buffered write to Flash memory.

Value Description

1 Set this bit to write the data stored in the **FWBn** registers to the location specified by the contents of the **FMA** register.

When read, a 1 indicates that the previous buffered Flash memory write access is not complete.

0 A write of 0 has no effect on the state of this bit.

When read, a 0 indicates that the previous buffered Flash memory write access is complete.

A buffered Flash memory write can take up to 4 ms.

Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030

This register provides a bitwise status of which **FWB_n** registers have been written by the processor since the last write of the Flash memory write buffer. The entries with a 1 are written on the next write of the Flash memory write buffer. This register is cleared after the write operation by hardware. A protection violation on the write operation also clears this status.

Software can program the same 32 words to various Flash memory locations by setting the **FWB_[n]** bits after they are cleared by the write operation. The next write operation then uses the same data as the previous one. In addition, if a **FWB_n** register change should not be written to Flash memory, software can clear the corresponding **FWB_[n]** bit to preserve the existing data when the next write operation occurs.

Flash Write Buffer Valid (FWBVAL)

Base 0x400F.D000

Offset 0x030

Type R/W, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|--------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | FWB[n] | | | | | | | | | | | | | | | |
| 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 |
| | FWB[n] | | | | | | | | | | | | | | | |
| 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 | FWB[n] | R/W | 0x0 | Flash Memory Write Buffer |

Value Description

- | | |
|---|--|
| 1 | The corresponding FWB_n register has been updated since the last buffer write operation and is ready to be written to Flash memory. |
| 0 | The corresponding FWB_n register has no new data to be written. |

Bit 0 corresponds to **FWB0**, offset 0x100, and bit 31 corresponds to **FWB31**, offset 0x13C.

Register 9: Flash Write Buffer n (FWBn), offset 0x100 - 0x13C

These 32 registers hold the contents of the data to be written into the Flash memory on a buffered Flash memory write operation. The offset selects one of the 32-bit registers. Only **FWBn** registers that have been updated since the preceding buffered Flash memory write operation are written into the Flash memory, so it is not necessary to write the entire bank of registers in order to write 1 or 2 words. The **FWBn** registers are written into the Flash memory with the **FWB0** register corresponding to the address contained in **FMA**. **FWB1** is written to the address **FMA+0x4** etc. Note that only data bits that are 0 result in the Flash memory being modified. A data bit that is 1 leaves the content of the Flash memory bit at its previous value.

Flash Write Buffer n (FWBn)

Base 0x400F.D000
Offset 0x100 - 0x13C
Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | DATA | | | | | | | | | | | | | | | |
| 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 |
| | DATA | | | | | | | | | | | | | | | |
| 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 | DATA | R/W | 0x0000.0000 | Data |

Data to be written into the Flash memory.

8.6 Memory Register Descriptions (System Control Offset)

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

Register 10: ROM Control (RMCTL), offset 0x0F0

This register provides control of the ROM controller state. This register offset is relative to the System Control base address of 0x400F.E000.

ROM Control (RMCTL)

Base 0x400F.E000

Offset 0x0F0

Type R/W1C, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | BA |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | BA | R/W1C | - | Boot Alias |

Upon reset, the system control module checks the first two words of the Flash memory to see if it has been programmed. If the first two words of Flash memory contain 0xFFFF.FFFF then it has not yet been programmed, and this bit is then set by hardware so that the on-chip ROM appears at address 0x0.

Value Description

- | | |
|---|---|
| 1 | The microcontroller's ROM appears at address 0x0. This bit is set automatically if the first two words of the Flash memory contain 0xFFFF.FFFF. |
| 0 | The Flash memory is at address 0x0. |

This bit is cleared by writing a 1 to this bit position.

Register 11: ROM Version Register (RMVER), offset 0x0F4**Note:** Offset is relative to System Control base address of 0x400FE000.

A 32-bit read-only register containing the ROM content version information.

ROM Version Register (RMVER)

Base 0x400F.E000

Offset 0x0F4

Type RO, reset 0x0202.5400

| | | | | | | | | | | | | | | | | |
|-------|------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | CONT | | | | | | | | SIZE | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | VER | | | | | | | | REV | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 31:24 | CONT | RO | 0x02 | ROM Contents Value Description 0x02 Stellaris Boot Loader & DriverLib with AES |
| 23:16 | SIZE | RO | 0x02 | ROM Size This field encodes the size of the ROM. Value Description 0x02 Stellaris Boot Loader & DriverLib with AES, ethernet |
| 15:8 | VER | RO | 0x54 | ROM Version |
| 7:0 | REV | RO | 0x0 | ROM Revision |

Register 12: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

Note: This register is aliased for backwards compatibility.

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 (**FMPPE_n** 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 **FMPPE_n** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 0 (FMPRE0)

Base 0x400F.E000
Offset 0x130 and 0x200
Type R/W, reset 0xFFFF.FFFF

| | | | | | | | | | | | | | | | | |
|-------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". 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 13: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

Note: This register is aliased for backwards compatibility.

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPPE_n** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPE_n** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPPE_n** and **FMPPE_n** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 0 (FMPPE0)

Base 0x400F.E000
Offset 0x134 and 0x400
Type R/W, reset 0xFFFF.FFFF

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

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

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 14: 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. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter.

User Debug (USER_DBG)

Base 0x400F.E000

Offset 0x1D0

Type R/W, reset 0xFFFF.FFFE

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|--|--|--|--|--|--|--|--|--|--|--|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | | | | | | | | | | |
| | NW | DATA | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | R/W | | | | | | | | | | | |
| Reset | 1 | 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 | | | | | | | | | | | | |
| | DATA | | | | | | | | | | | | | | | DBG1 | DBG0 | | | | | | | | | | | |
| 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 | R/W | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | | | | | | | | | | | |

| 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 15: User Register 0 (USER_REG0), offset 0x1E0

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

This register provides 31 bits of user-defined data that is non-volatile and can only be committed once. Bit 31 indicates that the register is available to be committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG section.

User Register 0 (USER_REG0)

Base 0x400F.E000

Offset 0x1E0

Type R/W, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | NW | DATA | | | | | | | | | | | | | | |
| 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 |
| | DATA | | | | | | | | | | | | | | | |
| 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 | 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 16: User Register 1 (USER_REG1), offset 0x1E4

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

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

User Register 1 (USER_REG1)

Base 0x400F.E000

Offset 0x1E4

Type R/W, reset 0xFFFF.FFFF

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

| 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 17: User Register 2 (USER_REG2), offset 0x1E8

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

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

User Register 2 (USER_REG2)

Base 0x400F.E000
 Offset 0x1E8
 Type R/W, reset 0xFFFF.FFFF

| | | | | | | | | | | | | | | | | |
|-------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | NW | DATA | | | | | | | | | | | | | | |
| 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 |
| | DATA | | | | | | | | | | | | | | | |
| 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 | 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 18: User Register 3 (USER_REG3), offset 0x1EC

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

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

User Register 3 (USER_REG3)

Base 0x400F.E000

Offset 0x1EC

Type R/W, reset 0xFFFF.FFFF

| | | | | | | | | | | | | | | | | |
|-------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | NW | DATA | | | | | | | | | | | | | | |
| 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 |
| | DATA | | | | | | | | | | | | | | | |
| 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 | 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 19: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

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

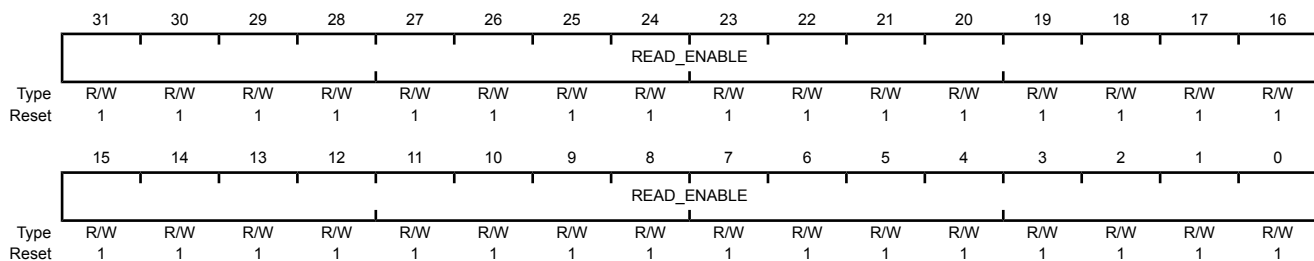
This register stores the read-only protection bits for each 2-KB flash block (**FMPPE_n** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPRE_n** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPRE_n** and **FMPPE_n** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 1 (FMPRE1)

Base 0x400F.E000

Offset 0x204

Type R/W, reset 0xFFFFFFFF



| Bit/Field | Name | Type | Reset | Description |
|-----------|-------------|------|------------|-------------------|
| 31:0 | READ_ENABLE | R/W | 0xFFFFFFFF | Flash Read Enable |

Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

| 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 20: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPE_n** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPRE_n** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPRE_n** and **FMPPE_n** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 2 (FMPRE2)

Base 0x400F.E000

Offset 0x208

Type R/W, reset 0xFFFFFFFF

| | | | | | | | | | | | | | | | | |
|-------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations". Value Description 0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB. |

Register 21: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 3 (FMPRE3)

Base 0x400F.E000

Offset 0x20C

Type R/W, reset 0xFFFFFFFF

| | | | | | | | | | | | | | | | | |
|-------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |

Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

| Value | Description |
|-------|-------------|
|-------|-------------|

| | |
|------------|---|
| 0xFFFFFFFF | Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB. |
|------------|---|

Register 22: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREN** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 1 (FMPPE1)

Base 0x400F.E000

Offset 0x404

Type R/W, reset 0xFFFF.FFFF

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

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

| 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 23: 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 (**FMPPE_n** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPE_n** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPPE_n** and **FMPPE_n** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 2 (FMPPE2)

Base 0x400F.E000

Offset 0x408

Type R/W, reset 0xFFFF.FFFF

| | | | | | | | | | | | | | | | | |
|-------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |

| Value | Description |
|------------|---|
| 0xFFFFFFFF | Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB. |

Register 24: 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 (**FMPPE_n** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPE_n** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPPE_n** and **FMPPE_n** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 3 (FMPPE3)

Base 0x400F.E000

Offset 0x40C

Type R/W, reset 0xFFFF.FFFF

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

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

| Value | Description |
|------------|---|
| 0xFFFFFFFF | Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB. |

9 Micro Direct Memory Access (μDMA)

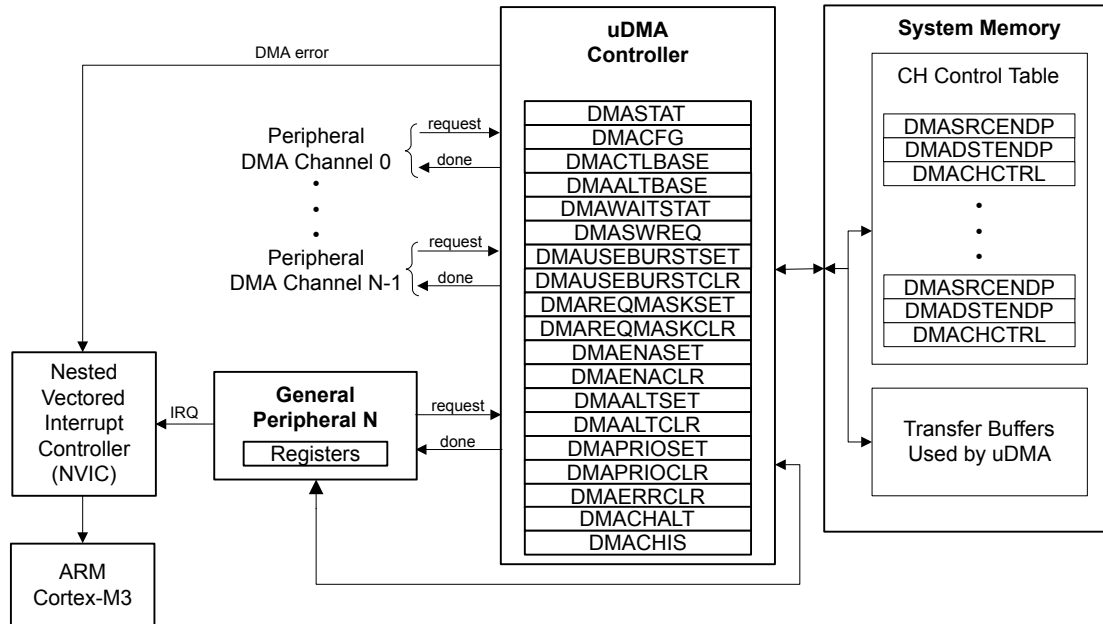
The LM3S9B90 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μDMA). The μDMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the available bus bandwidth. The μDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The μDMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable μDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
 - Basic for simple transfer scenarios
 - Ping-pong for continuous data flow
 - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
 - Independently configured and operated channels
 - Dedicated channels for supported on-chip modules: GP Timer, USB, UART, Ethernet, ADC, EPI, SSI, I²S
 - Alternate channel assignments
 - One channel each for receive and transmit path for bidirectional modules
 - Dedicated channel for software-initiated transfers
 - Per-channel configurable bus arbitration scheme
 - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between μDMA controller and the processor core
 - μDMA controller access is subordinate to core access
 - RAM striping
 - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment

- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

9.1 Block Diagram

Figure 9-1. μ DMA Block Diagram



9.2 Functional Description

The μ DMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the microcontroller's Cortex-M3 processor core. It supports multiple data sizes and address increment schemes, multiple levels of priority among DMA channels, and several transfer modes to allow for sophisticated programmed data transfers. The μ DMA controller's usage of the bus is always subordinate to the processor core, so it never holds up a bus transaction by the processor. Because the μ DMA controller is only using otherwise-idle bus cycles, the data transfer bandwidth it provides is essentially free, with no impact on the rest of the system. The bus architecture has been optimized to greatly enhance the ability of the processor core and the μ DMA controller to efficiently share the on-chip bus, thus improving performance. The optimizations include RAM striping and peripheral bus segmentation, which in many cases allow both the processor core and the μ DMA controller to access the bus and perform simultaneous data transfers.

Each peripheral function that is supported has a dedicated channel on the μ DMA controller that can be configured independently. The μ DMA controller implements a unique configuration method using channel control structures that are maintained in system memory by the processor. While simple transfer modes are supported, it is also possible to build up sophisticated "task" lists in memory that allow the μ DMA controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The μ DMA controller also supports the use of ping-pong buffering to accommodate constant streaming of data to or from a peripheral.

Each channel also has a configurable arbitration size. The arbitration size is the number of items that are transferred in a burst before the μ DMA controller re-arbitrates for channel priority. Using the

arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time it makes a μ DMA service request.

9.2.1 Channel Assignments

μ DMA channels 0-31 are assigned to peripherals according to the following table. The **DMA Channel Alternate Select (DMACHALT)** register (see page 307) can be used to specify the alternate assignment. Most channels have primary and alternate assignments. If the primary function is not available on this microcontroller, the alternate function becomes the primary function. If the alternate function is not available, the primary function is the only option.

Note: Channels noted in the table as "Available for software" may be assigned to peripherals in the future. However, they are currently available for software use. Channel 30 is dedicated for software use.

The USB endpoints mapped to μ DMA channels 0-3 can be changed with the **USBDMASEL** register (see page 993).

Table 9-1. μ DMA Channel Assignments

| μ DMA Channel | Peripheral Assigned | Alternate Assignment |
|-------------------|--------------------------|--------------------------|
| 0 | USB Endpoint 1 Receive | UART2 Receive |
| 1 | USB Endpoint 1 Transmit | UART2 Transmit |
| 2 | USB Endpoint 2 Receive | General-Purpose Timer 3A |
| 3 | USB Endpoint 2 Transmit | General-Purpose Timer 3B |
| 4 | USB Endpoint 3 Receive | General-Purpose Timer 2A |
| 5 | USB Endpoint 3 Transmit | General-Purpose Timer 2B |
| 6 | Ethernet Receive | General-Purpose Timer 2A |
| 7 | Ethernet Transmit | General-Purpose Timer 2B |
| 8 | UART0 Receive | SSI1 Receive |
| 9 | UART0 Transmit | SSI1 Transmit |
| 10 | SSI0 Receive | UART1 Receive |
| 11 | SSI0 Transmit | UART1 Transmit |
| 12 | Available for software | UART2 Receive |
| 13 | Available for software | UART2 Transmit |
| 14 | ADC0 Sample Sequencer 0 | General-Purpose Timer 2A |
| 15 | ADC0 Sample Sequencer 1 | General-Purpose Timer 2B |
| 16 | ADC0 Sample Sequencer 2 | Available for software |
| 17 | ADC0 Sample Sequencer 3 | Available for software |
| 18 | General-Purpose Timer 0A | General-Purpose Timer 1A |
| 19 | General-Purpose Timer 0B | General-Purpose Timer 1B |
| 20 | General-Purpose Timer 1A | EPI0 Receive |
| 21 | General-Purpose Timer 1B | EPI0 Transmit |
| 22 | UART1 Receive | Available for software |
| 23 | UART1 Transmit | Available for software |
| 24 | SSI1 Receive | ADC1 Sample Sequencer 0 |
| 25 | SSI1 Transmit | ADC1 Sample Sequencer 1 |
| 26 | Available for software | ADC1 Sample Sequencer 2 |
| 27 | Available for software | ADC1 Sample Sequencer 3 |

Table 9-1. μ DMA Channel Assignments (continued)

| μ DMA Channel | Peripheral Assigned | Alternate Assignment |
|-------------------|----------------------------|------------------------|
| 28 | I ² S0 Receive | Available for software |
| 29 | I ² S0 Transmit | Available for software |
| 30 | Dedicated for software use | |
| 31 | Reserved | |

9.2.2 Priority

The μ DMA controller assigns priority to each channel based on the channel number and the priority level bit for the channel. Channel number 0 has the highest priority and as the channel number increases, the priority of a channel decreases. Each channel has a priority level bit to provide two levels of priority: default priority and high priority. If the priority level bit is set, then that channel has higher priority than all other channels at default priority. If multiple channels are set for high priority, then the channel number is used to determine relative priority among all the high priority channels.

The priority bit for a channel can be set using the **DMA Channel Priority Set (DMAPRIOSET)** register and cleared with the **DMA Channel Priority Clear (DMAPRIOCLR)** register.

9.2.3 Arbitration Size

When a μ DMA channel requests a transfer, the μ DMA controller arbitrates among all the channels making a request and services the μ DMA channel with the highest priority. Once a transfer begins, it continues for a selectable number of transfers before rearbitrating among the requesting channels again. The arbitration size can be configured for each channel, ranging from 1 to 1024 item transfers. After the μ DMA controller transfers the number of items specified by the arbitration size, it then checks among all the channels making a request and services the channel with the highest priority.

If a lower priority μ DMA channel uses a large arbitration size, the latency for higher priority channels is increased because the μ DMA controller completes the lower priority burst before checking for higher priority requests. Therefore, lower priority channels should not use a large arbitration size for best response on high priority channels.

The arbitration size can also be thought of as a burst size. It is the maximum number of items that are transferred at any one time in a burst. Here, the term arbitration refers to determination of μ DMA channel priority, not arbitration for the bus. When the μ DMA controller arbitrates for the bus, the processor always takes priority. Furthermore, the μ DMA controller is held off whenever the processor must perform a bus transaction on the same bus, even in the middle of a burst transfer.

9.2.4 Request Types

The μ DMA controller responds to two types of requests from a peripheral: single or burst. Each peripheral may support either or both types of requests. A single request means that the peripheral is ready to transfer one item, while a burst request means that the peripheral is ready to transfer multiple items.

The μ DMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted, and the μ DMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 9-2, which shows how each peripheral supports the two request types.

Table 9-2. Request Type Support

| Peripheral | Single Request Signal | Burst Request Signal |
|------------|-----------------------|----------------------|
| USB TX | None | FIFO TXRDY |

Table 9-2. Request Type Support (continued)

| Peripheral | Single Request Signal | Burst Request Signal |
|-----------------------|-----------------------|-------------------------------|
| USB RX | None | FIFO RXRDY |
| Ethernet TX | TX FIFO empty | None |
| Ethernet RX | RX packet received | None |
| UART TX | TX FIFO Not Full | TX FIFO Level (configurable) |
| UART RX | RX FIFO Not Empty | RX FIFO Level (configurable) |
| SSI TX | TX FIFO Not Full | TX FIFO Level (fixed at 4) |
| SSI RX | RX FIFO Not Empty | RX FIFO Level (fixed at 4) |
| ADC | None | Sequencer \overline{IE} bit |
| General-Purpose Timer | None | Raw interrupt pulse |
| I ² S TX | None | FIFO service request |
| I ² S RX | None | FIFO service request |
| EPI TX | None | TX FIFO not full |
| EPI RX | None | RX FIFO not empty |

9.2.4.1 Single Request

When a single request is detected, and not a burst request, the μ DMA controller transfers one item and then stops to wait for another request.

9.2.4.2 Burst Request

When a burst request is detected, the μ DMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART generates a burst request based on the FIFO trigger level. In this case, the arbitration size should be set to the amount of data that the FIFO can transfer when the trigger level is reached. A burst transfer runs to completion once it is started, and cannot be interrupted, even by a higher priority channel. Burst transfers complete in a shorter time than the same number of non-burst transfers.

It may be desirable to use only burst transfers and not allow single transfers. For example, perhaps the nature of the data is such that it only makes sense when transferred together as a single unit rather than one piece at a time. The single request can be disabled by using the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register. By setting the bit for a channel in this register, the μ DMA controller only responds to burst requests for that channel.

9.2.5 Channel Configuration

The μ DMA controller uses an area of system memory to store a set of channel control structures in a table. The control table may have one or two entries for each μ DMA channel. Each entry in the table structure contains source and destination pointers, transfer size, and transfer mode. The control table can be located anywhere in system memory, but it must be contiguous and aligned on a 1024-byte boundary.

Table 9-3 on page 263 shows the layout in memory of the channel control table. Each channel may have one or two control structures in the control table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table, and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only

the first half of the table must be allocated in memory; the second half of the control table is not necessary, and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

Any unused memory in the control table may be used by the application. This includes the control structures for any channels that are unused by the application as well as the unused control word for each channel.

Table 9-3. Control Structure Memory Map

| Offset | Channel |
|--------|---------------|
| 0x0 | 0, Primary |
| 0x10 | 1, Primary |
| ... | ... |
| 0x1F0 | 31, Primary |
| 0x200 | 0, Alternate |
| 0x210 | 1, Alternate |
| ... | ... |
| 0x3F0 | 31, Alternate |

Table 9-4 shows an individual control structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four long words: the source end pointer, the destination end pointer, the control word, and an unused entry. The end pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

Table 9-4. Channel Control Structure

| Offset | Description |
|--------|-------------------------|
| 0x000 | Source End Pointer |
| 0x004 | Destination End Pointer |
| 0x008 | Control Word |
| 0x00C | Unused |

The control word contains the following fields:

- Source and destination data sizes
- Source and destination address increment size
- Number of transfers before bus arbitration
- Total number of items to transfer
- Useburst flag
- Transfer mode

The control word and each field are described in detail in “ μ DMA Channel Control Structure” on page 281. The μ DMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode indicates "stopped." Because the control word is modified by the μ DMA controller, it must be

reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

Prior to starting a transfer, a μ DMA channel must be enabled by setting the appropriate bit in the **DMA Channel Enable Set (DMAENASET)** register. A channel can be disabled by setting the channel bit in the **DMA Channel Enable Clear (DMAENACLR)** register. At the end of a complete μ DMA transfer, the controller automatically disables the channel.

9.2.6 Transfer Modes

The μ DMA controller supports several transfer modes. Two of the modes support simple one-time transfers. Several complex modes support a continuous flow of data.

9.2.6.1 Stop Mode

While Stop is not actually a transfer mode, it is a valid value for the mode field of the control word. When the mode field has this value, the μ DMA controller does not perform any transfers and disables the channel if it is enabled. At the end of a transfer, the μ DMA controller updates the control word to set the mode to Stop.

9.2.6.2 Basic Mode

In Basic mode, the μ DMA controller performs transfers as long as there are more items to transfer, and a transfer request is present. This mode is used with peripherals that assert a μ DMA request signal whenever the peripheral is ready for a data transfer. Basic mode should not be used in any situation where the request is momentary even though the entire transfer should be completed. For example, a software-initiated transfer creates a momentary request, and in Basic mode, only one item is transferred on a software request.

When all of the items have been transferred using Basic mode, the μ DMA controller sets the mode for that channel to Stop.

9.2.6.3 Auto Mode

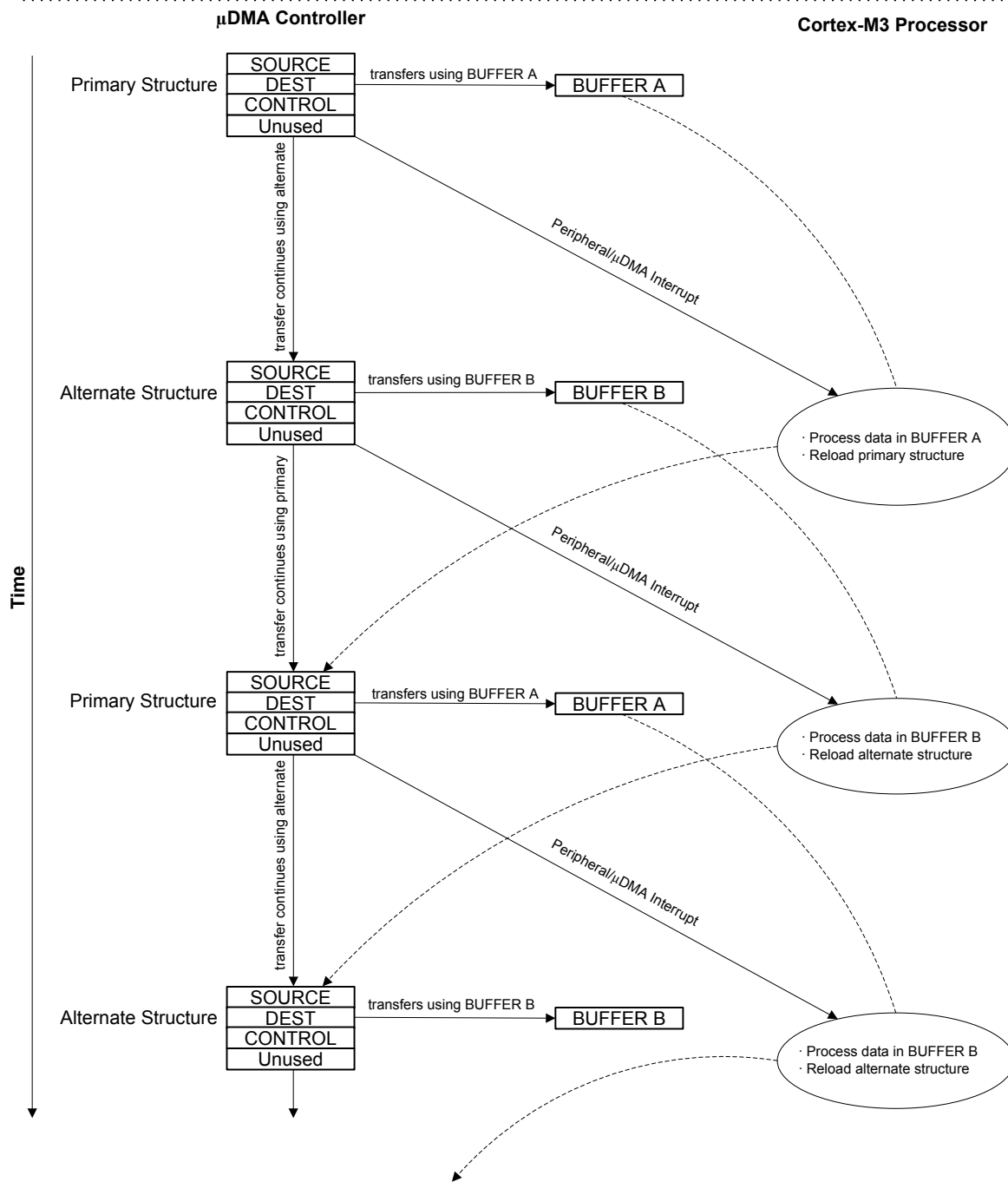
Auto mode is similar to Basic mode, except that once a transfer request is received, the transfer runs to completion, even if the μ DMA request is removed. This mode is suitable for software-triggered transfers. Generally, Auto mode is not used with a peripheral.

When all the items have been transferred using Auto mode, the μ DMA controller sets the mode for that channel to Stop.

9.2.6.4 Ping-Pong

Ping-Pong mode is used to support a continuous data flow to or from a peripheral. To use Ping-Pong mode, both the primary and alternate data structures must be implemented. Both structures are set up by the processor for data transfer between memory and a peripheral. The transfer is started using the primary control structure. When the transfer using the primary control structure is complete, the μ DMA controller reads the alternate control structure for that channel to continue the transfer. Each time this happens, an interrupt is generated, and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and alternate control structures to switch back and forth between buffers as the data flows to or from the peripheral.

Refer to Figure 9-2 for an example showing operation in Ping-Pong mode.

Figure 9-2. Example of Ping-Pong μ DMA Transaction

9.2.6.5 Memory Scatter-Gather

Memory Scatter-Gather mode is a complex mode used when data must be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather μ DMA operation could be used to selectively read the payload of several stored packets of a communication protocol and store them together in sequence in a memory buffer.

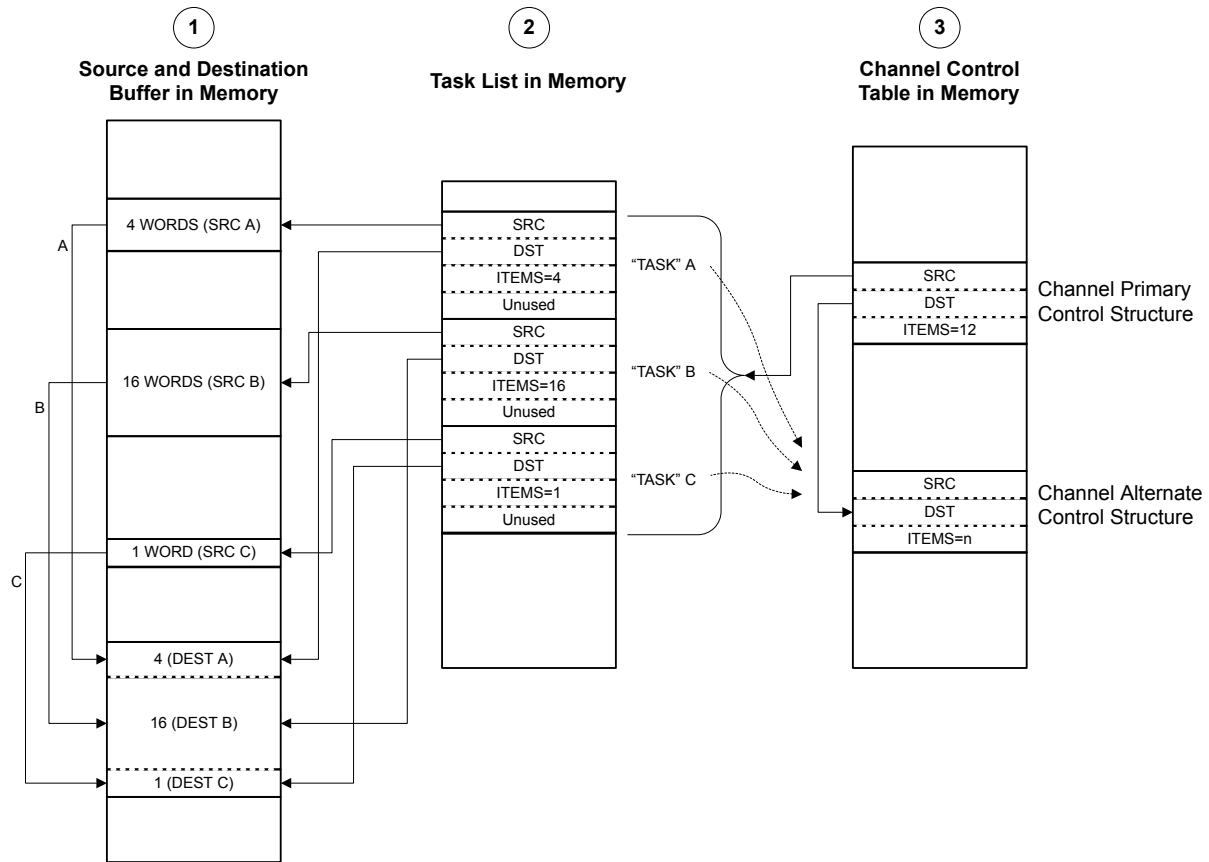
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to Scatter-Gather mode. Each entry in the table is copied in turn to the alternate structure where it is then executed. The μ DMA controller alternates between using the primary control structure to copy the next transfer instruction from the list and then executing the new transfer instruction. The end of the list is marked by programming the control word for the last entry to use Basic transfer mode. Once the last transfer is performed using Basic mode, the μ DMA controller stops. A completion interrupt is generated only after the last transfer. It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly, by programming a write to the software trigger for another channel, or indirectly, by causing a peripheral action that results in a μ DMA request.

By programming the μ DMA controller using this method, a set of arbitrary transfers can be performed based on a single μ DMA request.

Refer to Figure 9-3 on page 267 and Figure 9-4 on page 268, which show an example of operation in Memory Scatter-Gather mode. This example shows a *gather* operation, where data in three separate buffers in memory is copied together into one buffer. Figure 9-3 on page 267 shows how the application sets up a μ DMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

Figure 9-4 on page 268 shows the sequence as the μ DMA controller performs the three sets of copy operations. First, using the primary control structure, the μ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the μ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

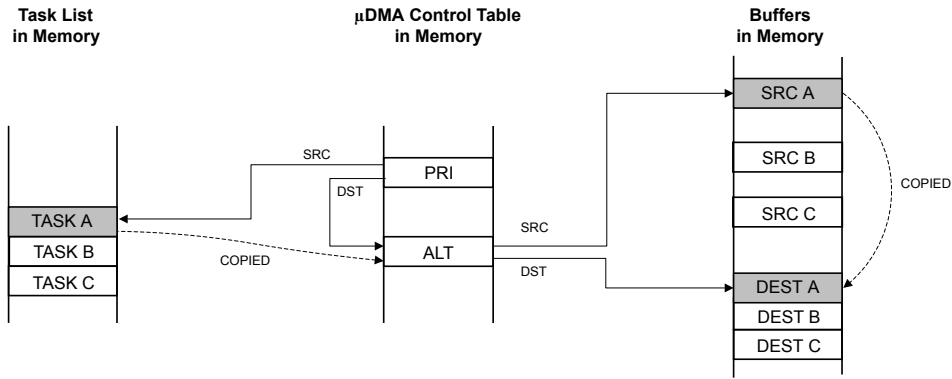
Figure 9-3. Memory Scatter-Gather, Setup and Configuration



NOTES:

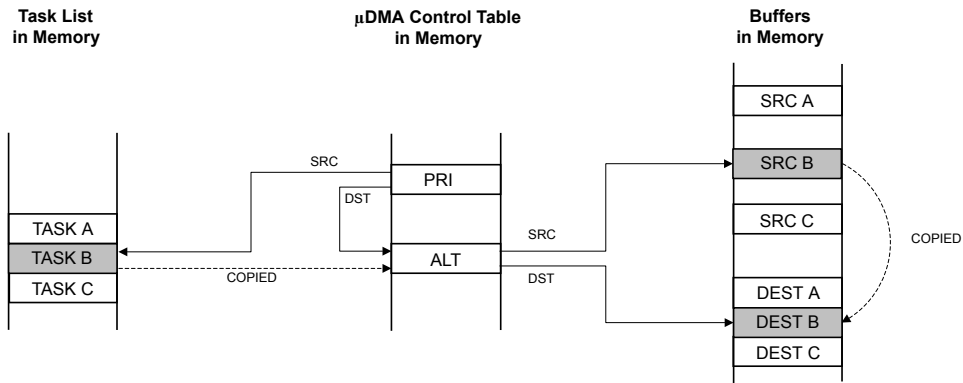
1. Application has a need to copy data items from three separate locations in memory into one combined buffer.
2. Application sets up μ DMA "task list" in memory, which contains the pointers and control configuration for three μ DMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the μ DMA controller.

Figure 9-4. Memory Scatter-Gather, μ DMA Copy Sequence



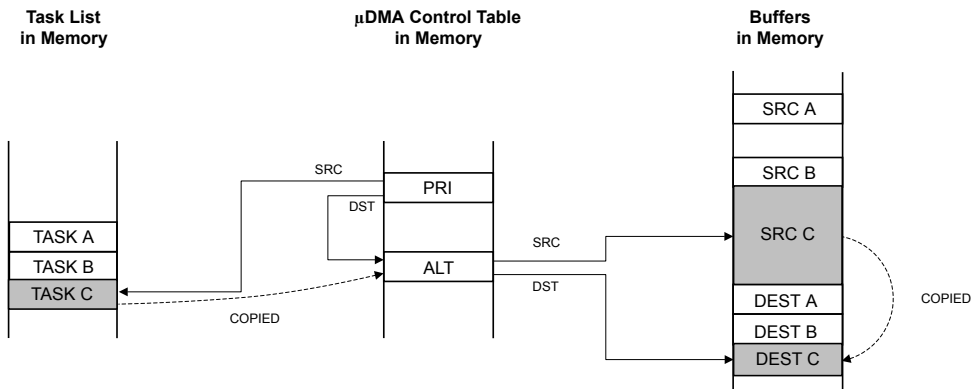
Using the channel's primary control structure, the μ DMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer A to the destination buffer.



Using the channel's primary control structure, the μ DMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer B to the destination buffer.



Using the channel's primary control structure, the μ DMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer C to the destination buffer.

9.2.6.6 Peripheral Scatter-Gather

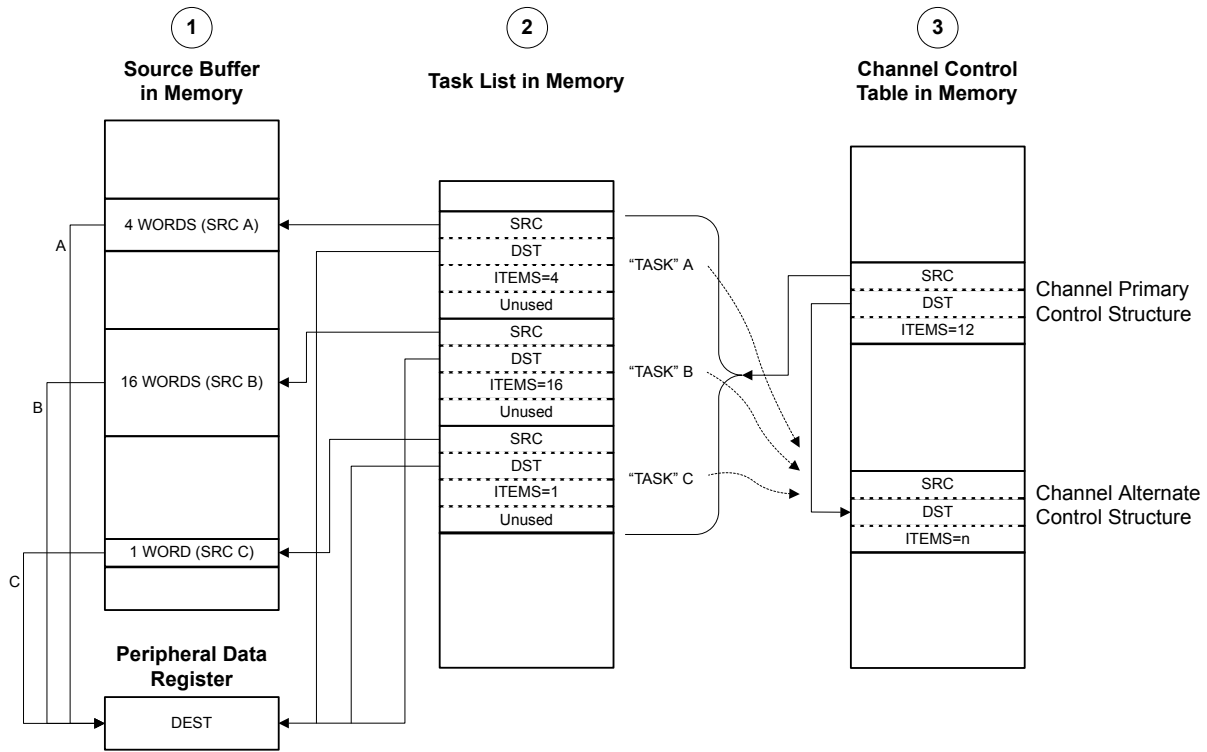
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a μ DMA request. Upon detecting a request from the peripheral, the μ DMA controller uses the primary control structure to copy one entry from the list to the alternate control structure and then performs the transfer. At the end of this transfer, the next transfer is started only if the peripheral again asserts a μ DMA request. The μ DMA controller continues to perform transfers from the list only when the peripheral is making a request, until the last transfer is complete. A completion interrupt is generated only after the last transfer.

By using this method, the μ DMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Refer to Figure 9-5 on page 270 and Figure 9-6 on page 271, which show an example of operation in Peripheral Scatter-Gather mode. This example shows a gather operation, where data from three separate buffers in memory is copied to a single peripheral data register. Figure 9-5 on page 270 shows how the application sets up a μ DMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

Figure 9-6 on page 271 shows the sequence as the μ DMA controller performs the three sets of copy operations. First, using the primary control structure, the μ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the μ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

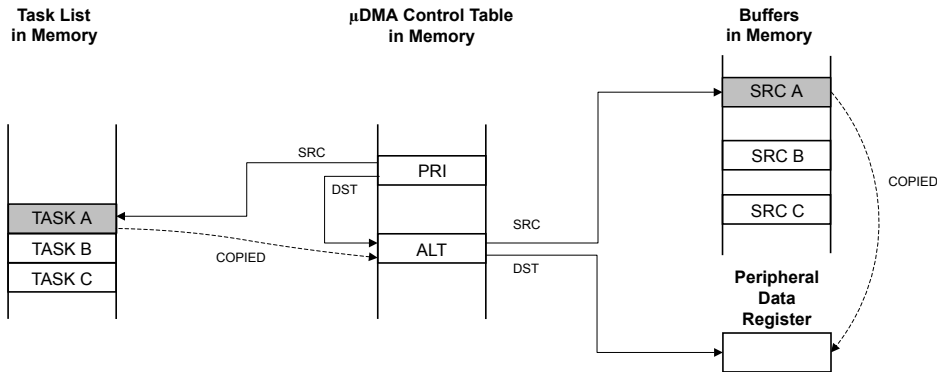
Figure 9-5. Peripheral Scatter-Gather, Setup and Configuration



NOTES:

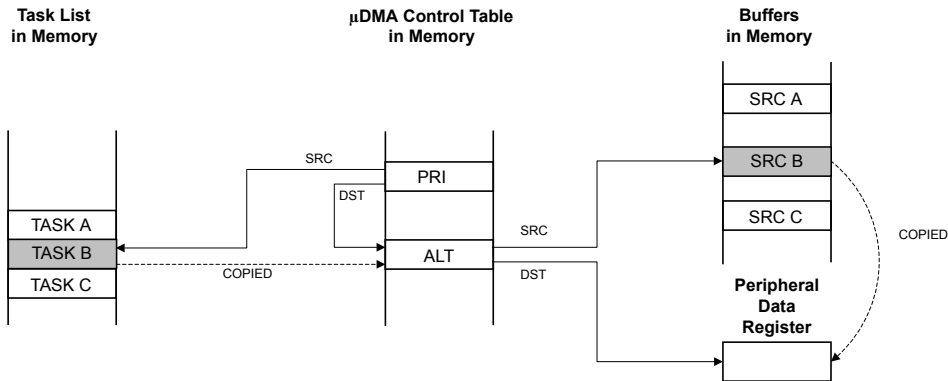
1. Application has a need to copy data items from three separate locations in memory into a peripheral data register.
2. Application sets up μ DMA "task list" in memory, which contains the pointers and control configuration for three μ DMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it is executed by the μ DMA controller.

Figure 9-6. Peripheral Scatter-Gather, μ DMA Copy Sequence



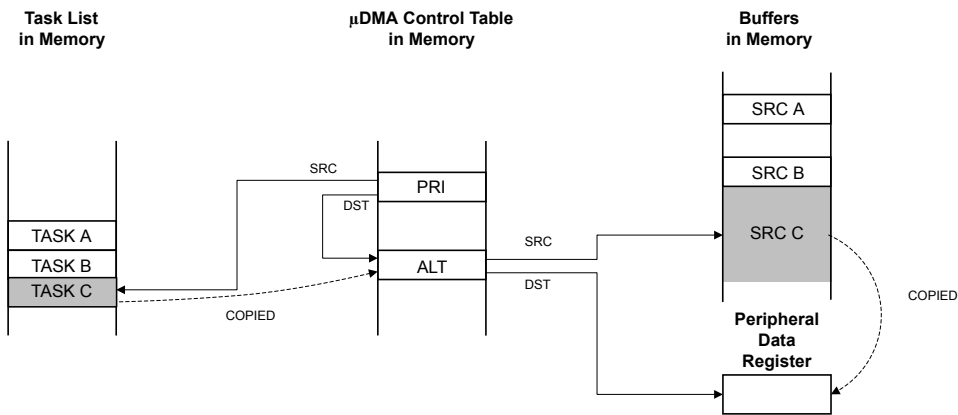
Using the channel's primary control structure, the μ DMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer A to the peripheral data register.



Using the channel's primary control structure, the μ DMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer B to the peripheral data register.



Using the channel's primary control structure, the μ DMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer C to the peripheral data register.

9.2.7 Transfer Size and Increment

The μ DMA controller supports transfer data sizes of 8, 16, or 32 bits. The source and destination data size must be the same for any given transfer. The source and destination address can be auto-incremented by bytes, half-words, or words, or can be set to no increment. The source and destination address increment values can be set independently, and it is not necessary for the address increment to match the data size as long as the increment is the same or larger than the data size. For example, it is possible to perform a transfer using 8-bit data size, but using an address increment of full words (4 bytes). The data to be transferred must be aligned in memory according to the data size (8, 16, or 32 bits).

Table 9-5 shows the configuration to read from a peripheral that supplies 8-bit data.

Table 9-5. μ DMA Read Example: 8-Bit Peripheral

| Field | Configuration |
|-------------------------------|----------------------------------|
| Source data size | 8 bits |
| Destination data size | 8 bits |
| Source address increment | No increment |
| Destination address increment | Byte |
| Source end pointer | Peripheral read FIFO register |
| Destination end pointer | End of the data buffer in memory |

9.2.8 Peripheral Interface

Each peripheral that supports μ DMA has a single request and/or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 9-2 on page 261). The request signal can be disabled or enabled using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The μ DMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the μ DMA channel is configured correctly and enabled, and the peripheral asserts the request signal, the μ DMA controller begins the transfer.

When a μ DMA transfer is complete, the μ DMA controller generates an interrupt, see “Interrupts and Errors” on page 273 for more information.

For more information on how a specific peripheral interacts with the μ DMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

9.2.9 Software Request

One μ DMA channel is dedicated to software-initiated transfers. This channel also has a dedicated interrupt to signal completion of a μ DMA transfer. A transfer is initiated by software by first configuring and enabling the transfer, and then issuing a software request using the **DMA Channel Software Request (DMASWREQ)** register. For software-based transfers, the Auto transfer mode should be used.

It is possible to initiate a transfer on any channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral μ DMA channel, then the completion interrupt occurs on the interrupt vector for the peripheral instead of the software interrupt vector. Any channel may be used for software requests as long as the corresponding peripheral is not using μ DMA for data transfer.

9.2.10 Interrupts and Errors

When a μ DMA transfer is complete, the μ DMA controller generates a completion interrupt on the interrupt vector of the peripheral. Therefore, if μ DMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the μ DMA transfer completion interrupt. If the transfer uses the software μ DMA channel, then the completion interrupt occurs on the dedicated software μ DMA interrupt vector (see Table 9-6).

When μ DMA is enabled for a peripheral, the μ DMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (the interrupts are still reported in the peripheral's interrupt registers). Thus, when a large amount of data is transferred using μ DMA, instead of receiving multiple interrupts from the peripheral as data flows, the interrupt controller receives only one interrupt when the transfer is complete. Unmasked peripheral error interrupts continue to be sent to the interrupt controller.

When a μ DMA channel generates a completion interrupt, the `CHIS` bit corresponding to the peripheral channel is set in the **DMA Channel Interrupt Status (DMACHIS)** register (see page 308). This register can be used by the peripheral interrupt handler code to determine if the interrupt was caused by the μ DMA channel or an error event reported by the peripheral's interrupt registers. The completion interrupt request from the μ DMA controller is automatically cleared when the interrupt handler is activated.

If the μ DMA controller encounters a bus or memory protection error as it attempts to perform a data transfer, it disables the μ DMA channel that caused the error and generates an interrupt on the μ DMA error interrupt vector. The processor can read the **DMA Bus Error Clear (DMAERRCLR)** register to determine if an error is pending. The `ERRCLR` bit is set if an error occurred. The error can be cleared by writing a 1 to the `ERRCLR` bit.

Table 9-6 shows the dedicated interrupt assignments for the μ DMA controller.

Table 9-6. μ DMA Interrupt Assignments

| Interrupt | Assignment |
|-----------|-------------------------------------|
| 46 | μ DMA Software Channel Transfer |
| 47 | μ DMA Error |

9.3 Initialization and Configuration

9.3.1 Module Initialization

Before the μ DMA controller can be used, it must be enabled in the System Control block and in the peripheral. The location of the channel control structure must also be programmed.

The following steps should be performed one time during system initialization:

1. The μ DMA peripheral must be enabled in the System Control block. To do this, set the `UDMA` bit of the System Control **RCGC2** register (see page 184).
2. Enable the μ DMA controller by setting the `MASTEREN` bit of the **DMA Configuration (DMACFG)** register.
3. Program the location of the channel control table by writing the base address of the table to the **DMA Channel Control Base Pointer (DMACTLBASE)** register. The base address must be aligned on a 1024-byte boundary.

9.3.2 Configuring a Memory-to-Memory Transfer

μ DMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

9.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Program bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μ DMA controller to respond to single and burst requests.
4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μ DMA controller to recognize requests for this channel.

9.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example transfers 256 words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in Table 9-7.

Table 9-7. Channel Control Structure Offsets for Channel 30

| Offset | Description |
|----------------------------|------------------------------------|
| Control Table Base + 0x1E0 | Channel 30 Source End Pointer |
| Control Table Base + 0x1E4 | Channel 30 Destination End Pointer |
| Control Table Base + 0x1E8 | Channel 30 Control Word |

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Program the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.
2. Program the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 9-8.

Table 9-8. Channel Control Word Configuration for Memory Transfer Example

| Field in DMACHCTL | Bits | Value | Description |
|-------------------|-------|-------|--------------------------------------|
| DSTINC | 31:30 | 2 | 32-bit destination address increment |
| DSTSIZE | 29:28 | 2 | 32-bit destination data size |
| SRCINC | 27:26 | 2 | 32-bit source address increment |
| SRCSIZE | 25:24 | 2 | 32-bit source data size |
| reserved | 23:18 | 0 | Reserved |

Table 9-8. Channel Control Word Configuration for Memory Transfer Example (continued)

| Field in DMACHCTL | Bits | Value | Description |
|-------------------|-------|-------|--------------------------------|
| ARBSIZE | 17:14 | 3 | Arbitrates after 8 transfers |
| XFERSIZE | 13:4 | 255 | Transfer 256 items |
| NXTUSEBURST | 3 | 0 | N/A for this transfer type |
| XFERMODE | 2:0 | 2 | Use Auto-request transfer mode |

9.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the **DMA Channel Enable Set (DMAENASET)** register.
2. Issue a transfer request by setting bit 30 of the **DMA Channel Software Request (DMASWREQ)** register.

The μ DMA transfer begins. If the interrupt is enabled, then the processor is notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the **DMAENASET** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the **XFERMODE** field of the channel control word at offset 0x1E8. This field is automatically cleared at the end of the transfer.

9.3.3 Configuring a Peripheral for Simple Transmit

This example configures the μ DMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral uses μ DMA channel 7.

9.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μ DMA controller to respond to single and burst requests.
4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μ DMA controller to recognize requests for this channel.

9.3.3.2 Configure the Channel Control Structure

This example transfers 64 bytes from a memory buffer to the peripheral's transmit FIFO register using μ DMA channel 7. The control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 9-9.

Table 9-9. Channel Control Structure Offsets for Channel 7

| Offset | Description |
|----------------------------|------------------------------|
| Control Table Base + 0x070 | Channel 7 Source End Pointer |

Table 9-9. Channel Control Structure Offsets for Channel 7 (continued)

| Offset | Description |
|----------------------------|-----------------------------------|
| Control Table Base + 0x074 | Channel 7 Destination End Pointer |
| Control Table Base + 0x078 | Channel 7 Control Word |

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Program the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.
2. Program the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to Table 9-10.

Table 9-10. Channel Control Word Configuration for Peripheral Transmit Example

| Field in DMACHCTL | Bits | Value | Description |
|-------------------|-------|-------|--|
| DSTINC | 31:30 | 3 | Destination address does not increment |
| DSTSIZE | 29:28 | 0 | 8-bit destination data size |
| SRCINC | 27:26 | 0 | 8-bit source address increment |
| SRCSIZE | 25:24 | 0 | 8-bit source data size |
| reserved | 23:18 | 0 | Reserved |
| ARBSIZE | 17:14 | 2 | Arbitrates after 4 transfers |
| XFERSIZE | 13:4 | 63 | Transfer 64 items |
| NXTUSEBURST | 3 | 0 | N/A for this transfer type |
| XFERMODE | 2:0 | 1 | Use Basic transfer mode |

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst `SET[7]` bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

9.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the **DMA Channel Enable Set (DMAENASET)** register.

The μ DMA controller is now configured for transfer on channel 7. The controller makes transfers to the peripheral whenever the peripheral asserts a μ DMA request. The transfers continue until the entire buffer of 64 bytes has been transferred. When that happens, the μ DMA controller disables the channel and sets the `XFERMODE` field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the **DMA Channel Enable Set (DMAENASET)** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the `XFERMODE` field of the channel control word at offset 0x078. This field is automatically cleared at the end of the transfer.

If peripheral interrupts are enabled, then the peripheral interrupt handler receives an interrupt when the entire transfer is complete.

9.3.4 Configuring a Peripheral for Ping-Pong Receive

This example configures the μ DMA controller to continuously receive 8-bit data from a peripheral into a pair of 64-byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral uses μ DMA channel 8.

9.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 8 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 8 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 8 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μ DMA controller to respond to single and burst requests.
4. Set bit 8 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μ DMA controller to recognize requests for this channel.

9.3.4.2 Configure the Channel Control Structure

This example transfers bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the μ DMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 9-11.

Table 9-11. Primary and Alternate Channel Control Structure Offsets for Channel 8

| Offset | Description |
|----------------------------|---|
| Control Table Base + 0x080 | Channel 8 Primary Source End Pointer |
| Control Table Base + 0x084 | Channel 8 Primary Destination End Pointer |
| Control Table Base + 0x088 | Channel 8 Primary Control Word |
| Control Table Base + 0x280 | Channel 8 Alternate Source End Pointer |
| Control Table Base + 0x284 | Channel 8 Alternate Destination End Pointer |
| Control Table Base + 0x288 | Channel 8 Alternate Control Word |

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Program the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.

2. Program the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.
3. Program the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.
4. Program the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088 and the alternate control word at offset 0x288 are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 9-12.
2. Program the alternate channel control word at offset 0x288 according to Table 9-12.

Table 9-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example

| Field in DMACHCTL | Bits | Value | Description |
|-------------------|-------|-------|-------------------------------------|
| DSTINC | 31:30 | 0 | 8-bit destination address increment |
| DSTSIZE | 29:28 | 0 | 8-bit destination data size |
| SRCINC | 27:26 | 3 | Source address does not increment |
| SRCSIZE | 25:24 | 0 | 8-bit source data size |
| reserved | 23:18 | 0 | Reserved |
| ARBSIZE | 17:14 | 3 | Arbitrates after 8 transfers |
| XFERSIZE | 13:4 | 63 | Transfer 64 items |
| NXTUSEBURST | 3 | 0 | N/A for this transfer type |
| XFERMODE | 2:0 | 3 | Use Ping-Pong transfer mode |

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst `SET[8]` bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

9.3.4.3 Configure the Peripheral Interrupt

An interrupt handler should be configured when using μ DMA Ping-Pong mode, it is best to use an interrupt handler. However, the Ping-Pong mode can be configured without interrupts by polling. The interrupt handler is triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

9.3.4.4 Enable the μ DMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the **DMA Channel Enable Set (DMAENASET)** register.

9.3.4.5 Process Interrupts

The μ DMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the μ DMA request signal, the μ DMA controller makes transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it switches to the alternate channel control structure and makes transfers into buffer B. At the same time, the primary channel control word mode field is configured to indicate Stopped, and an interrupt is

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data or set a flag that the data must be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the `XFERMODE` field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:
 - a. Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.
 - b. Reprogram the primary channel control word at offset 0x88 according to Table 9-12 on page 278.
2. Read the alternate channel control word at offset 0x288 and check the `XFERMODE` field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:
 - a. Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.
 - b. Reprogram the alternate channel control word at offset 0x288 according to Table 9-12 on page 278.

9.3.5 Configuring Alternate Channels

Alternate peripherals can be assigned to each μ DMA channel using the **DMACHALT** register. Each bit represents a μ DMA channel. If the bit is set, then the alternate peripheral is used for the channel.

Refer to Table 9-1 on page 260 for alternate channel assignments.

For example, to use SS11 Receive on channel 8 instead of UART0, set bit 8 of the **DMACHALT** register.

9.4 Register Map

Table 9-13 on page 280 lists the μ DMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, that is, the base address is n/a (not applicable). In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See "Channel Configuration" on page 262 and Table 9-3 on page 263 for a description of how the entries in the channel control table are located in memory. The μ DMA register addresses are given as a hexadecimal increment, relative to the μ DMA base address of 0x400F.F000. Note that the μ DMA module clock must be enabled before the registers can be programmed (see page 184).

Table 9-13. μ DMA Register Map

| Offset | Name | Type | Reset | Description | See page |
|---|-----------------|-------|-------------|---|----------|
| μDMA Channel Control Structure (Offset from Channel Control Table Base) | | | | | |
| 0x000 | DMASRCENDP | R/W | - | DMA Channel Source Address End Pointer | 282 |
| 0x004 | DMADSTENDP | R/W | - | DMA Channel Destination Address End Pointer | 283 |
| 0x008 | DMACHCTL | R/W | - | DMA Channel Control Word | 284 |
| μDMA Registers (Offset from μDMA Base Address) | | | | | |
| 0x000 | DMASTAT | RO | 0x001F.0000 | DMA Status | 289 |
| 0x004 | DMACFG | WO | - | DMA Configuration | 291 |
| 0x008 | DMACTLBASE | R/W | 0x0000.0000 | DMA Channel Control Base Pointer | 292 |
| 0x00C | DMAALTBASE | RO | 0x0000.0200 | DMA Alternate Channel Control Base Pointer | 293 |
| 0x010 | DMAWAITSTAT | RO | 0x0000.0000 | DMA Channel Wait-on-Request Status | 294 |
| 0x014 | DMASWREQ | WO | - | DMA Channel Software Request | 295 |
| 0x018 | DMAUSEBURSTSET | R/W | 0x0000.0000 | DMA Channel Useburst Set | 296 |
| 0x01C | DMAUSEBURSTCLR | WO | - | DMA Channel Useburst Clear | 297 |
| 0x020 | DMAREQMASKSET | R/W | 0x0000.0000 | DMA Channel Request Mask Set | 298 |
| 0x024 | DMAREQMASKCLR | WO | - | DMA Channel Request Mask Clear | 299 |
| 0x028 | DMAENASET | R/W | 0x0000.0000 | DMA Channel Enable Set | 300 |
| 0x02C | DMAENACLAR | WO | - | DMA Channel Enable Clear | 301 |
| 0x030 | DMAALTSET | R/W | 0x0000.0000 | DMA Channel Primary Alternate Set | 302 |
| 0x034 | DMAALTCLR | WO | - | DMA Channel Primary Alternate Clear | 303 |
| 0x038 | DMAPRIOSET | R/W | 0x0000.0000 | DMA Channel Priority Set | 304 |
| 0x03C | DMAPRIOCLR | WO | - | DMA Channel Priority Clear | 305 |
| 0x04C | DMAERRCLR | R/W | 0x0000.0000 | DMA Bus Error Clear | 306 |
| 0x500 | DMACHALT | R/W | 0x0000.0000 | DMA Channel Alternate Select | 307 |
| 0x504 | DMACHIS | R/W1C | 0x0000.0000 | DMA Channel Interrupt Status | 308 |
| 0xFD0 | DMAPeriphID4 | RO | 0x0000.0004 | DMA Peripheral Identification 4 | 313 |
| 0xFE0 | DMAPeriphID0 | RO | 0x0000.0030 | DMA Peripheral Identification 0 | 309 |
| 0xFE4 | DMAPeriphID1 | RO | 0x0000.00B2 | DMA Peripheral Identification 1 | 310 |
| 0xFE8 | DMAPeriphID2 | RO | 0x0000.000B | DMA Peripheral Identification 2 | 311 |
| 0xFEC | DMAPeriphID3 | RO | 0x0000.0000 | DMA Peripheral Identification 3 | 312 |
| 0xFF0 | DMAPrimeCellID0 | RO | 0x0000.000D | DMA PrimeCell Identification 0 | 314 |
| 0xFF4 | DMAPrimeCellID1 | RO | 0x0000.00F0 | DMA PrimeCell Identification 1 | 315 |
| 0xFF8 | DMAPrimeCellID2 | RO | 0x0000.0005 | DMA PrimeCell Identification 2 | 316 |

Table 9-13. μ DMA Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|-------------|--------------------------------|----------|
| 0xFFC | DMAPrimeCellID3 | RO | 0x0000.00B1 | DMA PrimeCell Identification 3 | 317 |

9.5 μ DMA Channel Control Structure

The μ DMA Channel Control Structure holds the transfer settings for a μ DMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 262 for an explanation of the Channel Control Table and the Channel Control Structure.

The channel control structure is one entry in the channel control table. Each channel has a primary and alternate structure. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

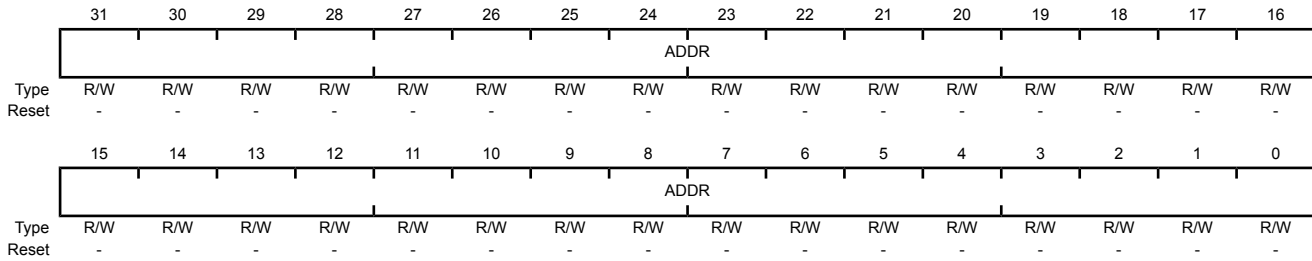
Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure and is used to specify the source address for a μ DMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the μ DMA module base address.

DMA Channel Source Address End Pointer (DMASRCENDP)

Base n/a
Offset 0x000
Type R/W, reset -



| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 31:0 | ADDR | R/W | - | Source Address End Pointer This field points to the last address of the μ DMA transfer source (inclusive). If the source address is not incrementing (the SRCINC field in the DMACHCTL register is 0x3), then this field points at the source location itself (such as a peripheral data register). |

Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

DMA Channel Destination Address End Pointer (DMADSTENDP) is part of the Channel Control Structure and is used to specify the destination address for a μ DMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the μ DMA module base address.

DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a
Offset 0x004
Type R/W, reset -

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

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---------------------------------|
| 31:0 | ADDR | R/W | - | Destination Address End Pointer |

This field points to the last address of the μ DMA transfer destination (inclusive). If the destination address is not incrementing (the `DSTINC` field in the `DMACHCTL` register is 0x3), then this field points at the destination location itself (such as a peripheral data register).

Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

DMA Channel Control Word (DMACHCTL) is part of the Channel Control Structure and is used to specify parameters of a μ DMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the μ DMA module base address.

DMA Channel Control Word (DMACHCTL)

Base n/a
Offset 0x008
Type R/W, reset -

| | | | | | | | | | | | | | | | | |
|-------|---------|-----|----------|-----|--------|-----|---------|-----|----------|-----|-----|-----|-------------|----------|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | DSTINC | | DSTSIZE | | SRCINC | | SRCSIZE | | reserved | | | | ARBSIZE | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ARBSIZE | | XFERSIZE | | | | | | | | | | NXTUSEBURST | XFERMODE | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|--|------|-------|--|-------|-------------|-----|--------------------------------------|-----|--|-----|---------------------------------------|-----|--|
| 31:30 | DSTINC | R/W | - | <p>Destination Address Increment</p> <p>This field configures the destination address increment.</p> <p>The address increment value must be equal or greater than the value of the destination size (DSTSIZE).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte Increment by 8-bit locations</td> </tr> <tr> <td>0x1</td> <td>Half-word Increment by 16-bit locations</td> </tr> <tr> <td>0x2</td> <td>Word Increment by 32-bit locations</td> </tr> <tr> <td>0x3</td> <td>No increment Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel</td> </tr> </tbody> </table> | Value | Description | 0x0 | Byte Increment by 8-bit locations | 0x1 | Half-word Increment by 16-bit locations | 0x2 | Word Increment by 32-bit locations | 0x3 | No increment Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Byte Increment by 8-bit locations | | | | | | | | | | | | | |
| 0x1 | Half-word Increment by 16-bit locations | | | | | | | | | | | | | |
| 0x2 | Word Increment by 32-bit locations | | | | | | | | | | | | | |
| 0x3 | No increment Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---|------|-------|--|-------|-------------|-----|--------------------------------------|-----|--|-----|---------------------------------------|-----|---|
| 29:28 | DSTSIZE | R/W | - | <p>Destination Data Size</p> <p>This field configures the destination item data size.</p> <p>Note: DSTSIZE must be the same as SRCSIZE.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte 8-bit data size</td> </tr> <tr> <td>0x1</td> <td>Half-word 16-bit data size</td> </tr> <tr> <td>0x2</td> <td>Word 32-bit data size</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </tbody> </table> | Value | Description | 0x0 | Byte 8-bit data size | 0x1 | Half-word 16-bit data size | 0x2 | Word 32-bit data size | 0x3 | Reserved |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Byte 8-bit data size | | | | | | | | | | | | | |
| 0x1 | Half-word 16-bit data size | | | | | | | | | | | | | |
| 0x2 | Word 32-bit data size | | | | | | | | | | | | | |
| 0x3 | Reserved | | | | | | | | | | | | | |
| 27:26 | SRCINC | R/W | - | <p>Source Address Increment</p> <p>This field configures the source address increment.</p> <p>The address increment value must be equal or greater than the value of the source size (SRCSIZE).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte Increment by 8-bit locations</td> </tr> <tr> <td>0x1</td> <td>Half-word Increment by 16-bit locations</td> </tr> <tr> <td>0x2</td> <td>Word Increment by 32-bit locations</td> </tr> <tr> <td>0x3</td> <td>No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel</td> </tr> </tbody> </table> | Value | Description | 0x0 | Byte Increment by 8-bit locations | 0x1 | Half-word Increment by 16-bit locations | 0x2 | Word Increment by 32-bit locations | 0x3 | No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Byte Increment by 8-bit locations | | | | | | | | | | | | | |
| 0x1 | Half-word Increment by 16-bit locations | | | | | | | | | | | | | |
| 0x2 | Word Increment by 32-bit locations | | | | | | | | | | | | | |
| 0x3 | No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel | | | | | | | | | | | | | |
| 25:24 | SRCSIZE | R/W | - | <p>Source Data Size</p> <p>This field configures the source item data size.</p> <p>Note: DSTSIZE must be the same as SRCSIZE.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte 8-bit data size.</td> </tr> <tr> <td>0x1</td> <td>Half-word 16-bit data size.</td> </tr> <tr> <td>0x2</td> <td>Word 32-bit data size.</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </tbody> </table> | Value | Description | 0x0 | Byte 8-bit data size. | 0x1 | Half-word 16-bit data size. | 0x2 | Word 32-bit data size. | 0x3 | Reserved |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Byte 8-bit data size. | | | | | | | | | | | | | |
| 0x1 | Half-word 16-bit data size. | | | | | | | | | | | | | |
| 0x2 | Word 32-bit data size. | | | | | | | | | | | | | |
| 0x3 | Reserved | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|-------------------------------------|------|-------|--|-------|-------------|-----|------------|--|-------------------------------------|-----|-------------|-----|-------------|-----|-------------|-----|--------------|-----|--------------|-----|--------------|-----|---------------|-----|---------------|-----|---------------|---------|----------------|
| 23:18 | reserved | R/W | - | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17:14 | ARBSIZE | R/W | - | <p>Arbitration Size</p> <p>This field configures the number of transfers that can occur before the μDMA controller re-arbitrates. The possible arbitration rate configurations represent powers of 2 and are shown below.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>1 Transfer</td> </tr> <tr> <td></td> <td>Arbitrates after each μDMA transfer</td> </tr> <tr> <td>0x1</td> <td>2 Transfers</td> </tr> <tr> <td>0x2</td> <td>4 Transfers</td> </tr> <tr> <td>0x3</td> <td>8 Transfers</td> </tr> <tr> <td>0x4</td> <td>16 Transfers</td> </tr> <tr> <td>0x5</td> <td>32 Transfers</td> </tr> <tr> <td>0x6</td> <td>64 Transfers</td> </tr> <tr> <td>0x7</td> <td>128 Transfers</td> </tr> <tr> <td>0x8</td> <td>256 Transfers</td> </tr> <tr> <td>0x9</td> <td>512 Transfers</td> </tr> <tr> <td>0xA-0xF</td> <td>1024 Transfers</td> </tr> </tbody> </table> <p>In this configuration, no arbitration occurs during the μDMA transfer because the maximum transfer size is 1024.</p> | Value | Description | 0x0 | 1 Transfer | | Arbitrates after each μDMA transfer | 0x1 | 2 Transfers | 0x2 | 4 Transfers | 0x3 | 8 Transfers | 0x4 | 16 Transfers | 0x5 | 32 Transfers | 0x6 | 64 Transfers | 0x7 | 128 Transfers | 0x8 | 256 Transfers | 0x9 | 512 Transfers | 0xA-0xF | 1024 Transfers |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | 1 Transfer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Arbitrates after each μDMA transfer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | 2 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | 4 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | 8 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | 16 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | 32 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | 64 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | 128 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | 256 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9 | 512 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA-0xF | 1024 Transfers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13:4 | XFERSIZE | R/W | - | <p>Transfer Size (minus 1)</p> <p>This field configures the total number of items to transfer. The value of this field is 1 less than the number to transfer (value 0 means transfer 1 item). The maximum value for this 10-bit field is 1023 which represents a transfer size of 1024 items.</p> <p>The transfer size is the number of items, not the number of bytes. If the data size is 32 bits, then this value is the number of 32-bit words to transfer.</p> <p>The μDMA controller updates this field immediately prior to entering the arbitration process, so it contains the number of outstanding items that is necessary to complete the μDMA cycle.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | NXTUSEBURST | R/W | - | <p>Next Useburst</p> <p>This field controls whether the Useburst SET[n] bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the μDMA controller uses single transfers to complete the transaction. If this bit is set, then the controller uses a burst transfer to complete the last transfer.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | | | |
|-----------|-------------------------------------|------|-------|---|-------|-------------|-----|------|-----|-------|-----|--------------|-----|-----------|-----|-----------------------|-----|---------------------------------|-----|---------------------------|-----|-------------------------------------|
| 2:0 | XFERMODE | R/W | - | <p>μDMA Transfer Mode</p> <p>This field configures the operating mode of the μDMA cycle. Refer to “Transfer Modes” on page 264 for a detailed explanation of transfer modes.</p> <p>Because this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Stop</td> </tr> <tr> <td>0x1</td> <td>Basic</td> </tr> <tr> <td>0x2</td> <td>Auto-Request</td> </tr> <tr> <td>0x3</td> <td>Ping-Pong</td> </tr> <tr> <td>0x4</td> <td>Memory Scatter-Gather</td> </tr> <tr> <td>0x5</td> <td>Alternate Memory Scatter-Gather</td> </tr> <tr> <td>0x6</td> <td>Peripheral Scatter-Gather</td> </tr> <tr> <td>0x7</td> <td>Alternate Peripheral Scatter-Gather</td> </tr> </tbody> </table> | Value | Description | 0x0 | Stop | 0x1 | Basic | 0x2 | Auto-Request | 0x3 | Ping-Pong | 0x4 | Memory Scatter-Gather | 0x5 | Alternate Memory Scatter-Gather | 0x6 | Peripheral Scatter-Gather | 0x7 | Alternate Peripheral Scatter-Gather |
| Value | Description | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | Stop | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | Basic | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | Auto-Request | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | Ping-Pong | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | Memory Scatter-Gather | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | Alternate Memory Scatter-Gather | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | Peripheral Scatter-Gather | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Alternate Peripheral Scatter-Gather | | | | | | | | | | | | | | | | | | | | | |

XFERMODE Bit Field Values.

Stop

Channel is stopped or configuration data is invalid. No more transfers can occur.

Basic

For each trigger (whether from a peripheral or a software request), the μDMA controller performs the number of transfers specified by the `ARBSIZE` field.

Auto-Request

The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of `XFERSIZE` items without any further requests.

Ping-Pong

This mode uses both the primary and alternate control structures for this channel. When the number of transfers specified by the `XFERSIZE` field have completed for the current control structure (primary or alternate), the μDMA controller switches to the other one. These switches continue until one of the control structures is not set to ping-pong mode. At that point, the μDMA controller stops. An interrupt is generated on completion of the transfers configured by each control structure. See “Ping-Pong” on page 264.

Memory Scatter-Gather

When using this mode, the primary control structure for the channel is configured to allow a list of operations (tasks) to be performed. The source address pointer specifies the start of a table of tasks to be copied to the alternate control structure for this channel. The `XFERMODE` field for the alternate control structure should be configured to 0x5 (Alternate memory scatter-gather) to perform the task. When the task completes, the μDMA switches back to the primary channel control structure, which then copies the next task to the alternate control structure. This process continues until the table of tasks is empty. The last task must have an `XFERMODE` value other than 0x5. Note that for continuous operation, the last task can update the primary channel control structure back to the start of the list or to another list. See “Memory Scatter-Gather” on page 265.

Alternate Memory Scatter-Gather

This value must be used in the alternate channel control data structure when the μDMA controller operates in Memory Scatter-Gather mode.

Peripheral Scatter-Gather

This value must be used in the primary channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode. In this mode, the μDMA controller operates exactly the same as in Memory Scatter-Gather mode, except that instead of performing the number of transfers specified by the `XFERSIZE` field in the alternate control structure at one time, the μDMA controller only performs the number of transfers specified by the `ARBSIZE` field per trigger; see Basic mode for details. See “Peripheral Scatter-Gather” on page 269.

Alternate Peripheral Scatter-Gather

This value must be used in the alternate channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode.

9.6 μDMA Register Descriptions

The register addresses given are relative to the μDMA base address of 0x400F.F000.

Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the μ DMA controller. You cannot read this register when the μ DMA controller is in the reset state.

DMA Status (DMASTAT)

Base 0x400F.F000

Offset 0x000

Type RO, reset 0x001F.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-------|----|----|----------|----------|----|----|--------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | DMACHANS | | | | |
| 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 | 1 | 1 | 1 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | STATE | | | | reserved | | | MASTEN |
| 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 | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|---|------|-------|--|-------|-------------|-----|------|-----|----------------------------------|-----|-----------------------------|-----|----------------------------------|-----|----------------------|-----|---------------------------|-----|---|-----|----------------------------------|-----|---------|-----|------|---------|-----------|
| 31:21 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | | | |
| 20:16 | DMACHANS | RO | 0x1F | Available μ DMA Channels Minus 1 This field contains a value equal to the number of μ DMA channels the μ DMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 μ DMA channels. | | | | | | | | | | | | | | | | | | | | | | | | |
| 15:8 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | | | |
| 7:4 | STATE | RO | 0x0 | Control State Machine Status This field shows the current status of the control state machine. Status can be one of the following. <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Idle</td> </tr> <tr> <td>0x1</td> <td>Reading channel controller data.</td> </tr> <tr> <td>0x2</td> <td>Reading source end pointer.</td> </tr> <tr> <td>0x3</td> <td>Reading destination end pointer.</td> </tr> <tr> <td>0x4</td> <td>Reading source data.</td> </tr> <tr> <td>0x5</td> <td>Writing destination data.</td> </tr> <tr> <td>0x6</td> <td>Waiting for μDMA request to clear.</td> </tr> <tr> <td>0x7</td> <td>Writing channel controller data.</td> </tr> <tr> <td>0x8</td> <td>Stalled</td> </tr> <tr> <td>0x9</td> <td>Done</td> </tr> <tr> <td>0xA-0xF</td> <td>Undefined</td> </tr> </tbody> </table> | Value | Description | 0x0 | Idle | 0x1 | Reading channel controller data. | 0x2 | Reading source end pointer. | 0x3 | Reading destination end pointer. | 0x4 | Reading source data. | 0x5 | Writing destination data. | 0x6 | Waiting for μ DMA request to clear. | 0x7 | Writing channel controller data. | 0x8 | Stalled | 0x9 | Done | 0xA-0xF | Undefined |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | Idle | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | Reading channel controller data. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | Reading source end pointer. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | Reading destination end pointer. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | Reading source data. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | Writing destination data. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | Waiting for μ DMA request to clear. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Writing channel controller data. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | Stalled | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9 | Done | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA-0xF | Undefined | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3:1 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 0 | MASTEN | RO | 0 | Master Enable Status |
| | | | | Value Description |
| | | | | 0 The μ DMA controller is disabled. |
| | | | | 1 The μ DMA controller is enabled. |

Register 5: DMA Configuration (DMACFG), offset 0x004

The **DMACFG** register controls the configuration of the μ DMA controller.

DMA Configuration (DMACFG)

Base 0x400F.F000

Offset 0x004

Type WO, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:1 | reserved | WO | - | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | MASTEN | WO | - | Controller Master Enable |
| | | | | Value Description |
| | | | | 0 Disables the μ DMA controller. |
| | | | | 1 Enables μ DMA controller. |

Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

The **DMACTLBASE** register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that must be assigned to the μ DMA controller depends on the number of μ DMA channels used and whether the alternate channel control data structure is used. See “Channel Configuration” on page 262 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. This register cannot be read when the μ DMA controller is in the reset state.

DMA Channel Control Base Pointer (DMACTLBASE)

Base 0x400F.F000
 Offset 0x008
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:10 | ADDR | R/W | 0x0000.00 | Channel Control Base Address This field contains the pointer to the base address of the channel control table. The base address must be 1024-byte aligned. |
| 9:0 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C

The **DMAALTBASE** register returns the base address of the alternate channel control data. This register removes the necessity for application software to calculate the base address of the alternate channel control structures. This register cannot be read when the μ DMA controller is in the reset state.

DMA Alternate Channel Control Base Pointer (DMAALTBASE)

Base 0x400F.F000
Offset 0x00C
Type RO, reset 0x0000.0200

| | | | | | | | | | | | | | | | | |
|-------|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | ADDR | | | | | | | | | | | | | | | |
| 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 |
| | ADDR | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

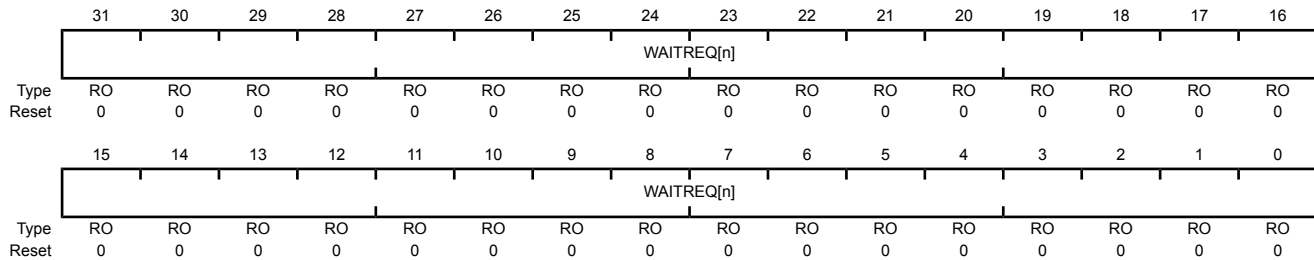
| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------------|--|
| 31:0 | ADDR | RO | 0x0000.0200 | Alternate Channel Address Pointer This field provides the base address of the alternate channel control structures. |

Register 8: DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010

This read-only register indicates that the μ DMA channel is waiting on a request. A peripheral can hold off the μ DMA from performing a single request until the peripheral is ready for a burst request to enhance the μ DMA performance. The use of this feature is dependent on the design of the peripheral and is not controllable by software in any way. This register cannot be read when the μ DMA controller is in the reset state.

DMA Channel Wait-on-Request Status (DMAWAITSTAT)

Base 0x400F.F000
 Offset 0x010
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|------------|------|-------------|--|
| 31:0 | WAITREQ[n] | RO | 0x0000.0000 | Channel [n] Wait Status These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0. Value Description 1 The corresponding channel is waiting on a request. 0 The corresponding channel is not waiting on a request. |

Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014

Each bit of the **DMASWREQ** register represents the corresponding μ DMA channel. Setting a bit generates a request for the specified μ DMA channel.

DMA Channel Software Request (DMASWREQ)

Base 0x400F.F000

Offset 0x014

Type WO, reset -



| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|--|------|-------|---|---|--|---|-----------------------|
| 31:0 | SWREQ[n] | WO | - | <p>Channel [n] Software Request</p> <p>These bits generate software requests. Bit 0 corresponds to channel 0.</p> <p>Value Description</p> <table border="1"> <tr> <td>1</td> <td>Generate a software request for the corresponding channel.</td> </tr> <tr> <td>0</td> <td>No request generated.</td> </tr> </table> <p>These bits are automatically cleared when the software request has been completed.</p> | 1 | Generate a software request for the corresponding channel. | 0 | No request generated. |
| 1 | Generate a software request for the corresponding channel. | | | | | | | |
| 0 | No request generated. | | | | | | | |

Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

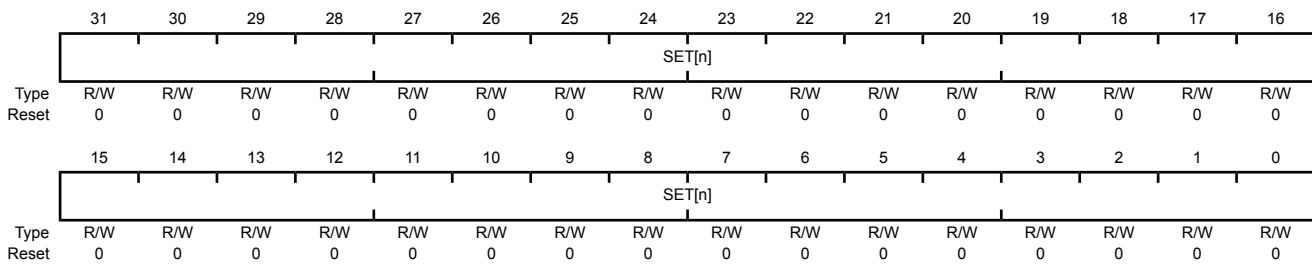
Each bit of the **DMAUSEBURSTSET** register represents the corresponding μ DMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

When there are fewer items remaining to transfer than the arbitration (burst) size, the μ DMA controller automatically clears the corresponding $SET[n]$ bit, allowing the remaining items to transfer using single requests. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to "Request Types" on page 261 for more details about request types.

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
 Offset 0x018
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------------|--------------------------|
| 31:0 | SET[n] | R/W | 0x0000.0000 | Channel [n] Useburst Set |

| Value | Description |
|-------|---|
| 0 | μ DMA channel [n] responds to single or burst requests. |
| 1 | μ DMA channel [n] responds only to burst requests. |

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding $CLR[n]$ bit in the **DMAUSEBURSTCLR** register.

Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C

Each bit of the **DMAUSEBURSTCLR** register represents the corresponding μ DMA channel. Setting a bit clears the corresponding **SET[n]** bit in the **DMAUSEBURSTSET** register.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000

Offset 0x01C

Type WO, reset -

| | | | | | | | | | | | | | | | | |
|-------|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | CLR[n] | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | CLR[n] | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|----------------------------|
| 31:0 | CLR[n] | WO | - | Channel [n] Useburst Clear |

Value Description

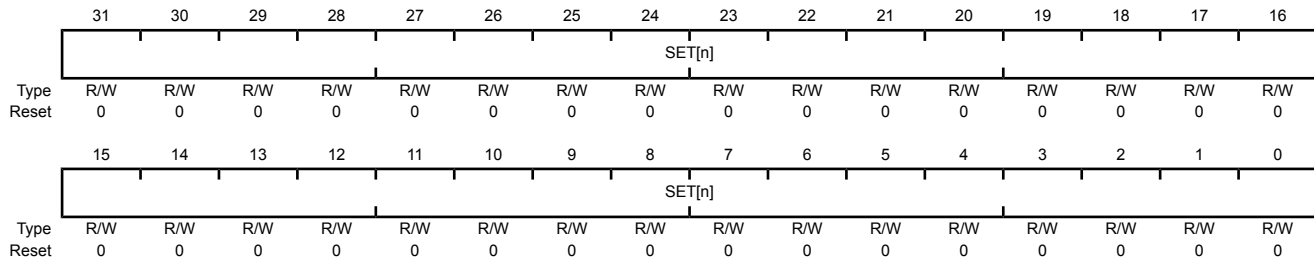
| | |
|---|--|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the DMAUSEBURSTSET register meaning that μ DMA channel [n] responds to single and burst requests. |

Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020

Each bit of the **DMAREQMASKSET** register represents the corresponding μ DMA channel. Setting a bit disables μ DMA requests for the channel. Reading the register returns the request mask status. When a μ DMA channel's request is masked, that means the peripheral can no longer request μ DMA transfers. The channel can then be used for software-initiated transfers.

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
 Offset 0x020
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------------|------------------------------|
| 31:0 | SET[n] | R/W | 0x0000.0000 | Channel [n] Request Mask Set |

| Value | Description |
|-------|--|
| 0 | The peripheral associated with channel [n] is enabled to request μ DMA transfers. |
| 1 | The peripheral associated with channel [n] is not able to request μ DMA transfers. Channel [n] may be used for software-initiated transfers. |

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAREQMASKCLR** register.

Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

Each bit of the **DMAREQMASKCLR** register represents the corresponding μ DMA channel. Setting a bit clears the corresponding **SET[n]** bit in the **DMAREQMASKSET** register.

DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000

Offset 0x024

Type WO, reset -



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--------------------------------|
| 31:0 | CLR[n] | WO | - | Channel [n] Request Mask Clear |

Value Description

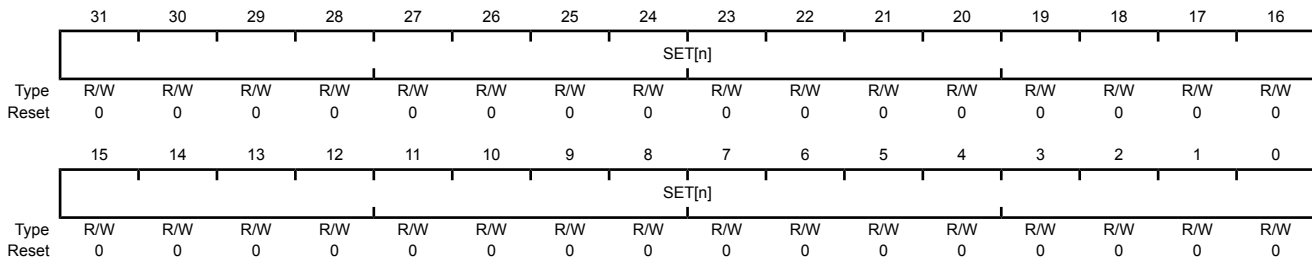
| Value | Description |
|-------|--|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the DMAREQMASKSET register meaning that the peripheral associated with channel [n] is enabled to request μ DMA transfers. |

Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding μ DMA channel. Setting a bit enables the corresponding μ DMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
 Offset 0x028
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------------|------------------------|
| 31:0 | SET[n] | R/W | 0x0000.0000 | Channel [n] Enable Set |

| Value | Description |
|-------|------------------------------------|
| 0 | μ DMA Channel [n] is disabled. |
| 1 | μ DMA Channel [n] is enabled. |

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAENACLR** register.

Register 15: DMA Channel Enable Clear (DMAENACLRL), offset 0x02C

Each bit of the **DMAENACLRL** register represents the corresponding μ DMA channel. Setting a bit clears the corresponding **SET[n]** bit in the **DMAENASET** register.

DMA Channel Enable Clear (DMAENACLRL)

Base 0x400F.F000

Offset 0x02C

Type WO, reset -

| | | | | | | | | | | | | | | | | |
|-------|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | CLR[n] | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | CLR[n] | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--------------------------------|
| 31:0 | CLR[n] | WO | - | Clear Channel [n] Enable Clear |

| Value | Description |
|-------|---|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the DMAENASET register meaning that channel [n] is disabled for μ DMA transfers. |

Note: The controller disables a channel when it completes the μ DMA cycle.

Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding μ DMA channel. Setting a bit configures the μ DMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding μ DMA channel.

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
 Offset 0x030
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------------|---------------------------|
| 31:0 | SET[n] | R/W | 0x0000.0000 | Channel [n] Alternate Set |

| Value | Description |
|-------|---|
| 0 | μ DMA channel [n] is using the primary control structure. |
| 1 | μ DMA channel [n] is using the alternate control structure. |

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAALTCLR** register.

Note: For Ping-Pong and Scatter-Gather cycle types, the μ DMA controller automatically sets these bits to select the alternate channel control data structure.

Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

Each bit of the **DMAALTCLR** register represents the corresponding μ DMA channel. Setting a bit clears the corresponding **SET[n]** bit in the **DMAALTSET** register.

DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000

Offset 0x034

Type WO, reset -



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|-----------------------------|
| 31:0 | CLR[n] | WO | - | Channel [n] Alternate Clear |

| Value | Description |
|-------|--|
| 0 | No effect. |
| 1 | Setting a bit clears the corresponding SET[n] bit in the DMAALTSET register meaning that channel [n] is using the primary control structure. |

Note: For Ping-Pong and Scatter-Gather cycle types, the μ DMA controller automatically sets these bits to select the alternate channel control data structure.

Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the **DMAPRIOSET** register represents the corresponding μ DMA channel. Setting a bit configures the μ DMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
 Offset 0x038
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------------|--------------------------|
| 31:0 | SET[n] | R/W | 0x0000.0000 | Channel [n] Priority Set |

- Value Description
- 0 μ DMA channel [n] is using the default priority level.
 - 1 μ DMA channel [n] is using a high priority level.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAPRIOCLR** register.

Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

Each bit of the **DMAPRIOCLR** register represents the corresponding μ DMA channel. Setting a bit clears the corresponding $SET[n]$ bit in the **DMAPRIOSET** register.

DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000

Offset 0x03C

Type WO, reset -



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 31:0 | CLR[n] | WO | - | Channel [n] Priority Clear |
| | | | | Value Description |
| | | | | 0 No effect. |
| | | | | 1 Setting a bit clears the corresponding $SET[n]$ bit in the DMAPRIOSET register meaning that channel [n] is using the default priority level. |

Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the μ DMA bus error status. The error status is set if the μ DMA controller encountered a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the μ DMA controller. The other channels are unaffected.

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
 Offset 0x04C
 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 | | | | | | | | | | | | | | | ERRCLR | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | ERRCLR | R/W1C | 0 | μ DMA Bus Error Status |
| | | | | Value Description |
| | | | | 0 No bus error is pending. |
| | | | | 1 A bus error is pending. |
| | | | | This bit is cleared by writing a 1 to it. |

Register 21: DMA Channel Alternate Select (DMACHALT), offset 0x500

Each bit of the **DMACHALT** register represents the corresponding μ DMA channel. Setting a bit selects the alternate channel assignment as specified in Table 9-1 on page 260.

DMA Channel Alternate Select (DMACHALT)

Base 0x400F.F000

Offset 0x500

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | CHALT[n] | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | CHALT[n] | | | | | | | | | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

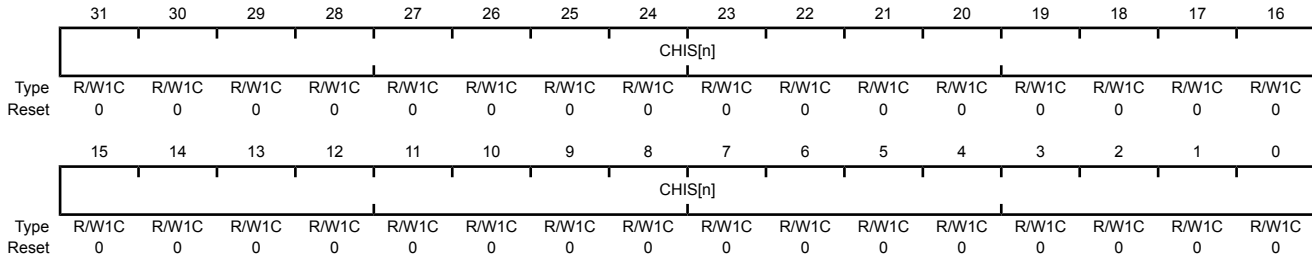
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:0 | CHALT[n] | R/W | - | Channel [n] Alternate Assignment Select |
| | | | | Value Description |
| | | | | 0 Use the primary channel assignment. |
| | | | | 1 Use the alternate channel assignment. |

Register 22: DMA Channel Interrupt Status (DMACHIS), offset 0x504

Each bit of the **DMACHIS** register represents the corresponding μ DMA channel. A bit is set when that μ DMA channel causes a completion interrupt. The bits are cleared by a writing a 1.

DMA Channel Interrupt Status (DMACHIS)

Base 0x400F.F000
 Offset 0x504
 Type R/W1C, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|-------|-------------|------------------------------|
| 31:0 | CHIS[n] | R/W1C | 0x0000.0000 | Channel [n] Interrupt Status |

Value Description

| | |
|---|--|
| 1 | The corresponding μ DMA channel caused an interrupt. |
| 0 | The corresponding μ DMA channel has not caused an interrupt. |

This bit is cleared by writing a 1 to it.

Register 23: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 0 (DMAPeriphID0)

Base 0x400F.F000

Offset 0xFE0

Type RO, reset 0x0000.0030

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID0 | | | | | | | |
| 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 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x30 | μ DMA Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. |

Register 24: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 1 (DMAPeriphID1)

Base 0x400F.F000
 Offset 0xFE4
 Type RO, reset 0x0000.00B2

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID1 | | | | | | | |
| 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 | 1 | 1 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0xB2 | μ DMA Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. |

Register 25: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000

Offset 0xFE8

Type RO, reset 0x0000.000B

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID2 | | | | | | | |
| 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 | 1 | 1 |

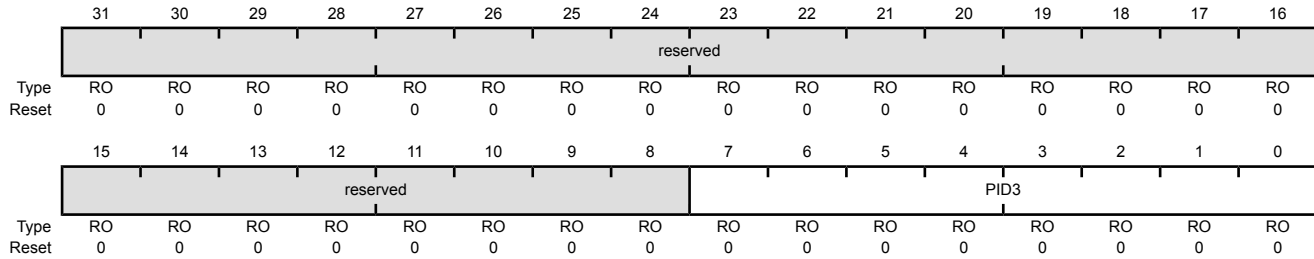
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x0B | μ DMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. |

Register 26: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

The DMAPeriphIDn registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 3 (DMAPeriphID3)

Base 0x400F.F000
 Offset 0xFEC
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x00 | μ DMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. |

Register 27: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 4 (DMAPeriphID4)

Base 0x400F.F000

Offset 0xFD0

Type RO, reset 0x0000.0004

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID4 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x04 | μ DMA Peripheral ID Register Can be used by software to identify the presence of this peripheral. |

Register 28: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 0 (DMAPCellID0)

Base 0x400F.F000
 Offset 0xFF0
 Type RO, reset 0x0000.000D



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | μ DMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system. |

Register 29: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 1 (DMAPCellID1)

Base 0x400F.F000

Offset 0xFF4

Type RO, reset 0x0000.00F0

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | CID1 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | <p>μDMA PrimeCell ID Register [15:8]</p> <p>Provides software a standard cross-peripheral identification system.</p> |

Register 30: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 2 (DMAPCellID2)

Base 0x400F.F000
 Offset 0xFF8
 Type RO, reset 0x0000.0005

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID2 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:8 | 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 | μ DMA PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system. |

Register 31: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 3 (DMAPCellID3)

Base 0x400F.F000

Offset 0xFFC

Type RO, reset 0x0000.00B1

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID3 | | | | | | | |
| 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 | 1 | 1 | 0 | 0 | 0 | 1 |

| 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 | <p>μDMA PrimeCell ID Register [31:24]</p> <p>Provides software a standard cross-peripheral identification system.</p> |

10 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of nine physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G, Port H, Port J). The GPIO module supports up to 60 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 60 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant input/outputs
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced Host Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
 - Interrupt generation masking
 - Edge-triggered on rising, falling, or both
 - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
 - Weak pull-up or pull-down resistors
 - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can 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

10.1 Signal Description

GPIO signals have alternate hardware functions. Table 10-2 on page 319 lists the GPIO pins and their analog and digital alternate functions. The A_{INx} and V_{REFA} analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding $AMSEL$ bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry ($C0-$, $C0+$, $C1-$, $C1+$, $C2-$, $C2+$,

USB0VBUS, USB0ID). These signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIO DEN)** register. The digital alternate hardware functions are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIO AFSEL)** and **GPIO DEN** registers and configuring the PMC_x bit field in the **GPIO Port Control (GPIO PCTL)** register to the numeric encoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIO AFSEL=0**, **GPIO DEN=0**, **GPIO PDR=0**, **GPIO PUR=0**, and **GPIO PCTL=0**) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 10-1. GPIO Pins With Non-Zero Reset Values

| GPIO Pins | Default State | GPIO AFSEL | GPIO DEN | GPIO PDR | GPIO PUR | GPIO PCTL |
|-----------|-------------------|------------|----------|----------|----------|-----------|
| PA[1:0] | UART0 | 1 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 1 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | I ² C0 | 1 | 1 | 0 | 0 | 0x1 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x3 |

Table 10-2. GPIO Pins and Alternate Functions

| IO | Pin | Analog Function | Digital Function (GPIO PCTL PMC _x Bit Field Encoding) ^a | | | | | | | | | | | |
|-----|-----|-----------------|---|------|--------------|------|--------|--------|------|----------|------------|------|----|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | |
| PA0 | 26 | - | U0Rx | - | - | - | - | - | - | - | I2C1SCL | U1Rx | - | - |
| PA1 | 27 | - | U0Tx | - | - | - | - | - | - | - | I2C1SDA | U1Tx | - | - |
| PA2 | 28 | - | SSI0Clk | - | - | - | - | - | - | - | I2S0RXSD | - | - | - |
| PA3 | 29 | - | SSI0Fss | - | - | - | - | - | - | - | I2S0RXMCLK | - | - | - |
| PA4 | 30 | - | SSI0Rx | - | - | - | CAN0Rx | - | - | - | I2S0TXSCK | - | - | - |
| PA5 | 31 | - | SSI0Tx | - | - | - | CAN0Tx | - | - | - | I2S0TXWS | - | - | - |
| PA6 | 34 | - | I2C1SCL | CCP1 | - | - | - | CAN0Rx | - | USB0EPEN | U1CTS | - | - | - |
| PA7 | 35 | - | I2C1SDA | CCP4 | - | - | - | CAN0Tx | CCP3 | USB0PFLT | U1DCD | - | - | - |
| PB0 | 66 | USB0ID | CCP0 | - | - | - | U1Rx | - | - | - | - | - | - | - |
| PB1 | 67 | USB0VBUS | CCP2 | - | - | CCP1 | U1Tx | - | - | - | - | - | - | - |
| PB2 | 72 | - | I2C0SCL | - | - | CCP3 | CCP0 | - | - | USB0EPEN | - | - | - | - |
| PB3 | 65 | - | I2C0SDA | - | - | - | - | - | - | USB0PFLT | - | - | - | - |
| PB4 | 92 | AIN10 C0- | - | - | - | U2Rx | CAN0Rx | - | U1Rx | EPI0S23 | - | - | - | - |
| PB5 | 91 | AIN11 C1- | C0o | CCP5 | CCP6 | CCP0 | CAN0Tx | CCP2 | U1Tx | EPI0S22 | - | - | - | - |
| PB6 | 90 | VREFA C0+ | CCP1 | CCP7 | C0o | - | - | CCP5 | - | - | I2S0TXSCK | - | - | - |
| PB7 | 89 | - | - | - | - | NMI | - | - | - | - | - | - | - | - |
| PC0 | 80 | - | - | - | TCK SWCLK | - | - | - | - | - | - | - | - | - |
| PC1 | 79 | - | - | - | TMS SWDIO | - | - | - | - | - | - | - | - | - |
| PC2 | 78 | - | - | - | TDI | - | - | - | - | - | - | - | - | - |

Table 10-2. GPIO Pins and Alternate Functions (continued)

| IO | Pin | Analog Function | Digital Function (GPIOCTL PMCx Bit Field Encoding) ^a | | | | | | | | | | |
|-----|-----|-----------------|---|---------|---------|----------|------|----------|----------|-----------|----------|---------|---------|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PC3 | 77 | - | - | - | TDO SWO | - | - | - | - | - | - | - | - |
| PC4 | 25 | - | CCP5 | - | - | - | CCP2 | CCP4 | - | EPI0S2 | CCP1 | - | - |
| PC5 | 24 | C1+ | CCP1 | C1o | C0o | - | CCP3 | USB0EPEN | - | EPI0S3 | - | - | - |
| PC6 | 23 | C2+ | CCP3 | - | C2o | - | U1Rx | CCP0 | USB0PFLT | EPI0S4 | - | - | - |
| PC7 | 22 | C2- | CCP4 | - | - | CCP0 | U1Tx | USB0PFLT | C1o | EPI0S5 | - | - | - |
| PD0 | 10 | AIN15 | - | CAN0Rx | - | U2Rx | U1Rx | CCP6 | - | I2S0RXSCK | U1CTS | - | - |
| PD1 | 11 | AIN14 | - | CAN0Tx | - | U2Tx | U1Tx | CCP7 | - | I2S0RXWS | U1DCD | CCP2 | - |
| PD2 | 12 | AIN13 | U1Rx | CCP6 | - | CCP5 | - | - | - | EPI0S20 | - | - | - |
| PD3 | 13 | AIN12 | U1Tx | CCP7 | - | CCP0 | - | - | - | EPI0S21 | - | - | - |
| PD4 | 97 | AIN7 | CCP0 | CCP3 | - | - | - | - | - | I2S0RXSD | U1RI | EPI0S19 | - |
| PD5 | 98 | AIN6 | CCP2 | CCP4 | - | - | - | - | - | I2S0RXCLK | U2Rx | EPI0S28 | - |
| PD6 | 99 | AIN5 | - | - | - | - | - | - | - | I2S0TXSCK | U2Tx | EPI0S29 | - |
| PD7 | 100 | AIN4 | - | C0o | CCP1 | - | - | - | - | I2S0TXWS | U1DTR | EPI0S30 | - |
| PE0 | 74 | - | - | SSI1Clk | CCP3 | - | - | - | - | EPI0S8 | USB0PFLT | - | - |
| PE1 | 75 | - | - | SSI1Fss | - | CCP2 | CCP6 | - | - | EPI0S9 | - | - | - |
| PE2 | 95 | AIN9 | CCP4 | SSI1Rx | - | - | CCP2 | - | - | EPI0S24 | - | - | - |
| PE3 | 96 | AIN8 | CCP1 | SSI1Tx | - | - | CCP7 | - | - | EPI0S25 | - | - | - |
| PE4 | 6 | AIN3 | CCP3 | - | - | - | U2Tx | CCP2 | - | - | I2S0TXWS | - | - |
| PE5 | 5 | AIN2 | CCP5 | - | - | - | - | - | - | - | I2S0TXSD | - | - |
| PE6 | 2 | AIN1 | - | C1o | - | - | - | - | - | - | U1CTS | - | - |
| PE7 | 1 | AIN0 | - | C2o | - | - | - | - | - | - | U1DCD | - | - |
| PF0 | 47 | - | CAN1Rx | - | - | - | - | - | - | I2S0TXSD | U1DSR | - | - |
| PF1 | 61 | - | CAN1Tx | - | - | - | - | - | - | I2S0TXCLK | U1RTS | CCP3 | - |
| PF2 | 60 | - | LED1 | - | - | - | - | - | - | - | SSI1Clk | - | - |
| PF3 | 59 | - | LED0 | - | - | - | - | - | - | - | SSI1Fss | - | - |
| PF4 | 42 | - | CCP0 | C0o | - | - | - | - | - | EPI0S12 | SSI1Rx | - | - |
| PF5 | 41 | - | CCP2 | C1o | - | - | - | - | - | EPI0S15 | SSI1Tx | - | - |
| PG0 | 19 | - | U2Rx | - | I2C1SCL | - | - | - | USB0EPEN | EPI0S13 | - | - | - |
| PG1 | 18 | - | U2Tx | - | I2C1SDA | - | - | - | - | EPI0S14 | - | - | - |
| PG7 | 36 | - | - | - | - | - | - | - | - | CCP5 | EPI0S31 | - | - |
| PH0 | 86 | - | CCP6 | - | - | - | - | - | - | EPI0S6 | - | - | - |
| PH1 | 85 | - | CCP7 | - | - | - | - | - | - | EPI0S7 | - | - | - |
| PH2 | 84 | - | - | C1o | - | - | - | - | - | EPI0S1 | - | - | - |
| PH3 | 83 | - | - | - | - | USB0EPEN | - | - | - | EPI0S0 | - | - | - |
| PH4 | 76 | - | - | - | - | USB0PFLT | - | - | - | EPI0S10 | - | - | SSI1Clk |
| PH5 | 63 | - | - | - | - | - | - | - | - | EPI0S11 | - | - | SSI1Fss |
| PH6 | 62 | - | - | - | - | - | - | - | - | EPI0S26 | - | - | SSI1Rx |
| PH7 | 15 | - | - | - | - | - | - | - | - | EPI0S27 | - | - | SSI1Tx |
| PJ0 | 14 | - | - | - | - | - | - | - | - | EPI0S16 | - | - | I2C1SCL |
| PJ1 | 87 | - | - | - | - | - | - | - | - | EPI0S17 | USB0PFLT | - | I2C1SDA |

Table 10-2. GPIO Pins and Alternate Functions (*continued*)

| IO | Pin | Analog Function | Digital Function (GPIOCTL PMCx Bit Field Encoding) ^a | | | | | | | | | | | |
|-----|-----|-----------------|---|---|---|---|---|---|---|---|---------|------|----|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | |
| PJ2 | 39 | - | - | - | - | - | - | - | - | - | EPI0S18 | CCP0 | - | - |

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

10.2 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 10-1 on page 321 and Figure 10-2 on page 322). The LM3S9B90 microcontroller contains nine ports and thus nine of these physical GPIO blocks. Note that not all pins may be implemented on every block. Some GPIO pins can function as I/O signals for the on-chip peripheral modules. For information on which GPIO pins are used for alternate hardware functions, refer to Table 24-5 on page 1035.

Figure 10-1. Digital I/O Pads

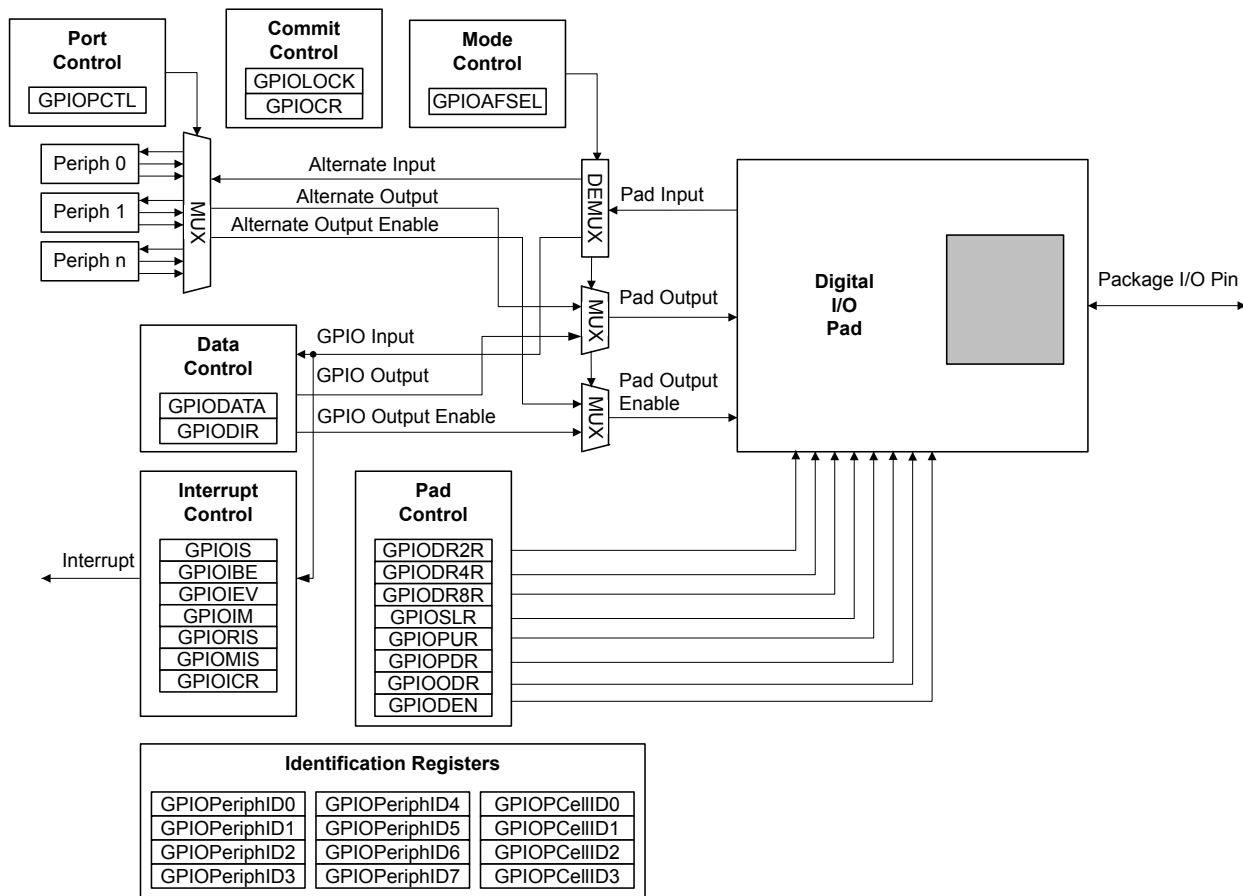
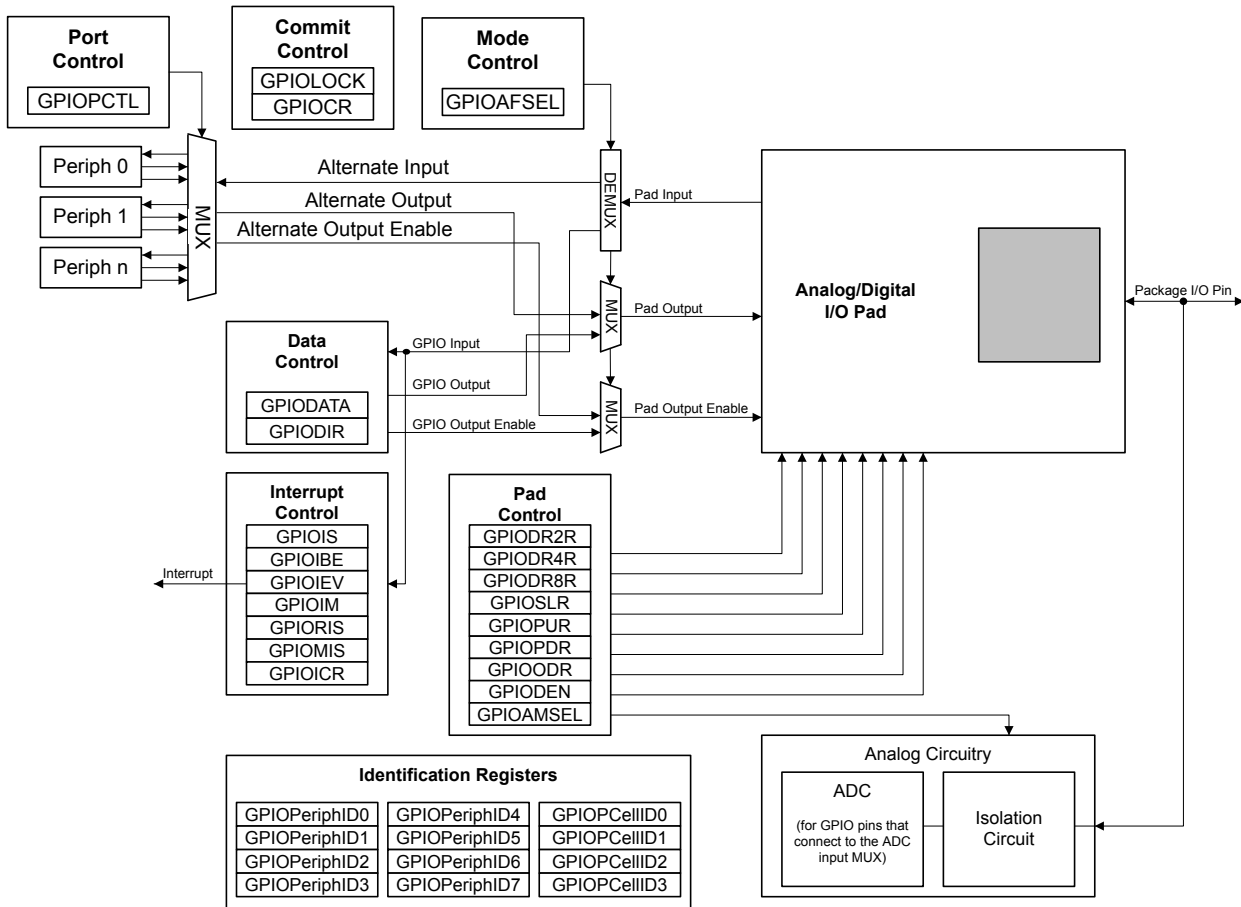


Figure 10-2. Analog/Digital I/O Pads



10.2.1 Data Control

The data control registers allow software to configure the operational modes of the GPIOs. The data direction register configures the GPIO as an input or an output while the data register either captures incoming data or drives it out to the pads.

Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris® microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.

10.2.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 331) is used to configure each individual pin as an input or output. When the data direction bit is cleared, the GPIO is configured as an input, and the corresponding data register bit captures and stores the value on the GPIO port. When the data direction bit is set, the GPIO is configured as an output, and the corresponding data register bit is driven out on the GPIO port.

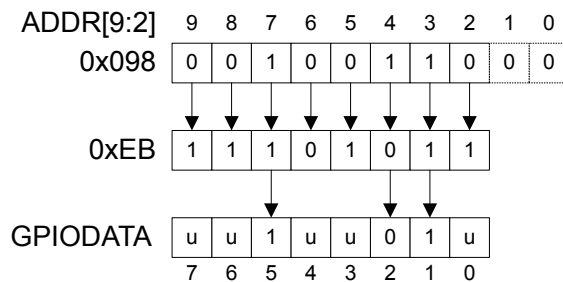
10.2.1.2 Data Register Operation

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 330) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the **GPIODATA** register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set, the value of the **GPIODATA** register is altered. If the address bit is cleared, the data bit is left unchanged.

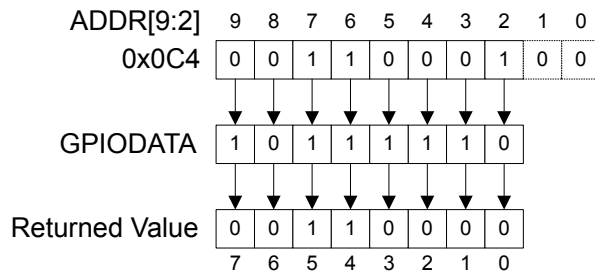
For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 10-3, where u indicates that data is unchanged by the write.

Figure 10-3. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 10-4.

Figure 10-4. GPIODATA Read Example



10.2.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. These registers are used to select the source of the interrupt, its polarity, and the edge properties. When one or more GPIO inputs cause an interrupt, a single interrupt output is sent to the interrupt controller for the entire GPIO port. For edge-triggered interrupts, software must clear the interrupt to enable any further interrupts. For a level-sensitive interrupt, the external source must hold the level constant for the interrupt to be recognized by the controller.

Three registers define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 332)

- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 333)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 334)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 335).

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 336 and page 337). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the interrupt controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the interrupt controller.

In addition to providing GPIO functionality, $PB4$ can also be used as an external trigger for the ADC. If $PB4$ is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), an interrupt for Port B is generated, and an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 530.

If no other Port B pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETENA) register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on $PB4$ and wait for the ADC interrupt, or the ADC interrupt must be disabled in the SETENA register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See the *ARM® Cortex™-M3 Technical Reference Manual* for more information.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 339).

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

10.2.3 Mode Control

The GPIO pins can be controlled by either software or hardware. Software control is the default for most signals and corresponds to the GPIO mode, where the **GPIO DATA** register is used to read or write the corresponding pins. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 340), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the **GPIO Port Control (GPIOCTL)** register which selects one of several peripheral functions for each GPIO. For information on the configuration options, refer to Table 24-5 on page 1035.

Note: If any pin is to be used as an ADC input, the appropriate bit in the **GPIOAMSEL** register must be set to disable the analog isolation circuit.

10.2.4 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 **NMI** pin ($PB7$) and the four **JTAG/SWD** pins ($PC[3:0]$). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 340), **GPIO Pull Up Select (GPIOPUR)** register (see page 346), **GPIO Pull-Down Select (GPIOPDR)** register (see page 348), and **GPIO Digital Enable (GPIODEN)** register (see page 351) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register

(see page 353) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 354) have been set.

10.2.5 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable for each GPIO.

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

10.2.6 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.

10.3 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris® parts. The other aperture, the Advanced Host Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHBCTL** register (see page 127).

To use the pins in a particular GPIO port, the clock for the port must be enabled by setting the appropriate GPIO Port bit field ($GPIO_n$) in the **RCGC2** register (see page 184).

On reset, all GPIO pins are configured out of reset to be undriven (tristate): **GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**, except for the pins shown in Table 10-1 on page 319. Table 10-3 on page 325 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 10-4 on page 326 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

Table 10-3. 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 ² C) | 1 | X | 1 | 1 | X | X | ? | ? | ? | ? |
| Digital Input (Timer CCP) | 1 | X | 0 | 1 | ? | ? | X | X | X | X |

Table 10-3. GPIO Pad Configuration Examples (continued)

| Configuration | GPIO Register Bit Value ^a | | | | | | | | | |
|-----------------------------|--------------------------------------|-----|-----|-----|-----|-----|------|------|------|-----|
| | AFSEL | DIR | ODR | DEN | PUR | PDR | DR2R | DR4R | DR8R | SLR |
| Digital Output (Timer PWM) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? |
| Digital Input/Output (SSI) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? |
| Digital Input/Output (UART) | 1 | X | 0 | 1 | ? | ? | ? | ? | ? | ? |
| 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 10-4. 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 |
| GPIOIBE | 0=single edge 1=both edges | X | X | X | X | X | 0 | X | X |
| GPIOIEV | 0=Low level, or falling edge 1=High level, or rising 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)

10.4 Register Map

Table 10-6 on page 328 lists the GPIO registers. Each GPIO port can be accessed through one of two bus apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris[®] parts. The other aperture, the Advanced Host Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus.

Important: The GPIO registers in this chapter are duplicated in each GPIO block; however, depending on the block, all eight bits may not be connected to a GPIO pad. In those cases, writing to unconnected bits has no effect, and reading unconnected bits returns no meaningful data.

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (APB): 0x4002.4000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (APB): 0x4002.5000
- GPIO Port F (AHB): 0x4005.D000
- GPIO Port G (APB): 0x4002.6000
- GPIO Port G (AHB): 0x4005.E000
- GPIO Port H (APB): 0x4002.7000
- GPIO Port H (AHB): 0x4005.F000
- GPIO Port J (APB): 0x4003.D000
- GPIO Port J (AHB): 0x4006.0000

Note that each GPIO module clock must be enabled before the registers can be programmed (see page 184).

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAENSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, **GPIOPUR=0**, and **GPIOPCTL=0**) with the exception of the pins shown in the table below. A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 10-5. GPIO Pins With Non-Zero Reset Values

| GPIO Pins | Default State | GPIOAENSEL | 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 ² C0 | 1 | 1 | 0 | 0 | 0x1 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x3 |

Note: The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the **NMI** pin and the four JTAG/SWD pins (**PB7** and **PC[3:0]**). These five pins are 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 **NMI** pin and the four JTAG/SWD pins (**PB7** and **PC[3:0]**). To ensure that the JTAG port is not accidentally programmed as a GPIO, these four pins default to non-committable. To ensure that the **NMI** pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of **GPIOCR** for Port C is 0x0000.00F0.

Table 10-6. GPIO Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|----------------|------|-------------|----------------------------------|----------|
| 0x000 | GPIODATA | R/W | 0x0000.0000 | GPIO Data | 330 |
| 0x400 | GPIODIR | R/W | 0x0000.0000 | GPIO Direction | 331 |
| 0x404 | GPIOIS | R/W | 0x0000.0000 | GPIO Interrupt Sense | 332 |
| 0x408 | GPIOIBE | R/W | 0x0000.0000 | GPIO Interrupt Both Edges | 333 |
| 0x40C | GPIOIEV | R/W | 0x0000.0000 | GPIO Interrupt Event | 334 |
| 0x410 | GPIOIM | R/W | 0x0000.0000 | GPIO Interrupt Mask | 335 |
| 0x414 | GPIORIS | RO | 0x0000.0000 | GPIO Raw Interrupt Status | 336 |
| 0x418 | GPIONIS | RO | 0x0000.0000 | GPIO Masked Interrupt Status | 337 |
| 0x41C | GPIOICR | W1C | 0x0000.0000 | GPIO Interrupt Clear | 339 |
| 0x420 | GPIOAFSEL | R/W | - | GPIO Alternate Function Select | 340 |
| 0x500 | GPIODR2R | R/W | 0x0000.00FF | GPIO 2-mA Drive Select | 342 |
| 0x504 | GPIODR4R | R/W | 0x0000.0000 | GPIO 4-mA Drive Select | 343 |
| 0x508 | GPIODR8R | R/W | 0x0000.0000 | GPIO 8-mA Drive Select | 344 |
| 0x50C | GPIOODR | R/W | 0x0000.0000 | GPIO Open Drain Select | 345 |
| 0x510 | GPIONPUR | R/W | - | GPIO Pull-Up Select | 346 |
| 0x514 | GPIONPDR | R/W | 0x0000.0000 | GPIO Pull-Down Select | 348 |
| 0x518 | GPIONSLR | R/W | 0x0000.0000 | GPIO Slew Rate Control Select | 350 |
| 0x51C | GPIONDEN | R/W | - | GPIO Digital Enable | 351 |
| 0x520 | GPIOLOCK | R/W | 0x0000.0001 | GPIO Lock | 353 |
| 0x524 | GPIOCR | - | - | GPIO Commit | 354 |
| 0x528 | GPIOAMSEL | R/W | 0x0000.0000 | GPIO Analog Mode Select | 356 |
| 0x52C | GPIONPCTL | R/W | - | GPIO Port Control | 358 |
| 0xFD0 | GPIONPeriphID4 | RO | 0x0000.0000 | GPIO Peripheral Identification 4 | 360 |
| 0xFD4 | GPIONPeriphID5 | RO | 0x0000.0000 | GPIO Peripheral Identification 5 | 361 |
| 0xFD8 | GPIONPeriphID6 | RO | 0x0000.0000 | GPIO Peripheral Identification 6 | 362 |
| 0xFDC | GPIONPeriphID7 | RO | 0x0000.0000 | GPIO Peripheral Identification 7 | 363 |
| 0xFE0 | GPIONPeriphID0 | RO | 0x0000.0061 | GPIO Peripheral Identification 0 | 364 |
| 0xFE4 | GPIONPeriphID1 | RO | 0x0000.0000 | GPIO Peripheral Identification 1 | 365 |
| 0xFE8 | GPIONPeriphID2 | RO | 0x0000.0018 | GPIO Peripheral Identification 2 | 366 |
| 0xFEC | GPIONPeriphID3 | RO | 0x0000.0001 | GPIO Peripheral Identification 3 | 367 |
| 0xFF0 | GPIONPCellID0 | RO | 0x0000.000D | GPIO PrimeCell Identification 0 | 368 |
| 0xFF4 | GPIONPCellID1 | RO | 0x0000.00F0 | GPIO PrimeCell Identification 1 | 369 |

Table 10-6. GPIO Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-------------|------|-------------|---------------------------------|----------|
| 0xFF8 | GPIOCellID2 | RO | 0x0000.0005 | GPIO PrimeCell Identification 2 | 370 |
| 0xFFC | GPIOCellID3 | RO | 0x0000.00B1 | GPIO PrimeCell Identification 3 | 371 |

10.5 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 331).

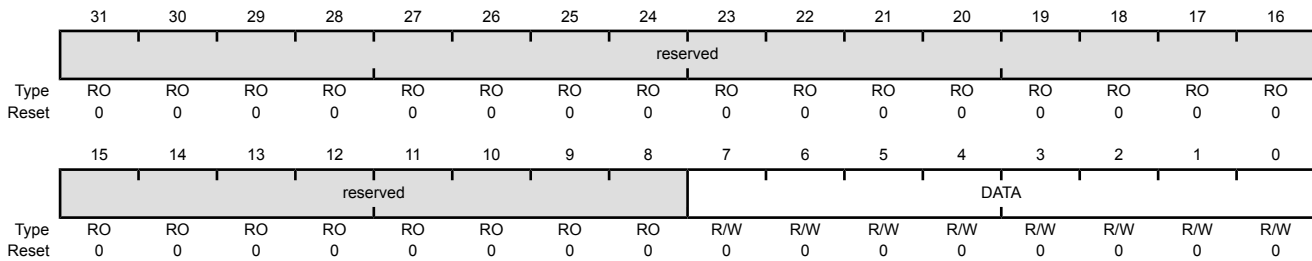
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set. Otherwise, the bit values remain unchanged by the write.

Similarly, the values read from this register are determined for each bit by the mask bit derived from the address used to access the data register, bits [9:2]. Bits that are set in the address mask cause the corresponding bits in **GPIODATA** to be read, and bits that are clear in the address mask cause the corresponding bits in **GPIODATA** to be read as 0, regardless of their value.

A read from **GPIODATA** returns the last bit value written if the respective pins are configured as outputs, or it returns the value on the corresponding input pin when these are configured as inputs. All bits are cleared by a reset.

GPIO Data (GPIODATA)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x000
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | R/W | 0x00 | GPIO Data |

This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See “Data Register Operation” on page 323 for examples of reads and writes.

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

The **GPIODIR** register is the data direction register. Setting a bit in the **GPIODIR** register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x400
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | DIR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

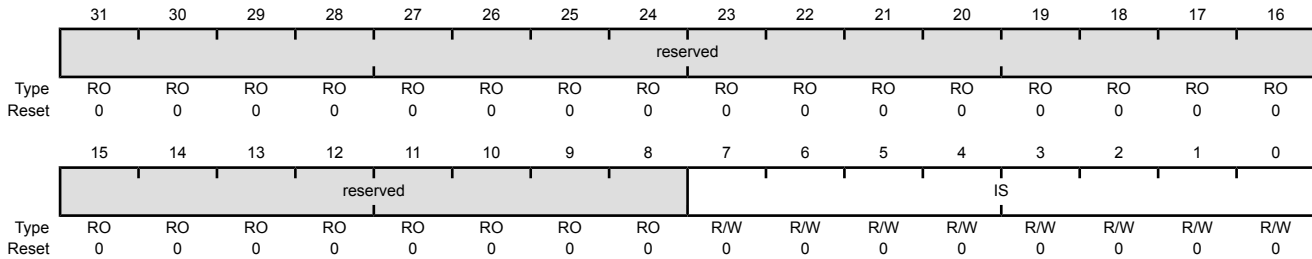
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DIR | R/W | 0x00 | GPIO Data Direction |
| | | | | Value Description |
| | | | | 0 Corresponding pin is an input. |
| | | | | 1 Corresponding pins is an output. |

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

The **GPIOIS** register is the interrupt sense register. Setting a bit in the **GPIOIS** register configures the corresponding pin to detect levels, while clearing a bit configures the corresponding pin to detect edges. All bits are cleared by a reset.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x404
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description | |
|-------------------|----------|---|-----------|---|--|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | |
| 7:0 | IS | R/W | 0x00 | GPIO Interrupt Sense | |
| Value Description | | | | | |
| | 0 | The edge on the corresponding pin is detected (edge-sensitive). | | | |
| | 1 | The level on the corresponding pin is detected (level-sensitive). | | | |

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

The **GPIOIBE** register allows both edges to cause interrupts. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 332) is set to detect edges, setting a bit in the **GPIOIBE** register configures the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 334). Clearing a bit configures the pin to be controlled by the **GPIOIEV** register. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x408
 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 | | | | | | | | IBE | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

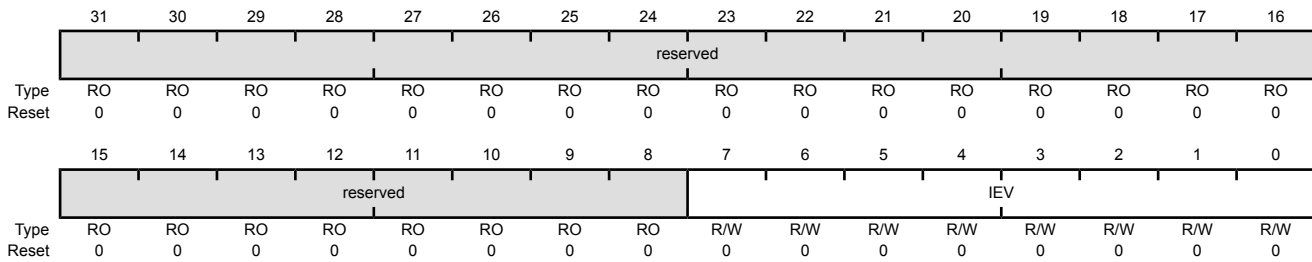
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | IBE | R/W | 0x00 | GPIO Interrupt Both Edges |
| | | | | Value Description |
| | | | | 0 Interrupt generation is controlled by the GPIO Interrupt Event (GPIOIEV) register (see page 334). |
| | | | | 1 Both edges on the corresponding pin trigger an interrupt. |

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

The **GPIOIEV** register is the interrupt event register. Setting a bit in the **GPIOIEV** register configures the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 332). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in the **GPIOIS** register. All bits are cleared by a reset.

GPIO Interrupt Event (GPIOIEV)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x40C
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | IEV | R/W | 0x00 | GPIO Interrupt Event |
| | | | | Value Description |
| | | | | 0 A falling edge or a Low level on the corresponding pin triggers an interrupt. |
| | | | | 1 A rising edge or a High level on the corresponding pin triggers an interrupt. |

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

The **GPIOIM** register is the interrupt mask register. Setting a bit in the **GPIOIM** register allows interrupts that are generated by the corresponding pin to be sent to the interrupt controller on the combined interrupt signal. Clearing a bit prevents an interrupt on the corresponding pin from being sent to the interrupt controller. All bits are cleared by a reset.

GPIO Interrupt Mask (GPIOIM)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x410
 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 | | | | | | | | IME | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | IME | R/W | 0x00 | GPIO Interrupt Mask Enable |

Value Description

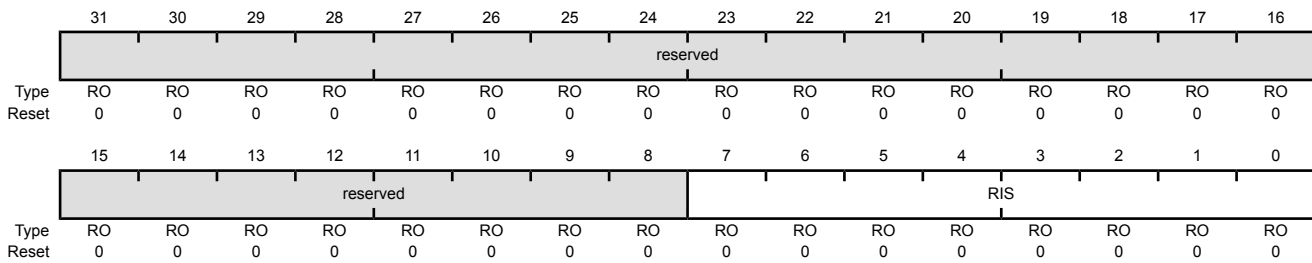
| Value | Description |
|-------|---|
| 0 | The interrupt from the corresponding pin is masked. |
| 1 | The interrupt from the corresponding pin is sent to the interrupt controller. |

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

The **GPIORIS** register is the raw interrupt status register. A bit in this register is set when an interrupt condition occurs on the corresponding GPIO pin. If the corresponding bit in the **GPIO Interrupt Mask (GPIOIM)** register (see page 335) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. A bit in this register can be cleared by writing a 1 to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x414
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | RIS | RO | 0x00 | GPIO Interrupt Raw Status |

| Value | Description |
|-------|---|
| 1 | An interrupt condition has occurred on the corresponding pin. |
| 0 | An interrupt condition has not occurred on the corresponding pin. |

A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

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

The **GPIOMIS** register is the masked interrupt status register. If a bit is set in this register, the corresponding interrupt has triggered an interrupt to the interrupt controller. If a bit is clear, either no interrupt has been generated, or the interrupt is masked.

In addition to providing GPIO functionality, **PB4** can also be used as an external trigger for the ADC. If **PB4** is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), an interrupt for Port B is generated, and an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 530.

If no other Port B pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the SETNA register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See the *ARM® Cortex™-M3 Technical Reference Manual* for more information.

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x418
 Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|----------|----|----|----|----|----|----|----|-----|----|----|----|----|----|----|----|
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | MIS | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 7:0 | MIS | RO | 0x00 | GPIO Masked Interrupt Status |
| | | | | Value Description |
| | | | | 1 An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller. |
| | | | | 0 An interrupt condition on the corresponding pin is masked or has not occurred. |
| | | | | A bit is cleared by writing a 1 to the corresponding bit in the GPIOICR register. |

Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C

The **GPIOICR** register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt bit in the **GPIOIRIS** and **GPIOMIS** registers. Writing a 0 has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x41C
 Type 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 | | | | | | | | IC | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 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 | IC | W1C | 0x00 | GPIO Interrupt Clear |
| | | | | Value Description |
| | | | | 1 The corresponding interrupt is cleared. |
| | | | | 0 The corresponding interrupt is unaffected. |

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

The **GPIOAFSEL** register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The **GPIO Port Control (GPIOCTL)** register is used to select one of the possible functions. Table 24-5 on page 1035 details which functions are muxed on each GPIO pin. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset (\overline{POR}) or asserting \overline{RST} puts the pins back to their default state.

Table 10-7. GPIO Pins With Non-Zero Reset Values

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

Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris[®] microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the \overline{NMI} pin ($\overline{PB7}$) and the four JTAG/SWD pins ($\overline{PC[3:0]}$). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 340), **GPIO Pull Up Select (GPIOPUR)** register (see page 346), **GPIO Pull-Down Select (GPIOPDR)** register (see page 348), and **GPIO Digital Enable (GPIODEN)** register (see page 351) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 353) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 354) have been set.

When using the I²C module, in addition to setting the **GPIOAFSEL** register bits for the I²C clock and data pins, the pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register (see examples in “Initialization and Configuration” on page 325).

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000

Offset 0x420
 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 | | | | | | | | AFSEL | | | | | | | |
| 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 | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | AFSEL | R/W | - | GPIO Alternate Function Select |

Value Description

- 0 The associated pin functions as a GPIO and is controlled by the GPIO registers.
- 1 The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 319.

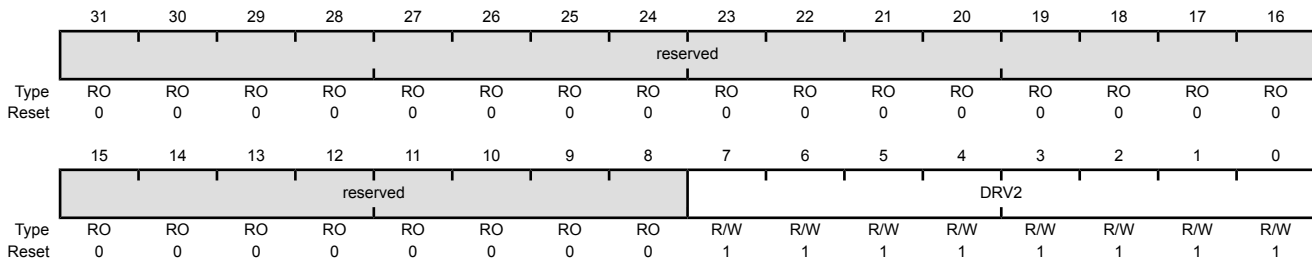
Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500

The **GPIODR2R** register is the 2-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the **DRV2** bit for a GPIO signal, the corresponding **DRV4** bit in the **GPIODR4R** register and **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware. By default, all GPIO pins have 2-mA drive.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x500

Type R/W, reset 0x0000.00FF



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DRV2 | R/W | 0xFF | Output Pad 2-mA Drive Enable |

| Value | Description |
|-------|--|
| 1 | The corresponding GPIO pin has 2-mA drive. |
| 0 | The drive for the corresponding GPIO pin is controlled by the GPIODR4R or GPIODR8R register. |

Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

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

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x504

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 | | | | | | | | DRV4 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DRV4 | R/W | 0x00 | Output Pad 4-mA Drive Enable |

Value Description

| Value | Description |
|-------|--|
| 1 | The corresponding GPIO pin has 4-mA drive. |
| 0 | The drive for the corresponding GPIO pin is controlled by the GPIODR2R or GPIODR8R register. |

Setting a bit in either the **GPIODR2** register or the **GPIODR8** register clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIODR8R** register is the 8-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the **DRV8** bit for a GPIO signal, the corresponding **DRV2** bit in the **GPIODR2R** register and **DRV4** bit in the **GPIODR4R** register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

Note: There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the V_{OH}/V_{OL} levels. See “Recommended DC Operating Conditions” on page 1039 for further information.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x508
 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 | | | | | | | | DRV8 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DRV8 | R/W | 0x00 | Output Pad 8-mA Drive Enable |

| Value | Description |
|-------|--|
| 1 | The corresponding GPIO pin has 8-mA drive. |
| 0 | The drive for the corresponding GPIO pin is controlled by the GPIODR2R or GPIODR4R register. |

Setting a bit in either the **GPIODR2** register or the **GPIODR4** register clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open-drain configuration of the corresponding GPIO pad. When open-drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 351). Corresponding bits in the drive strength and slew rate control registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an 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 I²C module, in addition to configuring the pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the I²C clock and data pins should be set (see examples in “Initialization and Configuration” on page 325).

GPIO Open Drain Select (GPIOODR)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x50C
 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 | | | | | | | | ODE | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | ODE | R/W | 0x00 | Output Pad Open Drain Enable |

Value Description

| | |
|---|--|
| 1 | The corresponding pin is configured as open drain. |
| 0 | The corresponding pin is not configured as open drain. |

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

The **GPIOPUR** register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 348). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are cleared prevent writes to the equivalent bit in this register.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset (\overline{POR}) or asserting \overline{RST} puts the pins back to their default state.

Table 10-8. GPIO Pins With Non-Zero Reset Values

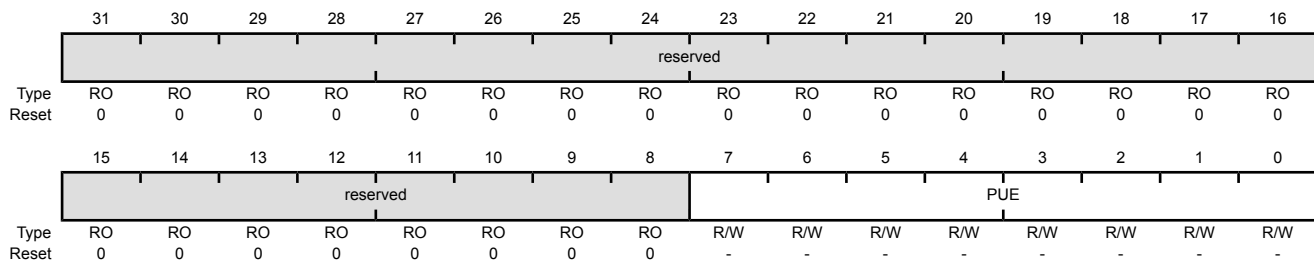
| GPIO Pins | Default State | GPIOAFSEL | GPIODEN | GPIOPDR | GPIOPUR | GPIOCTL |
|-----------|-------------------|-----------|---------|---------|---------|---------|
| PA[1:0] | UART0 | 1 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 1 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | I ² C0 | 1 | 1 | 0 | 0 | 0x1 |
| PC[3:0]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

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin (**PB7**) and the four **JTAG/SWD** pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 340), **GPIO Pull Up Select (GPIOPUR)** register (see page 346), **GPIO Pull-Down Select (GPIOPDR)** register (see page 348), and **GPIO Digital Enable (GPIODEN)** register (see page 351) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 353) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 354) have been set.

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000

Offset 0x510
 Type R/W, reset -



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PUE | R/W | - | Pad Weak Pull-Up Enable |

| Value | Description |
|-------|--|
| 1 | The corresponding pin has a weak pull-up resistor. |
| 0 | The corresponding pin is not affected. |

Setting a bit in the **GPIOPDR** register clears the corresponding bit in the **GPIOPUR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 319.

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

The **GPIOPDR** register is the pull-down control register. When a bit is set, a weak pull-down resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 346).

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL=0**, **GIODEN=0**, **GPIOPDR=0**, **GPIOPUR=0**, and **GPIOCTL=0**) with the exception of the pins shown in the table below. A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

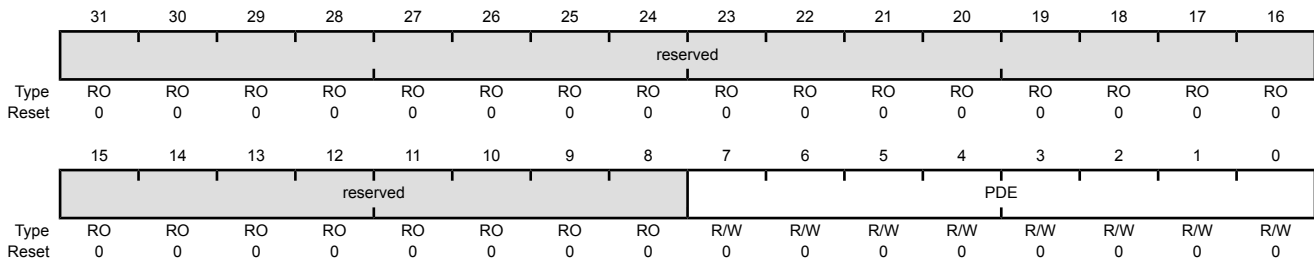
Table 10-9. GPIO Pins With Non-Zero Reset Values

| GPIO Pins | Default State | GPIOAFSEL | GIODEN | GPIOPDR | GPIOPUR | GPIOCTL |
|-----------|-------------------|-----------|--------|---------|---------|---------|
| PA[1:0] | UART0 | 1 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 1 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | I ² C0 | 1 | 1 | 0 | 0 | 0x1 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x3 |

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin (**PB7**) and the four **JTAG/SWD** pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 340), **GPIO Pull Up Select (GPIOPUR)** register (see page 346), **GPIO Pull-Down Select (GPIOPDR)** register (see page 348), and **GPIO Digital Enable (GIODEN)** register (see page 351) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 353) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 354) have been set.

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x514
 Type R/W, reset 0x0000.0000



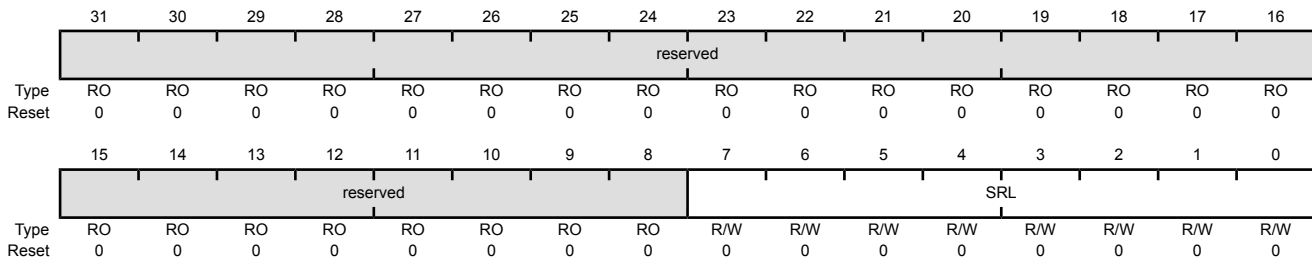
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|--|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PDE | R/W | 0x00 | Pad Weak Pull-Down Enable |
| | | | | <p>Value Description</p> <p>1 The corresponding pin has a weak pull-down resistor.</p> <p>0 The corresponding pin is not affected.</p> <p>Setting a bit in the GPIOPUR register clears the corresponding bit in the GPIOPDR register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.</p> |

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 344).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x518
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description | |
|-------------------|----------|--|-----------|---|--|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | |
| 7:0 | SRL | R/W | 0x00 | Slew Rate Limit Enable (8-mA drive only) | |
| Value Description | | | | | |
| | 1 | Slew rate control is enabled for the corresponding pin. | | | |
| | 0 | Slew rate control is disabled for the corresponding pin. | | | |

Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C

Note: Pins configured as digital inputs are Schmitt-triggered.

The **GPIODEN** register is the digital enable register. By default, all GPIO signals except those listed below are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding **GPIODEN** bit must be set.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset (\overline{POR}) or asserting \overline{RST} puts the pins back to their default state.

Table 10-10. 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 ² C0 | 1 | 1 | 0 | 0 | 0x1 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x3 |

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin (**PB7**) and the four JTAG/SWD pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 340), **GPIO Pull Up Select (GPIOPUR)** register (see page 346), **GPIO Pull-Down Select (GPIOPDR)** register (see page 348), and **GPIO Digital Enable (GPIODEN)** register (see page 351) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 353) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 354) have been set.

GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000

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 | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DEN | R/W | - | Digital Enable |

Value Description

- 0 The digital functions for the corresponding pin are disabled.
- 1 The digital functions for the corresponding pin are enabled.

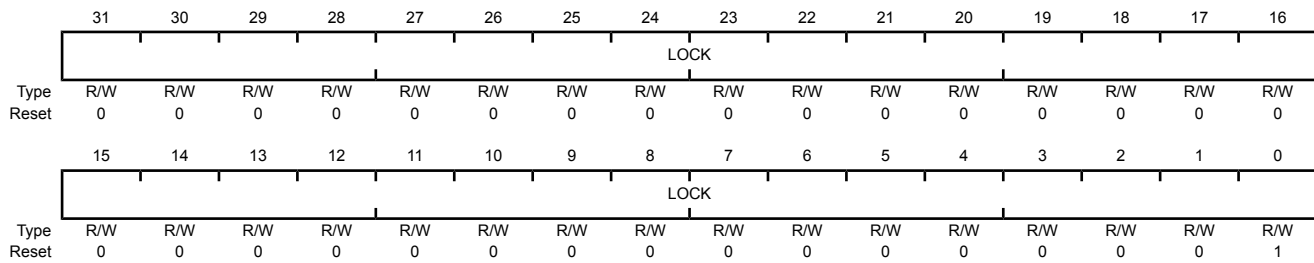
The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 319.

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

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 354). Writing 0x4C4F.434B to the **GPIOLOCK** register unlocks the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x0000.0001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x0000.0000.

GPIO Lock (GPIOLOCK)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x520
 Type R/W, reset 0x0000.0001



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

A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access. A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates.

A read of this register returns the following values:

| Value | Description |
|-------------|---|
| 0x0000.0001 | The GPIOCR register is locked and may not be modified. |
| 0x0000.0000 | The GPIOCR register is unlocked and may be modified. |

Register 20: GPIO Commit (GPIOCR), offset 0x524

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, and **GIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is cleared, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GIODEN** registers is committed to the register and reflects the new value.

The contents of the **GPIOCR** register can only be modified if the status in the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the status in the **GPIOLOCK** register is locked.

Important: This register is designed to prevent accidental programming of the registers that control connectivity to the NMI and JTAG/SWD debug hardware. By initializing the bits of the **GPIOCR** register to 0 for **PB7** and **PC[3:0]**, the NMI and JTAG/SWD debug port can only be converted to GPIOs through a deliberate set of writes to the **GPIOLOCK**, **GPIOCR**, and the corresponding registers.

Because this protection is currently only implemented on the NMI and JTAG/SWD pins on **PB7** and **PC[3:0]**, all of the other bits in the **GPIOCR** registers cannot be written with 0x0. These bits are hardwired to 0x1, ensuring that it is always possible to commit new values to the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GIODEN** register bits of these other pins.

GPIO Commit (GPIOCR)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000

Offset 0x524
 Type -, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | CR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | - | - | - | - | - | - | - | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CR | - | - | GPIO Commit |
| | | | | Value Description |
| | | | | 1 The corresponding GPIOAFSEL , GPIOPUR , GPIOPDR , or GIODEN bits can be written. |
| | | | | 0 The corresponding GPIOAFSEL , GPIOPUR , GPIOPDR , or GIODEN bits cannot be written. |
| | | | | Note: The default register type for the GPIOCR register is RO for all GPIO pins with the exception of the NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). These five pins are 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 NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as a GPIO, these four pins default to non-committable. To ensure that the NMI pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of GPIOCR for GPIO Port B is 0x0000.007F while the default reset value of GPIOCR for Port C is 0x0000.00F0. |

Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

Important: This register is only valid for ports D and E; the corresponding base addresses for the remaining ports are not valid.

If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be set to disable the analog isolation circuit.

The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 5-V source and affect analog operation, analog circuitry requires isolation from the pins when they are not used in their analog function.

Each bit of this register controls the isolation circuitry for the corresponding GPIO signal. For information on which GPIO pins can be used for ADC functions, refer to Table 24-5 on page 1035.

GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0x528
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-----------|-----|-----|-----|----------|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| 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 | | | | | | | | GPIOAMSEL | | | | reserved | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------|------|-------|---|
| 7:4 | GPIOAMSEL | R/W | 0x0 | GPIO Analog Mode Select Value Description 1 The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions. 0 The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers. Note: This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad. The reset state of this register is 0 for all signals. |
| 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 22: GPIO Port Control (GPIOCTL), offset 0x52C

The **GPIOCTL** register is used in conjunction with the **GPIOAFSEL** register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the **GPIOAFSEL** register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the **GPIOAFSEL** register, the corresponding GPIO signal is controlled by an associated peripheral. The **GPIOCTL** register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the defined encodings for the bit fields in this register, refer to Table 24-5 on page 1035. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL=0**, **GIODEN=0**, **GPIOPDR=0**, **GPIOPUR=0**, and **GPIOCTL=0**) with the exception of the pins shown in the table below. A Power-On-Reset (\overline{POR}) or asserting \overline{RST} puts the pins back to their default state.

Table 10-11. GPIO Pins With Non-Zero Reset Values

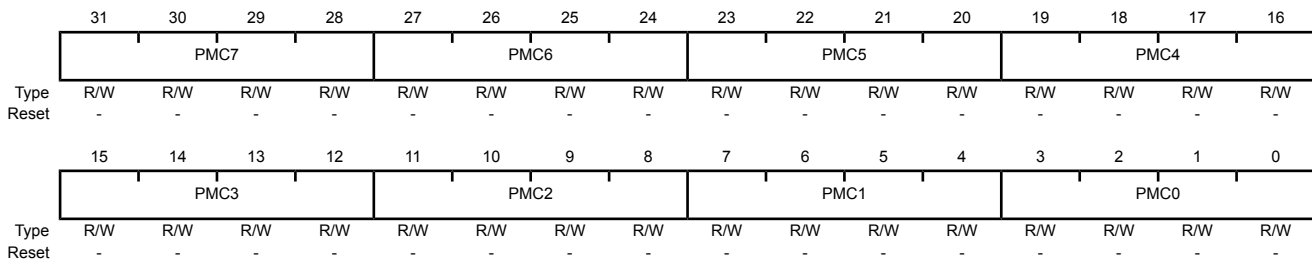
| GPIO Pins | Default State | GPIOAFSEL | GIODEN | GPIOPDR | GPIOPUR | GPIOCTL |
|-----------|-------------------|-----------|--------|---------|---------|---------|
| PA[1:0] | UART0 | 1 | 1 | 0 | 0 | 0x1 |
| PA[5:2] | SSI0 | 1 | 1 | 0 | 0 | 0x1 |
| PB[3:2] | I ² C0 | 1 | 1 | 0 | 0 | 0x1 |
| PC[3:0] | JTAG/SWD | 1 | 1 | 0 | 1 | 0x3 |

GPIO Port Control (GPIOCTL)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000

Offset 0x52C

Type R/W, reset -



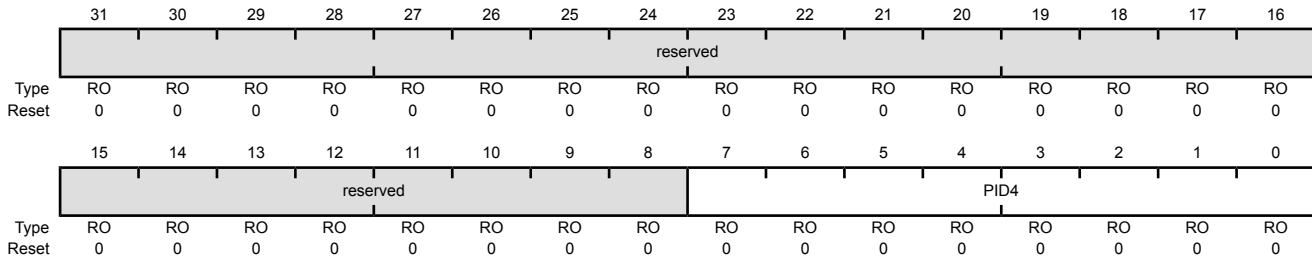
| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 31:28 | PMC7 | R/W | - | Port Mux Control 7 This field controls the configuration for GPIO pin 7. |
| 27:24 | PMC6 | R/W | - | Port Mux Control 6 This field controls the configuration for GPIO pin 6. |
| 23:20 | PMC5 | R/W | - | Port Mux Control 5 This field controls the configuration for GPIO pin 5. |
| 19:16 | PMC4 | R/W | - | Port Mux Control 4 This field controls the configuration for GPIO pin 4. |
| 15:12 | PMC3 | R/W | - | Port Mux Control 3 This field controls the configuration for GPIO pin 3. |
| 11:8 | PMC2 | R/W | - | Port Mux Control 2 This field controls the configuration for GPIO pin 2. |
| 7:4 | PMC1 | R/W | - | Port Mux Control 1 This field controls the configuration for GPIO pin 1. |
| 3:0 | PMC0 | R/W | - | Port Mux Control 0 This field controls the configuration for GPIO pin 0. |

Register 23: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 4 (GPIOPeriphID4)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFD0
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | GPIO Peripheral ID Register [7:0] |

Register 24: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 5 (GPIOPeriphID5)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFD4

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | PID5 | | | | | | | |
| 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 |

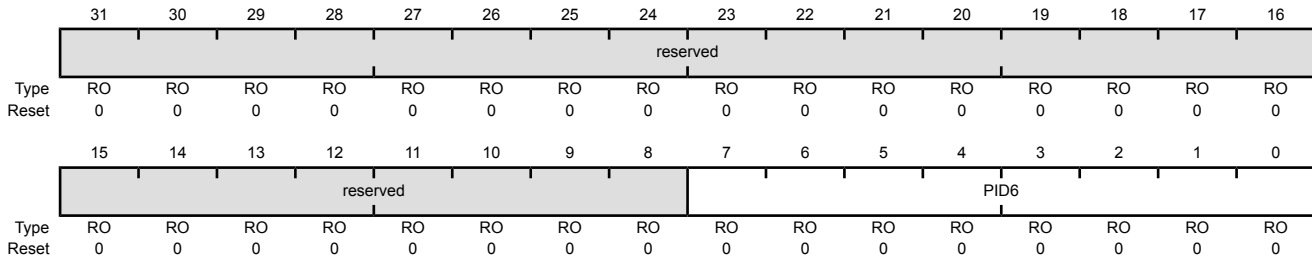
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | GPIO Peripheral ID Register [15:8] |

Register 25: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 6 (GPIOPeriphID6)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFD8
 Type RO, reset 0x0000.0000



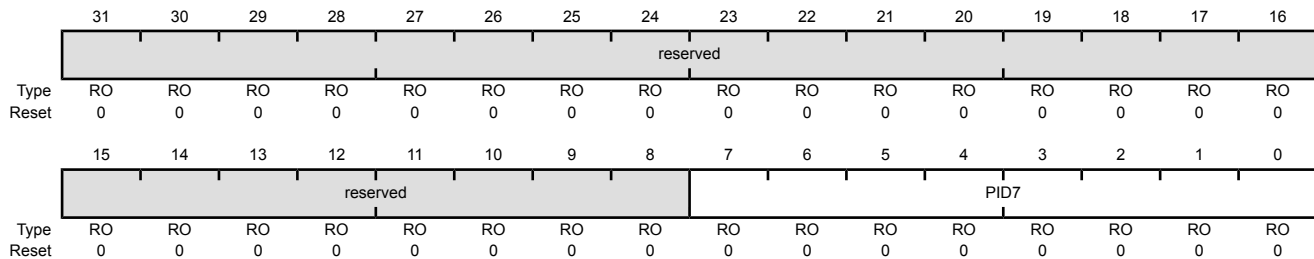
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | GPIO Peripheral ID Register [23:16] |

Register 26: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 7 (GPIOPeriphID7)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFDC
 Type RO, reset 0x0000.0000



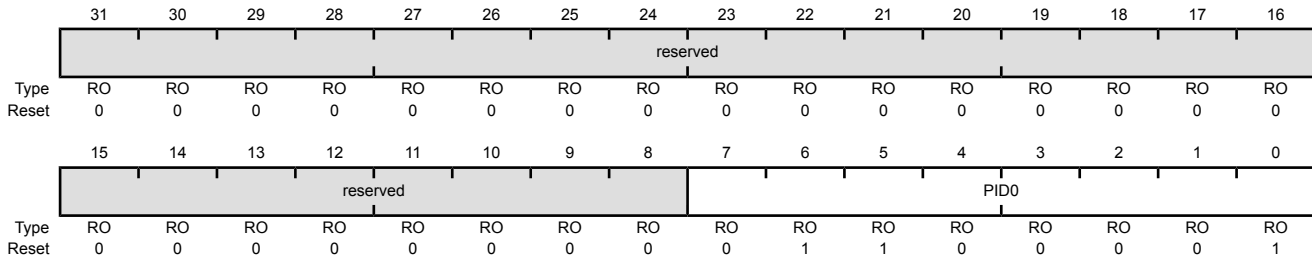
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | GPIO Peripheral ID Register [31:24] |

Register 27: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 0 (GPIOPeriphID0)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFE0
 Type RO, reset 0x0000.0061



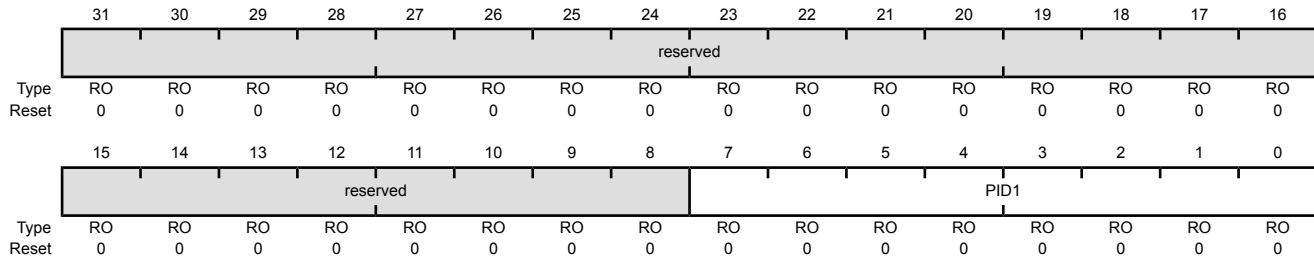
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x61 | GPIO Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. |

Register 28: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 1 (GPIOPeriphID1)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFE4
 Type RO, reset 0x0000.0000



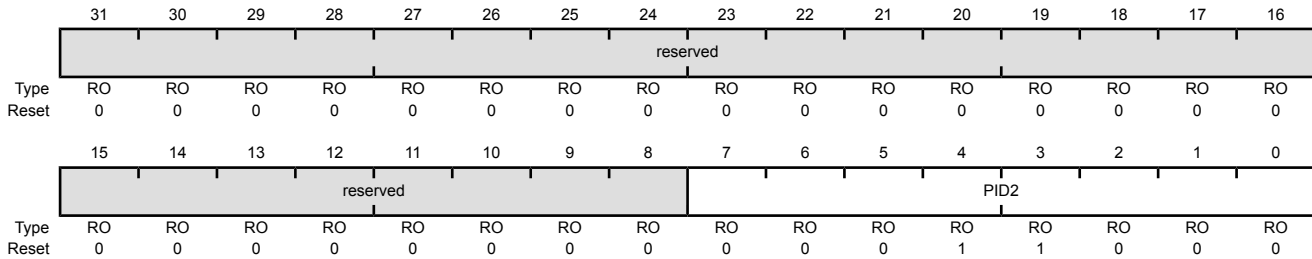
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x00 | GPIO Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. |

Register 29: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 2 (GPIOPeriphID2)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFE8
 Type RO, reset 0x0000.0018



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | GPIO Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. |

Register 30: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 3 (GPIOPeriphID3)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFEC

Type RO, 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 | | | | | | | | PID3 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

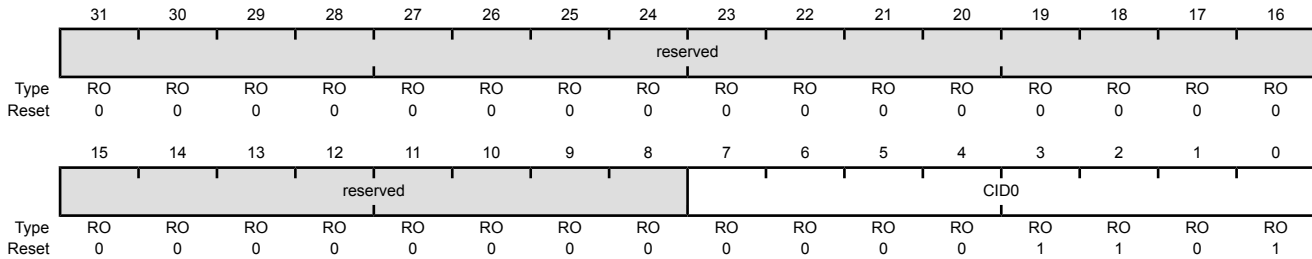
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | GPIO Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. |

Register 31: GPIO PrimeCell Identification 0 (GPIOCellID0), offset 0xFF0

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** 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 (GPIOCellID0)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFF0
 Type RO, reset 0x0000.000D



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | GPIO PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system. |

Register 32: GPIO PrimeCell Identification 1 (GPIOCellID1), offset 0xFF4

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** 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 (GPIOCellID1)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFF4
 Type RO, reset 0x0000.00F0

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | CID1 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

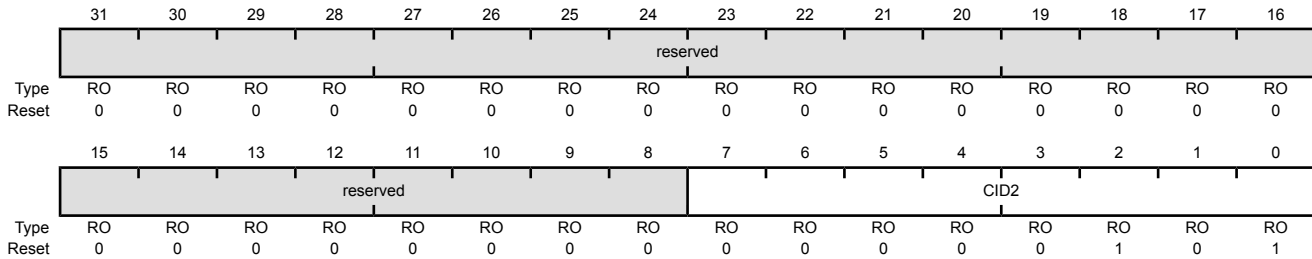
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | GPIO PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system. |

Register 33: GPIO PrimeCell Identification 2 (GPIOCellID2), offset 0xFF8

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** 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 (GPIOCellID2)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFF8
 Type RO, reset 0x0000.0005



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | GPIO PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system. |

Register 34: GPIO PrimeCell Identification 3 (GPIOCellID3), offset 0xFFC

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** 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 (GPIOCellID3)

GPIO Port A (APB) base: 0x4000.4000
 GPIO Port A (AHB) base: 0x4005.8000
 GPIO Port B (APB) base: 0x4000.5000
 GPIO Port B (AHB) base: 0x4005.9000
 GPIO Port C (APB) base: 0x4000.6000
 GPIO Port C (AHB) base: 0x4005.A000
 GPIO Port D (APB) base: 0x4000.7000
 GPIO Port D (AHB) base: 0x4005.B000
 GPIO Port E (APB) base: 0x4002.4000
 GPIO Port E (AHB) base: 0x4005.C000
 GPIO Port F (APB) base: 0x4002.5000
 GPIO Port F (AHB) base: 0x4005.D000
 GPIO Port G (APB) base: 0x4002.6000
 GPIO Port G (AHB) base: 0x4005.E000
 GPIO Port H (APB) base: 0x4002.7000
 GPIO Port H (AHB) base: 0x4005.F000
 GPIO Port J (APB) base: 0x4003.D000
 GPIO Port J (AHB) base: 0x4006.0000
 Offset 0xFFC
 Type RO, reset 0x0000.00B1

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID3 | | | | | | | |
| 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 | 1 | 1 | 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. |
| 7:0 | CID3 | RO | 0xB1 | GPIO PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system. |

11 External Peripheral Interface (EPI)

The External Peripheral Interface is a high-speed parallel bus for external peripherals or memory. It has several modes of operation to interface gluelessly to many types of external devices. The External Peripheral Interface is similar to a standard microprocessor address/data bus, except that it must typically be connected to just one type of external device. Enhanced capabilities include μ DMA support, clocking control and support for external FIFO buffers.

The EPI has the following features:

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

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

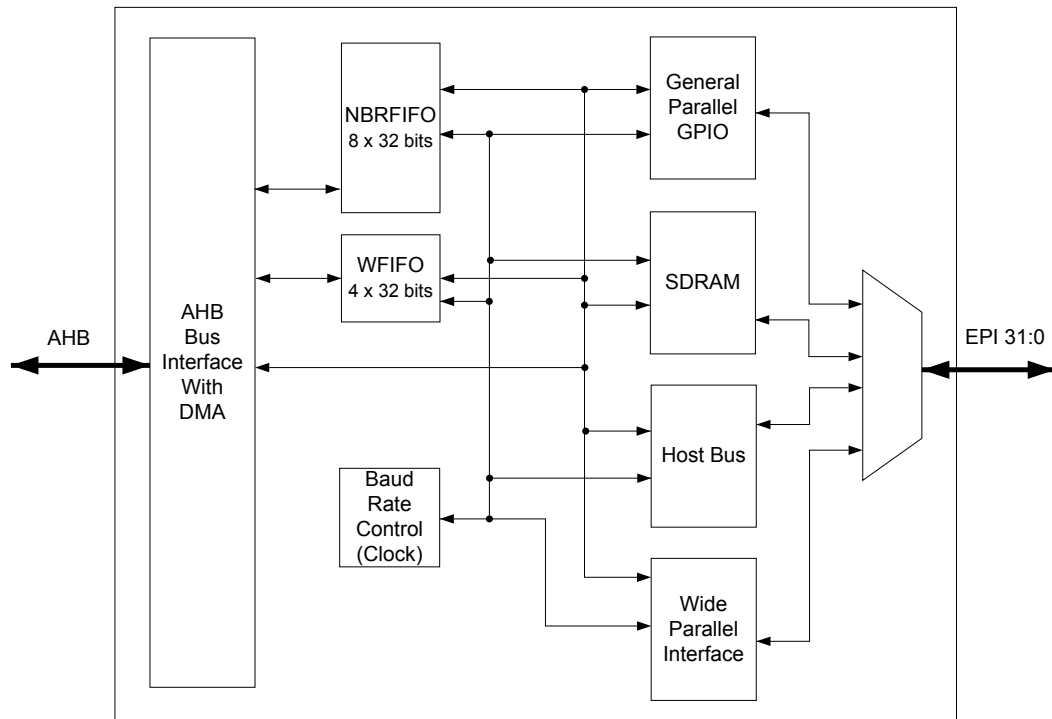
- Synchronous Dynamic Random Access Memory (SDRAM)
 - Supports x16 (single data rate) SDRAM at up to 50 MHz
 - Supports low-cost SDRAMs up to 64 MB (512 Mb)
 - Includes automatic refresh and access to all banks/rows
 - Includes a Sleep/Standby mode to keep contents active with minimal power draw
 - Multiplexed address/data interface for reduced pin count
- Host-bus
 - Traditional x8 and x16 MCU bus interface capabilities
 - Similar device compatibility options as PIC, ATmega, 8051, and others
 - Access to SRAM, NOR Flash memory, and other devices, with up to 1 MB of addressing
 - Support of both muxed and de-muxed address and data

- Access to a range of devices supporting the non-address FIFO x8 and x16 interface variant, with support for external FIFO (XFIFO) `EMPTY` and `FULL` signals
- Speed controlled, with read and write data wait-state counters
- Chip select modes include ALE, CSn, Dual CSn and ALE with dual CSn
- Manual chip-enable (or use extra address pins)
- General Purpose
 - Wide parallel interfaces for fast communications with CPLDs and FPGAs
 - Data widths up to 32-bits
 - Data rates up to 150 Mbytes/second
 - Optional “address” sizes from 4-bits to 16-bits
 - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
 - 1 to 32 bits, FIFOed with speed control
 - Useful for custom peripherals or for digital data acquisition and actuator controls

11.1 EPI Block Diagram

Figure 11-1 on page 374 provides a block diagram of a Stellaris® EPI module.

Figure 11-1. EPI Block Diagram



11.2 Signal Description

Table 11-1 on page 374 lists the external signals of the EPI controller and describes the function of each. The EPI controller signals are alternate functions for GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the EPI signals. The **AFSEL** bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the EPI controller function. The number in parentheses is the encoding that must be programmed into the **PMC_n** field in the **GPIO Port Control (GPIOPCTL)** register (page 358) to assign the EPI signals to the specified GPIO port pins. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318.

Table 11-1. Signals for External Peripheral Interface

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|-------------------------|
| EPI0S0 | 83 | PH3 (8) | I/O | TTL | EPI module 0 signal 0. |
| EPI0S1 | 84 | PH2 (8) | I/O | TTL | EPI module 0 signal 1. |
| EPI0S2 | 25 | PC4 (8) | I/O | TTL | EPI module 0 signal 2. |
| EPI0S3 | 24 | PC5 (8) | I/O | TTL | EPI module 0 signal 3. |
| EPI0S4 | 23 | PC6 (8) | I/O | TTL | EPI module 0 signal 4. |
| EPI0S5 | 22 | PC7 (8) | I/O | TTL | EPI module 0 signal 5. |
| EPI0S6 | 86 | PH0 (8) | I/O | TTL | EPI module 0 signal 6. |
| EPI0S7 | 85 | PH1 (8) | I/O | TTL | EPI module 0 signal 7. |
| EPI0S8 | 74 | PE0 (8) | I/O | TTL | EPI module 0 signal 8. |
| EPI0S9 | 75 | PE1 (8) | I/O | TTL | EPI module 0 signal 9. |
| EPI0S10 | 76 | PH4 (8) | I/O | TTL | EPI module 0 signal 10. |

Table 11-1. Signals for External Peripheral Interface (*continued*)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|-------------------------|
| EPI0S11 | 63 | PH5 (8) | I/O | TTL | EPI module 0 signal 11. |
| EPI0S12 | 42 | PF4 (8) | I/O | TTL | EPI module 0 signal 12. |
| EPI0S13 | 19 | PG0 (8) | I/O | TTL | EPI module 0 signal 13. |
| EPI0S14 | 18 | PG1 (8) | I/O | TTL | EPI module 0 signal 14. |
| EPI0S15 | 41 | PF5 (8) | I/O | TTL | EPI module 0 signal 15. |
| EPI0S16 | 14 | PJ0 (8) | I/O | TTL | EPI module 0 signal 16. |
| EPI0S17 | 87 | PJ1 (8) | I/O | TTL | EPI module 0 signal 17. |
| EPI0S18 | 39 | PJ2 (8) | I/O | TTL | EPI module 0 signal 18. |
| EPI0S19 | 97 | PD4 (10) | I/O | TTL | EPI module 0 signal 19. |
| EPI0S20 | 12 | PD2 (8) | I/O | TTL | EPI module 0 signal 20. |
| EPI0S21 | 13 | PD3 (8) | I/O | TTL | EPI module 0 signal 21. |
| EPI0S22 | 91 | PB5 (8) | I/O | TTL | EPI module 0 signal 22. |
| EPI0S23 | 92 | PB4 (8) | I/O | TTL | EPI module 0 signal 23. |
| EPI0S24 | 95 | PE2 (8) | I/O | TTL | EPI module 0 signal 24. |
| EPI0S25 | 96 | PE3 (8) | I/O | TTL | EPI module 0 signal 25. |
| EPI0S26 | 62 | PH6 (8) | I/O | TTL | EPI module 0 signal 26. |
| EPI0S27 | 15 | PH7 (8) | I/O | TTL | EPI module 0 signal 27. |
| EPI0S28 | 98 | PD5 (10) | I/O | TTL | EPI module 0 signal 28. |
| EPI0S29 | 99 | PD6 (10) | I/O | TTL | EPI module 0 signal 29. |
| EPI0S30 | 100 | PD7 (10) | I/O | TTL | EPI module 0 signal 30. |
| EPI0S31 | 36 | PG7 (9) | I/O | TTL | EPI module 0 signal 31. |

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

11.3 Functional Description

The EPI controller provides a glueless, programmable interface to a variety of common external peripherals such as SDRAM, Host Bus x8 and x16 devices, RAM, Flash memory, CPLDs and FPGAs. In addition, the EPI controller provides custom GPIO that can use a FIFO with speed control by using either the internal write FIFO (WFIFO) or the non-blocking read FIFO (NBRFIFO). The WFIFO can hold 4 words of data that are written to the external interface at the rate controlled by the **EPI Main Baud Rate (EPIBAUD)** register. The NBRFIFO can hold 8 words of data and samples at the rate controlled by the **EPIBAUD** register. The EPI controller provides predictable operation and thus has an advantage over regular GPIO which has more variable timing due to on-chip bus arbitration and delays across bus bridges. Blocking reads stall the CPU until the transaction completes. Non-blocking reads are performed in the background and allow the processor to continue operation. In addition, write data can also be stored in the WFIFO to allow multiple writes with no stalls.

Main read and write operations can be performed in subsets of the range 0x6000.0000 to 0xCFFF.FFFF. A read from an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value load, the read is done as a burst (when available) to maximize performance. A write to an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value store, the write is done as a burst (when available) to maximize performance.

11.3.1 Non-Blocking Reads

The EPI Controller supports a special kind of read called a non-blocking read, also referred to as a posted read. Where a normal read stalls the processor or μ DMA until the data is returned, a non-blocking read is performed in the background.

A non-blocking read is configured by writing the start address into a **EPIRADDRx** register, the size per transaction into a **EPIRSIZEx** register, and then the count of operations into a **EPIRPSTDx** register. After each read is completed, the result is written into the NBRFIFO and the **EPIRADDRx** register is incremented by the size (1, 2, or 4).

If the NBRFIFO is filled, then the reads pause until space is made available. The NBRFIFO can be configured to interrupt the processor or trigger the μ DMA based on fullness using the **EPIFIFOLVL** register. By using the trigger/interrupt method, the μ DMA (or processor) can keep space available in the NBRFIFO and allow the reads to continue unimpeded.

When performing non-blocking reads, the SDRAM controller issues two additional read transactions after the burst request is terminated. The data for these additional transfers is discarded. This situation is transparent to the user other than the additional EPI bus activity and can safely be ignored.

Two non-blocking read register sets are available to allow sequencing and ping-pong use. When one completes, the other then activates. So, for example, if 20 words are to be read from 0x100 and 10 words from 0x200, the **EPIRPSTD0** register can be set up with the read from 0x100 (with a count of 20), and the **EPIRPSTD1** register can be set up with the read from 0x200 (with a count of 10). When **EPIRPSTD0** finishes (count goes to 0), the **EPIRPSTD1** register then starts its operation. The NBRFIFO has then passed 30 values. When used with the μ DMA, it may transfer 30 values (simple sequence), or the primary/alternate model may be used to handle the first 20 in one way and the second 10 in another. It is also possible to reload the **EPIRPSTD0** register when it is finished (and the **EPIRPSTD1** register is active); thereby, keeping the interface constantly busy.

To cancel a non-blocking read, the **EPIRPSTDx** register is cleared. Care must be taken, however if the register set was active to drain away any values read into the NBRFIFO and ensure that any read in progress is allowed to complete.

To ensure that the cancel is complete, the following algorithm is used (using the **EPIRPSTD0** register for example):

```
EPIRPSTD0 = 0;
while ((EPISTAT & 0x11) == 0x10)
; // we are active and busy
// if here, then other one is active or interface no longer busy
cnt = (EPIRADDR0 – original_address) / EPIRSIZE0; // count of values read
cnt -= values_read_so_far;
// cnt is now number left in FIFO
while (cnt--)
value = EPIREADFIFO; // drain
```

The above algorithm can be optimized in code; however, the important point is to wait for the cancel to complete because the external interface could have been in the process of reading a value when the cancel came in, and it must be allowed to complete.

11.3.2 DMA Operation

The μ DMA can be used to efficiently transfer data to and from the NBRFIFO and the WFIFO. The μ DMA has one channel for write and one for read. The write channel can be configured to copy values to the WFIFO when the WFIFO is empty. For non-blocking reads, the start address, the size per transaction, and the count of elements must be programmed in the μ DMA. The NBRFIFO level at which the μ DMA triggers the read accesses must also be programmed. Note that both non-blocking read channels can be used, and they fill the NBRFIFO such that one runs to completion, then the next one starts (they do not interleave). For blocking reads, any μ DMA channel can be used as a memory-to-memory transfer (or memory to peripheral, where some other peripheral is used). In this situation, the μ DMA is blocked when reading, thus the μ DMA is not able to service another channel until the read is done. As a result, the arbitration size should normally be programmed to one access at a time. See “Micro Direct Memory Access (μ DMA)” on page 258 for more information on configuring the μ DMA.

11.4 Initialization and Configuration

To enable and initialize the EPI controller, the following steps are necessary:

1. Enable the EPI module using the **RCGC1** register. See page 175.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register. See page 184. To find out which GPIO port to enable, refer to Table 11-1 on page 374.
3. Set the GPIO **AFSEL** bits for the appropriate pins. See page 340. To determine which GPIOs to configure, see Table 24-4 on page 1028.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected. See page 342 and page 350.
5. Configure the **PMC_n** fields in the **GPIOPCTL** register to assign the EPI signals to the appropriate pins. See page 358 and Table 24-5 on page 1035.
6. Select the mode for the EPI block to SDRAM, HB8, HB16, or general parallel use, using the **MODE** field in the **EPI Configuration (EPICFG)** register. Set the mode-specific details (if needed) using the appropriate mode configuration **EPI xxx Configuration (EPIxxxCFG)** and **EPI xxx Configuration 2 (EPIxxxCFG2)** registers. Set the **EPI Main Baud Rate (EPIBAUD)** register if the baud rate must be slower than the system clock rate.
7. Configure the address mapping using the **EPI Address Map (EPIADDRMAP)** register. The selected start address and range is dependent on the type of external device and maximum address (as appropriate). For example, for a 512-Mb SDRAM, program the **ERADR** field to 0x1 for address 0x6000.0000 or 0x2 for address 0x8000.0000; and program the **ERSZ** field to 0x3 for 256 MB. If using General-Purpose mode and no address at all, program the **EPADR** field to 0x1 for address 0xA000.0000 or 0x2 for address 0xC000.0000; and program the **EPSZ** field to 0x0 for 256 bytes.
8. To read or write directly, use the mapped address area (configured with the **EPIADDRMAP** register). Up to 4 or 5 writes can be performed at once without blocking. Each read is blocked until the value is retrieved.
9. To perform a non-blocking read, see “Non-Blocking Reads” on page 376.

The following sub-sections describe the initialization and configuration for each of the modes of operation. Care must be taken to initialize everything properly to ensure correct operation. Control of the GPIO states is also important, as changes may cause the external device to interpret pin states as actions or commands (see “Register Descriptions” on page 329). Normally, a pull-up or pull-down is needed on the board to at least control the chip-select or chip-enable as the Stellaris® GPIOs come out of reset in tri-state.

11.4.1 SDRAM Mode

When activating the SDRAM mode, it is important to consider a few points:

1. Generally, it takes over 100 μ s from when the mode is activated to when the first operation is allowed. The SDRAM controller begins the SDRAM initialization sequence as soon as the mode is selected and enabled via the **EPICFG** register. It is important that the GPIOs are properly configured before the SDRAM mode is enabled, as the EPI controller is relying on the GPIO block's ability to drive the pins immediately. As part of the initialization sequence, the LOAD MODE REGISTER command is automatically sent to the SDRAM with a value of 0x27, which sets a CAS latency of 2 and a full page burst length.
2. The **INITSEQ** bit in the **EPI Status (EPISTAT)** register can be checked to determine when the initialization sequence is complete.
3. When using a frequency range and/or refresh value other than the default value, it is important to configure the **FREQ** and **RFSH** fields in the **EPI SDRAM Configuration (EPISDRAMCFG)** register shortly after activating the mode. After the 100- μ s startup time, the EPI block must be configured properly to keep the SDRAM contents stable.
4. The **SLEEP** bit in the **EPISDRAMCFG** register may be configured to put the SDRAM into a low-power self-refreshing state. It is important to note that the SDRAM mode must not be disabled once enabled, or else the SDRAM is no longer clocked and the contents are lost.

The **SIZE** field of the **EPISDRAMCFG** register must be configured correctly based on the amount of SDRAM in the system.

The **FREQ** field must be configured according to the value that represents the range being used. Based on the range selected, the number of external clocks used between certain operations (for example, PRECHARGE or ACTIVATE) is determined. If a higher frequency is given than is used, then the only downside is that the peripheral is slower (uses more cycles for these delays). If a lower frequency is given, incorrect operation occurs.

See “External Peripheral Interface (EPI)” on page 1049 for timing details for the SDRAM mode.

11.4.1.1 External Signal Connections

The Table 11-2 on page 378 table defines how EPI module signals should be connected to SDRAMs. The table applies when using a x16 SDRAM up to 512 Mb. Note that the EPI signals must use 8-mA drive when interfacing to SDRAM, see page 344. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 11-2. EPI SDRAM Signal Connections

| EPI Signal | SDRAM Signal ^a | |
|------------|---------------------------|----|
| EPIOS0 | A0 | D0 |
| EPIOS1 | A1 | D1 |
| EPIOS2 | A2 | D2 |

Table 11-2. EPI SDRAM Signal Connections (continued)

| EPI Signal | SDRAM Signal ^a | |
|-----------------|---------------------------|-----|
| EPI0S3 | A3 | D3 |
| EPI0S4 | A4 | D4 |
| EPI0S5 | A5 | D5 |
| EPI0S6 | A6 | D6 |
| EPI0S7 | A7 | D7 |
| EPI0S8 | A8 | D8 |
| EPI0S9 | A9 | D9 |
| EPI0S10 | A10 | D10 |
| EPI0S11 | A11 | D11 |
| EPI0S12 | A12 ^b | D12 |
| EPI0S13 | BA0 | D13 |
| EPI0S14 | BA1 | D14 |
| EPI0S15 | D15 | |
| EPI0S16 | DQML | |
| EPI0S17 | DQMH | |
| EPI0S18 | CASn | |
| EPI0S19 | RASn | |
| EPI0S20-EPI0S27 | not used | |
| EPI0S28 | WEn | |
| EPI0S29 | CSn | |
| EPI0S30 | CKE | |
| EPI0S31 | CLK | |

a. If 2 signals are listed, connect the EPI signal to both pins.

b. Only for 256/512 Mb SDRAMs

11.4.1.2 Refresh Configuration

The refresh count is based on the external clock speed and the number of rows per bank as well as the refresh period. The `RFSH` field represents how many external clock cycles remain before an AUTO-REFRESH is required. The normal formula is:

$$RFSH = (t_{Refresh_us} / number_rows) / ext_clock_period$$

A refresh period is normally 64 ms, or 64000 μ s. The number of rows is normally 4096 or 8192. The `ext_clock_period` is a value expressed in μ sec and is derived by dividing 1000 by the clock speed expressed in MHz. So, 50 MHz is 1000/50=20 ns, or 0.02 μ s. A typical SDRAM is 4096 rows per bank if the system clock is running at 50 MHz with an `EPIBAUD` register value of 0:

$$RFSH = (64000/4096) / 0.02 = 15.625 \mu s / 0.02 \mu s = 781.25$$

The default value in the `RFSH` field is 750 decimal or 0x2EE to allow for a margin of safety and providing 15 μ s per refresh. It is important to note that this number should always be smaller or equal to what is required by the above equation. For example, if running the external clock at 25 MHz (40 ns per clock period), 390 is the highest number that may be used. Note that the external clock may be 25 MHz when the system clock is 25 MHz or when the system clock is 50 MHz and configuring the `COUNT0` field in the `EPIBAUD` register to 1 (divide by 2).

If a number larger than allowed is used, the SDRAM is not refreshed often enough, and data is lost.

11.4.1.3 Bus Interface Speed

The **EPI SDRAM Configuration 2 (EPISDRAMCFG2)** register should be configured to optimize the speed of the processor and the SDRAM interface. The EPI Controller SDRAM interface can operate up to 50 MHz. In default mode, the **RCM** bit in the **EPISDRAMCFG2** register is clear, the **COUNT0** field in the **EPIBAUD** register is 0x0000, and read data is captured on the rising edge of the SDRAM clock (**EPIOS31**). This configuration operates at full speed for system clock (SysClk) speeds up to 50 MHz. However, if SysClk is running at higher speeds, the bus interface can run only as fast as half speed in default mode (the **COUNT0** field is at least 0x0001). If the **RCM** bit is set, read data is captured on the rising edge of a feedback SDRAM clock. This mode adds one clock of internal latency on the first read access (not SDRAM latency) but does allow the bus interface to run at full speed. Note that signals are asserted and deasserted off the falling edge of **EPIOS31**, and data is latched on the rising edge of **EPIOS31**. Table 11-3 on page 380 shows the configuration options at various speeds.

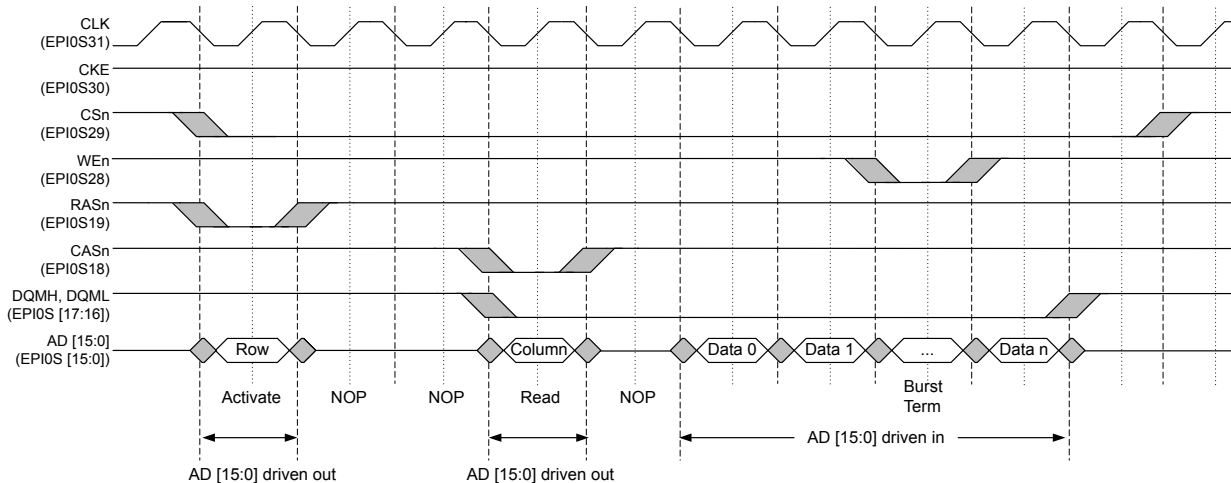
Table 11-3. SDRAM Interface Configurations

| SysClk | SDRAM Clock, RCM=0 | SDRAM Clock, RCM=1 |
|---------|---------------------|--------------------|
| 50 MHz | 50 MHz (COUNT0=0x0) | N/A |
| 66 MHz | 33 MHz (COUNT0=0x1) | 66 MHz (COUNT=0x0) |
| 80 MHz | 40 MHz (COUNT0=0x1) | N/A |
| 100 MHz | 50 MHz (COUNT0=0x1) | N/A |

11.4.1.4 Non-Blocking Read Cycle

Figure 11-2 on page 380 shows a non-blocking read cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the **EPIOS[15:0]** signals. With the programmed CAS latency of 2, the Read command with the column address on the **EPIOS[15:0]** signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the **EPIOS[15:0]** signals on every rising clock edge. The Burst Terminate command is issued during the cycle when the next-to-last halfword is read in. The **DQMH** and **DQML** signals are deasserted after the last halfword of data is received; the **CSn** signal deasserts on the following clock cycle, signaling the end of the read cycle. At least one clock period of inactivity separates any two SDRAM cycles.

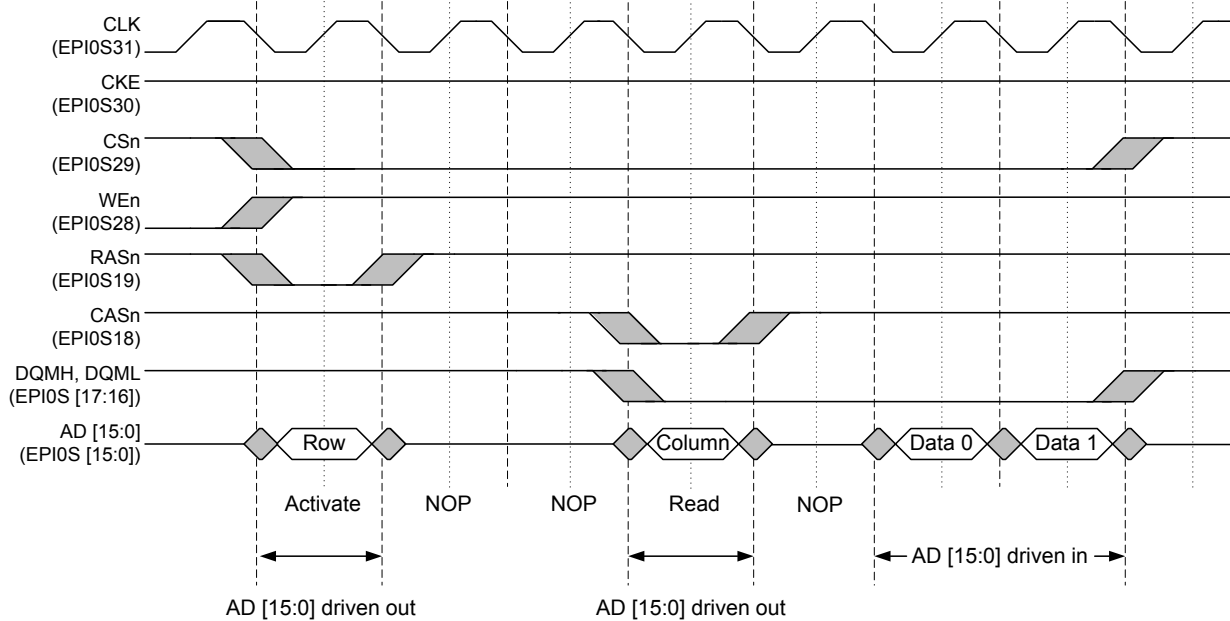
Figure 11-2. SDRAM Non-Blocking Read Cycle



11.4.1.5 Normal Read Cycle

Figure 11-3 on page 381 shows a normal read cycle of n halfwords; n can be 1 or 2. The cycle begins with the Activate command and the row address on the $EPIOS[15:0]$ signals. With the programmed CAS latency of 2, the Read command with the column address on the $EPIOS[15:0]$ signals follows after 2 clock cycles. Following one more NOP cycle, data is read in on the $EPIOS[15:0]$ signals on every rising clock edge. The $DQMH$, $DQML$, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the cycle. At least one clock period of inactivity separates any two SDRAM cycles.

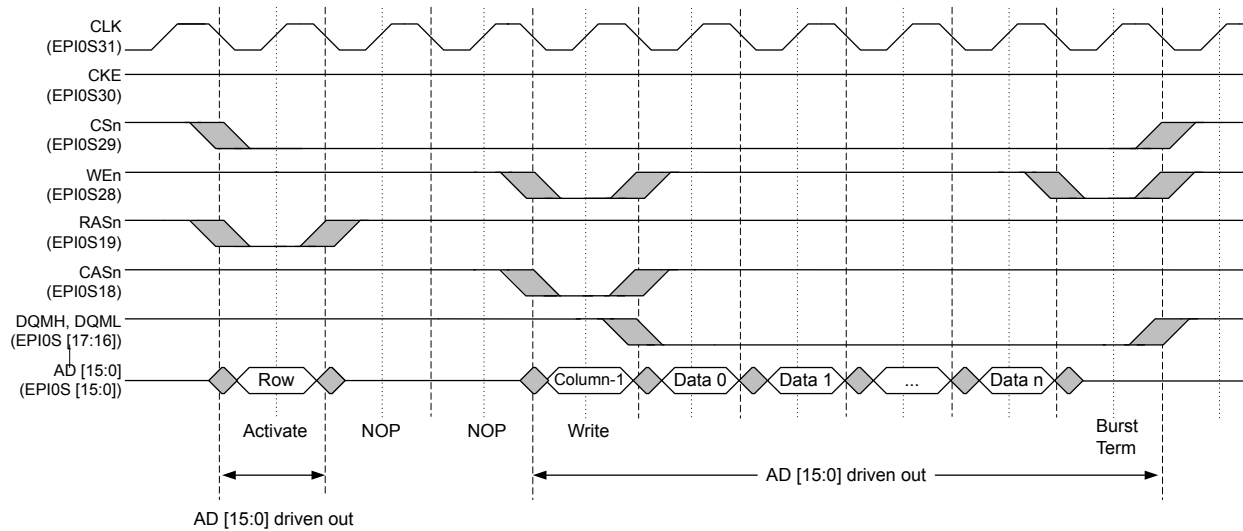
Figure 11-3. SDRAM Normal Read Cycle



11.4.1.6 Write Cycle

Figure 11-4 on page 382 shows a write cycle of n halfwords; n can be any number greater than or equal to 1. The cycle begins with the Activate command and the row address on the $EPIOS[15:0]$ signals. With the programmed CAS latency of 2, the Write command with the column address on the $EPIOS[15:0]$ signals follows after 2 clock cycles. When writing to SDRAMs, the Write command is presented with the first halfword of data. Because the address lines and the data lines are multiplexed, the column address is modified to be (programmed address - 1). During the Write command, the $DQMH$ and $DQML$ signals are high, so no data is written to the SDRAM. On the next clock, the $DQMH$ and $DQML$ signals are asserted, and the data associated with the programmed address is written. The Burst Terminate command occurs during the clock cycle following the write of the last halfword of data. The wEn , $DQMH$, $DQML$, and CSn signals are deasserted after the last halfword of data is received, signaling the end of the access. At least one clock period of inactivity separates any two SDRAM cycles.

Figure 11-4. SDRAM Write Cycle



11.4.2 Host Bus Mode

Host Bus supports the traditional 8-bit and 16-bit interfaces popularized by the 8051 devices and SRAM devices. This interface is asynchronous and uses strobe pins to control activity.

11.4.2.1 Control Pins

The main three strobes are ALE (Address latch enable), WRn (write), and RDn (sometimes called OEn, used for read). Note that the timings are designed for older logic and so are hold-time vs. setup-time specific. To ensure proper operation on this bus, the EPI block uses two system clocks per transition to allow significant skewing of control vs. data signals. So, for example, ALE rises one EPI clock before ADDR/DATA is asserted. Likewise, ALE falls (latch point) one EPI clock before DATA changes or tri-states. The same approach is used for the WRn and RDn/OEn strobes.

The ALE can be changed to CSn through the **EPI Host-Bus n Configuration 2 (EPIHBnCFG2)** register. The ALE is best used for Host-Bus muxed mode in which EPI address and data pins are shared. All Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold it until the data phase. CSn is best used for Host-Bus unmuxed mode in which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access is occurring. Both the ALE and the CSn modes can be enhanced to access two external devices using settings in the **EPIHBnCFG2** register.

For FIFO mode, the ALE is not used, and two input holds are optionally supported to gate input and output to what the XFIFO can handle.

Host-Bus 8 and Host-Bus 16 modes are very configurable. The user has the ability to connect 1 or 2 external devices to the EPI signals as well as control whether byte select signals are provided in HB16 mode. These capabilities depend on the configuration of the **MODE** field in the **EPIHBnCFG** register, the **CSCFG** field in the **EPIHBnCFG2** register, and the **BSEL** bit in the **EPIHB16CFG** register. When **BSEL=1**, byte select signals are provided, so byte-sized data can be read and written at any address, however these signals reduce the available address width by 2 pins. When **BSEL=0**, byte reads and writes at odd addresses only act on the even byte, and byte writes at even addresses write invalid values into the odd byte. As a result, accesses should be made as half-words (16-bits) or words (32-bits). In C/C++, programmers should use only short int and long int for accesses. Table

11-4 on page 383 shows the capabilities of the HB8 and HB16 modes as well as the available address bits with the possible combinations of these bits.

Table 11-4. Capabilities of Host Bus 8 and Host Bus 16 Modes

| Host Bus Type | MODE | CSCFG | Max # of External Devices | BSEL | Byte Access | Available Address |
|---------------|------|----------|---------------------------|------|-------------|-------------------|
| HB8 | 0x0 | 0x0, 0x1 | 1 | N/A | Always | 28 bits |
| HB8 | 0x0 | 0x2 | 2 | N/A | Always | 27 bits |
| HB8 | 0x0 | 0x3 | 2 | N/A | Always | 26 bits |
| HB8 | 0x1 | 0x0, 0x1 | 1 | N/A | Always | 20 bits |
| HB8 | 0x1 | 0x2 | 2 | N/A | Always | 19 bits |
| HB8 | 0x1 | 0x3 | 2 | N/A | Always | 18 bits |
| HB8 | 0x3 | 0x1 | 1 | N/A | Always | none |
| HB8 | 0x3 | 0x3 | 2 | N/A | Always | none |
| HB16 | 0x0 | 0x0, 0x1 | 1 | 0 | No | 28 bits |
| HB16 | 0x0 | 0x0, 0x1 | 1 | 1 | Yes | 26 bits |
| HB16 | 0x0 | 0x2 | 2 | 0 | No | 27 bits |
| HB16 | 0x0 | 0x2 | 2 | 1 | Yes | 25 bits |
| HB16 | 0x0 | 0x3 | 2 | 0 | No | 26 bits |
| HB16 | 0x0 | 0x3 | 2 | 1 | Yes | 24 bits |
| HB16 | 0x1 | 0x0, 0x1 | 1 | 0 | No | 12 bits |
| HB16 | 0x1 | 0x0, 0x1 | 1 | 1 | Yes | 10 bits |
| HB16 | 0x1 | 0x2 | 2 | 0 | No | 11 bits |
| HB16 | 0x1 | 0x2 | 2 | 1 | Yes | 9 bits |
| HB16 | 0x1 | 0x3 | 2 | 0 | No | 10 bits |
| HB16 | 0x1 | 0x3 | 2 | 1 | Yes | 8 bits |
| HB16 | 0x3 | 0x1 | 1 | 0 | No | none |
| HB16 | 0x3 | 0x1 | 1 | 1 | Yes | none |
| HB16 | 0x3 | 0x3 | 2 | 0 | No | none |
| HB16 | 0x3 | 0x3 | 2 | 1 | Yes | none |

Table 11-5 on page 383 shows how the EPI[31:0] signals function while in Host-Bus 8 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the EPIHB8CFG2 register and on the chip select configuration selected by the CSCFG field in the same register. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 11-5. EPI Host-Bus 8 Signal Connections

| EPI Signal | CSCFG | HB8 Signal (MODE =ADMUX) | HB8 Signal (MODE =ADNOMUX (SRAM)) | HB8 Signal (MODE =XFIFO) |
|------------|----------------|--------------------------|-----------------------------------|--------------------------|
| EPI0S0 | X ^a | AD0 | D0 | D0 |
| EPI0S1 | X | AD1 | D1 | D1 |
| EPI0S2 | X | AD2 | D2 | D2 |
| EPI0S3 | X | AD3 | D3 | D3 |
| EPI0S4 | X | AD4 | D4 | D4 |

Table 11-5. EPI Host-Bus 8 Signal Connections (continued)

| EPI Signal | CSCFG | HB8 Signal (MODE =ADMUX) | HB8 Signal (MODE =ADNOMUX (SRAM)) | HB8 Signal (MODE =XFIFO) |
|------------|-------|--------------------------|-----------------------------------|--------------------------|
| EPI0S5 | X | AD5 | D5 | D5 |
| EPI0S6 | X | AD6 | D6 | D6 |
| EPI0S7 | X | AD7 | D7 | D7 |
| EPI0S8 | X | A8 | A0 | - |
| EPI0S9 | X | A9 | A1 | - |
| EPI0S10 | X | A10 | A2 | - |
| EPI0S11 | X | A11 | A3 | - |
| EPI0S12 | X | A12 | A4 | - |
| EPI0S13 | X | A13 | A5 | - |
| EPI0S14 | X | A14 | A6 | - |
| EPI0S15 | X | A15 | A7 | - |
| EPI0S16 | X | A16 | A8 | - |
| EPI0S17 | X | A17 | A9 | - |
| EPI0S18 | X | A18 | A10 | - |
| EPI0S19 | X | A19 | A11 | - |
| EPI0S20 | X | A20 | A12 | - |
| EPI0S21 | X | A21 | A13 | - |
| EPI0S22 | X | A22 | A14 | - |
| EPI0S23 | X | A23 | A15 | - |
| EPI0S24 | X | A24 | A16 | - |
| EPI0S25 | 0x0 | A25 ^b | A17 | - |
| | 0x1 | | | - |
| | 0x2 | | | CS1n |
| | 0x3 | | | - |
| EPI0S26 | 0x0 | A26 | A18 | FEMPTY |
| | 0x1 | | | |
| | 0x2 | CS0n | CS0n | |
| | 0x3 | | | |
| EPI0S27 | 0x0 | A27 | A19 | FFULL |
| | 0x1 | | | |
| | 0x2 | CSn1 | CSn1 | |
| | 0x3 | | | |
| EPI0S28 | X | RDn/OEn | RDn/OEn | RDn |
| EPI0S29 | X | WRn | WRn | WRn |
| EPI0S30 | 0x0 | ALE | ALE | - |
| | 0x1 | CSn | CSn | CSn |
| | 0x2 | CS0n | CS0n | CS0n |
| | 0x3 | ALE | ALE | - |
| EPI0S31 | X | Clock | Clock | Clock |

a. "X" indicates the state of this field is a don't care.

b. When an entry straddles several row, the signal configuration is the same for all rows.

Table 11-6 on page 385 shows how the EPI[31:0] signals function while in Host-Bus 16 mode. Notice that the signal configuration changes based on the address/data mode selected by the MODE field in the **EPIHB16CFG2** register, on the chip select configuration selected by the CSCFG field in the same register, and on whether byte selects are used as configured by the BSEL bit in the **EPIHB16CFG** register. Any unused EPI controller signals can be used as GPIOs or another alternate function.

Table 11-6. EPI Host-Bus 16 Signal Connections

| EPI Signal | CSCFG | BSEL | HB16 Signal (MODE =ADMUX) | HB16 Signal (MODE =ADNOMUX (SRAM)) | HB16 Signal (MODE =XFIFO) |
|------------|----------------|--------|---------------------------|------------------------------------|---------------------------|
| EPI0S0 | X ^a | X | AD0 | D0 | D0 |
| EPI0S1 | X | X | AD1 | D1 | D1 |
| EPI0S2 | X | X | AD2 | D2 | D2 |
| EPI0S3 | X | X | AD3 | D3 | D3 |
| EPI0S4 | X | X | AD4 | D4 | D4 |
| EPI0S5 | X | X | AD5 | D5 | D5 |
| EPI0S6 | X | X | AD6 | D6 | D6 |
| EPI0S7 | X | X | AD7 | D7 | D7 |
| EPI0S8 | X | X | AD8 | D8 | D8 |
| EPI0S9 | X | X | AD9 | D9 | D9 |
| EPI0S10 | X | X | AD10 | D10 | D10 |
| EPI0S11 | X | X | AD11 | D11 | D11 |
| EPI0S12 | X | X | AD12 | D12 | D12 |
| EPI0S13 | X | X | AD13 | D13 | D13 |
| EPI0S14 | X | X | AD14 | D14 | D14 |
| EPI0S15 | X | X | AD15 | D15 | D15 |
| EPI0S16 | X | X | A16 | A0 ^b | - |
| EPI0S17 | X | X | A17 | A1 | - |
| EPI0S18 | X | X | A18 | A2 | - |
| EPI0S19 | X | X | A19 | A3 | - |
| EPI0S20 | X | X | A20 | A4 | - |
| EPI0S21 | X | X | A21 | A5 | - |
| EPI0S22 | X | X | A22 | A6 | - |
| EPI0S23 | X ^c | 0 | A23 | A7 | - |
| | | 1 | | | BSEL0n |
| EPI0S24 | 0x0 | 0 | A24 | A8 | - |
| | | 1 | | | BSEL1n |
| | 0x1 | 0 | | | - |
| | | 1 | | | BSEL1n |
| | 0x2 | 0 | | | - |
| | | 1 | | | BSEL1n |
| | 0x3 | 0 | | | - |
| | | 1 | | | BSEL0n |
| | | BSEL0n | BSEL0n | BSEL1n | |

Table 11-6. EPI Host-Bus 16 Signal Connections (*continued*)

| EPI Signal | CSCFG | BSEL | HB16 Signal (MODE =ADMUX) | HB16 Signal (MODE =ADNOMUX (SRAM)) | HB16 Signal (MODE =XFIFO) |
|------------|-------|------|---------------------------|------------------------------------|---------------------------|
| EPI0S25 | 0x0 | X | A25 | A9 | - |
| | | 0x1 | | | |
| | 0x2 | 0 | A25 | A9 | CS1n |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x3 | 0 | A25 | A9 | -- |
| | | 1 | BSEL1n | BSEL1n | |
| EPI0S26 | 0x0 | 0 | A26 | A10 | FEMPTY |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x1 | 0 | A26 | A10 | |
| | | 1 | BSEL0n | BSEL0n | |
| | 0x2 | 0 | A26 | A10 | |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x3 | X | CS0n | CS0n | |
| | | | | | |
| EPI0S27 | 0x0 | 0 | A27 | A11 | FFULL |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x1 | 0 | A27 | A11 | |
| | | 1 | BSEL1n | BSEL1n | |
| | 0x2 | X | CS1n | CS1n | |
| | 0x3 | X | | | |
| EPI0S28 | X | X | RDn/OEn | RDn/OEn | RDn |
| EPI0S29 | X | X | WRn | WRn | WRn |
| EPI0S30 | 0x0 | X | ALE | ALE | - |
| | 0x1 | X | CSn | CSn | CSn |
| | 0x2 | X | CS0n | CS0n | CS0n |
| | 0x3 | X | ALE | ALE | - |
| EPI0S31 | X | X | Clock | Clock | Clock |

a. "X" indicates the state of this field is a don't care.

b. In this mode, half-word accesses are used. AO is the LSB of the address and is equivalent to the system A1 address.

c. When an entry straddles several row, the signal configuration is the same for all rows.

11.4.2.2 Speed of Transactions

The COUNT0 field in the **EPIBAUD** register must be configured to set the main transaction rate based on what the slave device can support (including wiring considerations). The main control transitions are normally $\frac{1}{2}$ the baud rate (COUNT0 = 1) because the EPI block forces data vs. control to change on alternating clocks. When using dual chip-selects, each chip select can access the bus using differing baud rates by setting the CSBAUD bit in the **EPIHBnCFG2** register. In this case, the COUNT0 field controls the CS0n transactions, and the COUNT1 field controls the CS1n transactions.

Additionally, the Host-Bus mode provides read and write wait states for the data portion to support different classes of device. These wait states stretch the data period (hold the rising edge of data strobe) and may be used in all four sub-modes. The wait states are set using the WRWS and RDWS bits in the **EPI Host-Bus n Configuration (EPIHBnCFG)** register.

11.4.2.3 Sub-Modes of Host Bus 8/16

The EPI controller supports four variants of the Host-Bus model using 8 or 16 bits of data in all four cases. The four sub-modes are selected using the `MODE` bits in the **EPIHBnCFG** register, and are:

1. Address and data are muxed. This scheme is used by many 8051 devices, some Microchip PIC parts, and some ATmega parts. When used for standard SRAMs, a latch must be used between the microcontroller and the SRAM. This sub-mode is provided for compatibility with existing devices that support data transfers without a latch (for example, LCD controllers or CPLDs). In general, the de-muxed sub-mode should normally be used. The ALE configuration should be used in this mode, as all Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold until the data phase. The ALE configuration is controlled by configuring the `CSCFG` field to be 0x0 in the **EPIHBnCFG2** register. The ALE can be enhanced to access two external devices with the addition of two separate CSn signals. By configuring the `CSCFG` field in the to be 0x3 in the **EPIHBnCFG2**, `EPIOS30` functions as ALE, `EPIOS27` functions as CS1n, and `EPIOS26` functions as CS0n. The CSn is best used for Host-Bus unmuxed mode which EPI address and data pins are separate. The CSn indicates when the address and data phases of a read or write access are occurring.
2. Address and data are separate with 8 or 16 bits of data and up to 20 bits of address (1 MB). This scheme is used by more modern 8051 devices, as well as some PIC and ATmega parts. This mode is generally used with real SRAMs, many EEPROMs, and many NOR Flash memory devices. Note that there is no hardware command write support for Flash memory devices; this mode should only be used for Flash memory devices programmed at manufacturing time. If a Flash memory device must be written and does not support a direct programming model, the command mechanism must be performed in software. The CSn configuration should be used in this mode. The CSn signal indicates when the address and data phases of a read or write access is occurring. The CSn configuration is controlled by configuring the `CSCFG` field to be 0x1 in the **EPIHBnCFG2** register. The CSn configuration can be enhanced to access two external devices via separate CSn. By configuring the `CSCFG` field to be 0x2, `EPIOS30` functions as CS0n and `EPIOS27` functions as CS1n.
3. SRAM fast mode where address and data are separate. This sub-mode is used for real SRAMs which can be read more quickly by only changing the address (and not using RDn/OEn strobing). In this sub-mode, reads are performed by keeping the read mode selected (output enable is asserted) and then changing the address pins. The data pins are changed by the SRAM after the address pins change. For example, to read data from address 0x100 and then 0x101, the EPI controller asserts the output-enable signal and then configures the address pins to 0x100; the EPI controller then captures what is on the data pins and increments A0 to 1 (so the address is now 0x101); the EPI controller then captures what is on the data pins. Note that this mode consumes higher power because the SRAM must continuously drive the data pins. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available.
4. FIFO mode uses 8 or 16 bits of data, removes ALE and address pins and optionally adds external XFIFO FULL/EMPTY flag inputs. This scheme is used by many devices, such as radios, communication devices (including USB2 devices), and some FPGA configurations (FIFO through block RAM). This sub-mode provides the data side of the normal Host-Bus interface, but is paced by the FIFO control signals. It is important to consider that the XFIFO FULL/EMPTY control signals may stall the interface and could have an impact on blocking read latency from the processor or μ DMA.

The `WORD` bit in the **EPIHBnCFG2** register can be set to use memory more efficiently. By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses or bits [15:0] for Host-Bus 16 accesses. When the `WORD` bit is set, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8] (HB8) or [31:16] (HB16). In addition, for the three modes above (1, 2, 4) that the Host-Bus 16 mode supports, byte select signals can be optionally implemented by setting the `BSEL` bit in the **EPIHB16CFG** register.

See “External Peripheral Interface (EPI)” on page 1049 for timing details for the Host-Bus mode.

11.4.3 General-Purpose Mode

The **General-Purpose Mode Configuration (EPIGPCFG)** register is used to configure the control, data, and address pins, if used. Any unused EPI controller signals can be used as GPIOs or another alternate function. The general-purpose configuration can be used for custom interfaces with FPGAs, CPLDs, and digital data acquisition and actuator control.

General-Purpose mode is designed for three general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs. Three sizes of data and optional address are supported. Framing and clock-enable functions permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the **EPIBAUD** register baud rate (when used with the `WFIFO` and/or the `NBRFIFO`) or by the rate of accesses from software or μ DMA. Examples of this type of use include:
 - Reading 20 sensors at fixed time periods by configuring 20 pins to be inputs, configuring the `COUNT0` field in the **EPIBAUD** register to some divider, and then using non-blocking reads.
 - Implementing a very wide ganged PWM/PCM with fixed frequency for driving actuators, LEDs, etc.
 - Implementing SDIO 4 bit mode where commands are driven or captured on 6 pins with fixed timing, fed by the μ DMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free-running or gated), a framing signal (with frame size), a clock-enable input (to stretch transactions), a `READ` and `WRITE` strobe, an address (of varying sizes), and data (of varying sizes). Additionally, provisions are made for separating data and address phases.

The interface has the following optional features:

- Use of output clock (controlled by the `CLKPIN` bit in the **EPIGPCFG** register). Unclocked uses include general-purpose I/O and asynchronous interfaces (optionally using `READ` and `WRITE` strobes). Clocked interfaces allow for higher speeds and are much easier to connect to FPGAs and CPLDs (which usually include input clocks).
- Clock, if used, may be free running or gated (using the `CLKGATE` bit in the **EPIGPCFG** register). A free-running clock requires another method for determining when data is live, such as the frame pin or `READ/WRITE` strobes. A gated clock approach uses a setup-time model in which the clock controls when transactions are starting and stopping. Note that a gated clock can only be used when the `COUNT0` field in the **EPIBAUD** register has a value other than 0 (meaning the output clock is less than the system clock). The gated clock is held low until a new transaction is started and goes high at the end of the cycle where `READ/WRITE/FRAME` and address (and data if write) are emitted.

- Clock-enable input (iRDY) from the external device (controlled by the `RDYEN` bit in the **EPIGPCFG** register). The clock-enable signal uses `EPIOS27` and may only be used with a free-running clock. `RDYEN` gates transactions, no matter what state they are in. In addition, `RDYEN` is registered internally and holds the transaction state across multiple clocks if clock-disabled. Generally, `RDYEN` should be changed before the falling edge of the external clock. If the `COUNT0` field in the **EPIBAUD** register is 0, an external device can stretch the current state by clearing the `RDYEN` bit.
- Frame pin (controlled by the `FRMPIN` bit in the **EPIGPCFG** register). The frame pin may be used whether the clock is output or not, and whether the clock is free running or not. It may also be used along with the clock-enable. The frame may be a pulse (one clock) or may be 50/50 split across the frame size (controlled by the `FRM50` bit in the **EPIGPCFG** register). The frame count (the size of the frame as specified by the `FRMCNT` field in the **EPIGPCFG** register) may be between 1 and 15 clocks for pulsed and between 2 and 30 clocks for 50/50. The frame pin counts transactions and not clocks; a transaction is any clock where the `READ` or `WRITE` strobe is high (if used). So, if the `FRMCNT` bit is set, then the frame pin pulses every other transaction; if 2-cycle reads and writes are used, it pulses every other address phase. `FRM50` must be used with this in mind as it may hold state for many clocks waiting for the next transaction.
- `READ` and `WRITE` strobes may be used (controlled by the `RW` bit in the **EPIGPCFG** register). For interfaces where the direction is known (in advance, related to frame size, or other means), these strobes are not needed. For most other interfaces, `READ` and `WRITE` are used so the external peripheral knows what transaction is taking place, and if any transaction is taking place. `READ` is used in conjunction with separating the address and data phases (2-cycle mode), as explained below.
- Separation of address/request and data phases may be used on reads and writes using the `WR2CYC` and `RD2CYC` bits in the **EPIGPCFG** register. This configuration allows the external peripheral extra time to act and is more commonly used on reads. When configured to use an address as specified by the `ASIZE` field in the **EPIGPCFG** register, the address is emitted on the `READ` cycle (first cycle) and data is expected to be returned on the next cycle (when `READ` is not asserted). If no address is used, then `READ` is asserted on the first cycle and data is captured on the second cycle (when `READ` is not asserted), allowing more setup time for data. If single-cycle reads are used, then data is expected to be available on the same cycle as `READ` using the specified setup time. To use single-cycle reads, the external peripheral must have either fast combinatorial logic (relative to clock period) or must be able to setup the data in advance.

For writes, the output may be in one or two cycles. In the two-cycle case, the address (if any) is emitted on the first cycle with `WRITE` and the data is emitted on the second cycle (with `WRITE` not asserted). Although split address and write data phases are not normally needed for logic reasons, it may be useful to make read and write timings match. If 2-cycle reads or writes are used, the `RW` bit is automatically set.
- Address may be emitted (controlled by the `ASIZE` field in the **EPIGPCFG** register). The address may be 4 bits (16 possible values), 12 bits (4096 possible values), or 20 bits (1 M possible values). Size of address limits size of data, for example, 4 bits of address support 24 bits data. Address comes from the bottom bits of the address used for the transaction by the processor or μ DMA. The address signals may be used by the external peripheral as an address, code (command), or for other unrelated uses (such as a chip enable).
- Data may be 8 bits, 16 bits, 24 bits, or 32 bits (controlled by the `DSIZE` field in the **EPIGPCFG** register). 32-bit data cannot be used with address or clock or any other signal. 24-bit data can

only be used with 4-bit address or no address. 32-bit data requires that either the `WR2CYC` bit or the `RD2CYC` bit in the `EPIGPCFG` register is set.

- Memory can be used more efficiently by using the Word Access Mode. By default, the EPI controller uses data bits [7:0] when the `DSIZE` field in the `EPIGPCFG` register is 0x0; data bits [15:0] when the `DSIZE` field is 0x1; data bits [23:0] when the `DSIZE` field is 0x2; and data bits [31:0] when the `DSIZE` field is 0x3. When the `WORD` bit in the `EPIGPCFG2` register is set, the EPI controller automatically routes bytes of data onto the correct byte lanes such that data can be stored in bits [31:8] for `DSIZE=0x0` and bits [31:16] for `DSIZE=0x1`.
- When using the EPI as a GPIO interface, writes are FIFOed (up to 4 can be held at any time), and up to 32 pins are changed using the `EPIBAUD` clock rate specified by `COUNT0`. As a result, output pin control can be very precisely controlled as a function of time. By contrast, when writing to normal GPIOs, writes can only occur 8-bits at a time and take up to two clock cycles to complete. In addition, the write itself may be further delayed by the bus due to DMA or draining of a previous write. With both GPIO and EPI, reads may be performed directly, in which case the current pin states are read back. With EPI, the non-blocking interface may also be used to perform reads based on a fixed time rule via the `EPIBAUD` clock rate.

Table 11-7 on page 390 shows how the `EPI[31:0]` signals function while in General-Purpose mode. Notice that the address connections vary depending on the data-width restrictions of the external peripheral.

Table 11-7. EPI General Purpose Signal Connections

| EPI Signal | General-Purpose Signal (D8, A20) | General- Purpose Signal (D16, A12) | General- Purpose Signal (D24, A4) | General- Purpose Signal (D32) |
|------------|----------------------------------|------------------------------------|-----------------------------------|-------------------------------|
| EPI0S0 | D0 | D0 | D0 | D0 |
| EPI0S1 | D1 | D1 | D1 | D1 |
| EPI0S2 | D2 | D2 | D2 | D2 |
| EPI0S3 | D3 | D3 | D3 | D3 |
| EPI0S4 | D4 | D4 | D4 | D4 |
| EPI0S5 | D5 | D5 | D5 | D5 |
| EPI0S6 | D6 | D6 | D6 | D6 |
| EPI0S7 | D7 | D7 | D7 | D7 |
| EPI0S8 | A0 | D8 | D8 | D8 |
| EPI0S9 | A1 | D9 | D9 | D9 |
| EPI0S10 | A2 | D10 | D10 | D10 |
| EPI0S11 | A3 | D11 | D11 | D11 |
| EPI0S12 | A4 | D12 | D12 | D12 |
| EPI0S13 | A5 | D13 | D13 | D13 |
| EPI0S14 | A6 | D14 | D14 | D14 |
| EPI0S15 | A7 | D15 | D15 | D15 |
| EPI0S16 | A8 | A0 ^a | D16 | D16 |
| EPI0S17 | A9 | A1 | D17 | D17 |
| EPI0S18 | A10 | A2 | D18 | D18 |
| EPI0S19 | A11 | A3 | D19 | D19 |
| EPI0S20 | A12 | A4 | D20 | D20 |
| EPI0S21 | A13 | A5 | D21 | D21 |

Table 11-7. EPI General Purpose Signal Connections (continued)

| EPI Signal | General-Purpose Signal (D8, A20) | General- Purpose Signal (D16, A12) | General- Purpose Signal (D24, A4) | General- Purpose Signal (D32) |
|------------|----------------------------------|------------------------------------|-----------------------------------|-------------------------------|
| EPI0S22 | A14 | A6 | D22 | D22 |
| EPI0S23 | A15 | A7 | D23 | D23 |
| EPI0S24 | A16 | A8 | A0 ^b | D24 |
| EPI0S25 | A17 | A9 | A1 | D25 |
| EPI0S26 | A18 | A10 | A2 | D26 |
| EPI0S27 | A19/iRDY ^c | A11/iRDY ^c | A3/iRDY ^c | D27 |
| EPI0S28 | WR | WR | WR | D28 |
| EPI0S29 | RD | RD | RD | D29 |
| EPI0S30 | Frame | Frame | Frame | D30 |
| EPI0S31 | Clock | Clock | Clock | D31 |

a. In this mode, half-word accesses are used. AO is the LSB of the address and is equivalent to the system A1 address.

b. In this mode, word accesses are used. AO is the LSB of the address and is equivalent to the system A2 address.

c. This signal is iRDY if the `RDYEN` bit in the `EPIGPCFG` register is set.

11.5 Register Map

Table 11-8 on page 391 lists the EPI registers. The offset listed is a hexadecimal increment to the register's address, relative to the base address of 0x400D.0000. Note that the EPI controller clock must be enabled before the registers can be programmed (see page 175).

Note: A back-to-back write followed by a read of the same register reads the value that written by the first write access, not the value from the second write access. (This situation only occurs when the processor core attempts this action, the μ DMA does not do this.). To read back what was just written, another instruction must be generated between the write and read. Read-write does not have this issue, so use of read-write for clear of error interrupt cause is not affected.

Table 11-8. External Peripheral Interface (EPI) Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|--------------|------|-------------|-------------------------------------|----------|
| 0x000 | EPICFG | R/W | 0x0000.0000 | EPI Configuration | 393 |
| 0x004 | EPIBAUD | R/W | 0x0000.0000 | EPI Main Baud Rate | 394 |
| 0x010 | EPISDRAMCFG | R/W | 0x42EE.0000 | EPI SDRAM Configuration | 395 |
| 0x010 | EPIHB8CFG | R/W | 0x0000.FF00 | EPI Host-Bus 8 Configuration | 397 |
| 0x010 | EPIHB16CFG | R/W | 0x0000.FF00 | EPI Host-Bus 16 Configuration | 400 |
| 0x010 | EPIGPCFG | R/W | 0x0000.FF00 | EPI General-Purpose Configuration | 404 |
| 0x014 | EPISDRAMCFG2 | R/W | 0x0000.0000 | EPI SDRAM Configuration 2 | 408 |
| 0x014 | EPIHB8CFG2 | R/W | 0x0000.0000 | EPI Host-Bus 8 Configuration 2 | 409 |
| 0x014 | EPIHB16CFG2 | R/W | 0x0000.0000 | EPI Host-Bus 16 Configuration 2 | 411 |
| 0x014 | EPIGPCFG2 | R/W | 0x0000.0000 | EPI General-Purpose Configuration 2 | 413 |
| 0x01C | EPIADDRMAP | R/W | 0x0000.0000 | EPI Address Map | 414 |

Table 11-8. External Peripheral Interface (EPI) Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|--------------|-------|-------------|--------------------------------------|----------|
| 0x020 | EPIRSIZE0 | R/W | 0x0000.0003 | EPI Read Size 0 | 416 |
| 0x024 | EPIRADDR0 | R/W | 0x0000.0000 | EPI Read Address 0 | 417 |
| 0x028 | EPIRPSTD0 | R/W | 0x0000.0000 | EPI Non-Blocking Read Data 0 | 418 |
| 0x030 | EPIRSIZE1 | R/W | 0x0000.0003 | EPI Read Size 1 | 416 |
| 0x034 | EPIRADDR1 | R/W | 0x0000.0000 | EPI Read Address 1 | 417 |
| 0x038 | EPIRPSTD1 | R/W | 0x0000.0000 | EPI Non-Blocking Read Data 1 | 418 |
| 0x060 | EPISTAT | RO | 0x0000.0000 | EPI Status | 420 |
| 0x06C | EPIRFIFOCNT | RO | - | EPI Read FIFO Count | 422 |
| 0x070 | EPIREADFIFO | RO | 0x0000.0000 | EPI Read FIFO | 423 |
| 0x074 | EPIREADFIFO1 | RO | 0x0000.0000 | EPI Read FIFO Alias 1 | 423 |
| 0x078 | EPIREADFIFO2 | RO | 0x0000.0000 | EPI Read FIFO Alias 2 | 423 |
| 0x07C | EPIREADFIFO3 | RO | 0x0000.0000 | EPI Read FIFO Alias 3 | 423 |
| 0x080 | EPIREADFIFO4 | RO | 0x0000.0000 | EPI Read FIFO Alias 4 | 423 |
| 0x084 | EPIREADFIFO5 | RO | 0x0000.0000 | EPI Read FIFO Alias 5 | 423 |
| 0x088 | EPIREADFIFO6 | RO | 0x0000.0000 | EPI Read FIFO Alias 6 | 423 |
| 0x08C | EPIREADFIFO7 | RO | 0x0000.0000 | EPI Read FIFO Alias 7 | 423 |
| 0x200 | EPIFIFOLVL | R/W | 0x0000.0033 | EPI FIFO Level Selects | 424 |
| 0x204 | EPIWFIFOCNT | RO | 0x0000.0000 | EPI Write FIFO Count | 426 |
| 0x210 | EPIIM | R/W | 0x0000.0000 | EPI Interrupt Mask | 427 |
| 0x214 | EPIRIS | RO | 0x0000.0000 | EPI Raw Interrupt Status | 428 |
| 0x218 | EPIMIS | RO | 0x0000.0000 | EPI Masked Interrupt Status | 430 |
| 0x21C | EPIEISC | R/W1C | 0x0000.0000 | EPI Error Interrupt Status and Clear | 431 |

11.6 Register Descriptions

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

Register 1: EPI Configuration (EPICFG), offset 0x000

Important: The `MODE` field determines which configuration register is accessed for offsets 0x010 and 0x014. Any write to the `EPICFG` register resets the register contents at offsets 0x010 and 0x014.

The configuration register is used to enable the block, select a mode, and select the basic pin use (based on the mode). Note that attempting to program an undefined `MODE` field clears the `BLKEN` bit and disables the EPI controller.

EPI Configuration (EPICFG)

Base 0x400D.0000
Offset 0x000
Type 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 | | | | | | | | | | | | BLKEN | MODE | | | |
| 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 |
|-----------|----------|------|------------|---|
| 31:5 | 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 | BLKEN | R/W | 0 | Block Enable |
| | | | | Value Description |
| | | | | 1 The EPI controller is enabled. |
| | | | | 0 The EPI controller is disabled. |
| 3:0 | MODE | R/W | 0x0 | Mode Select |
| | | | | Value Description |
| | | | | 0x0 General Purpose |
| | | | | General-Purpose mode. Control, address, and data pins are configured using the <code>EPIGPCFG</code> and <code>EPIGPCFG2</code> registers. |
| | | | | 0x1 SDRAM |
| | | | | Supports SDR SDRAM. Control, address, and data pins are configured using the <code>EPISDRAMCFG</code> and <code>EPISDRAMCFG2</code> registers. |
| | | | | 0x2 8-Bit Host-Bus (HB8) |
| | | | | Host-bus 8-bit interface (also known as the MCU interface). Control, address, and data pins are configured using the <code>EPIHB8CFG</code> and <code>EPIHB8CFG2</code> registers. |
| | | | | 0x3-0xF Reserved |

Register 2: EPI Main Baud Rate (EPIBAUD), offset 0x004

The system clock is used internally to the EPI Controller. The baud rate counter can be used to divide the system clock down to control the speed on the external interface. If the mode selected emits an external clock, this register defines the clock emitted. If the mode selected does not use a clock, this register controls the speed of changes on the external interface. Care must be taken to program this register properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins. COUNT0 is the bit field used in all modes except in HB8 and HB16 modes with dual chip selects when different baud rates are selected, see page 409. If different baud rates are used, COUNT0 is associated with the address range specified by CS0 and COUNT1 is associated with the address range specified by CS1.

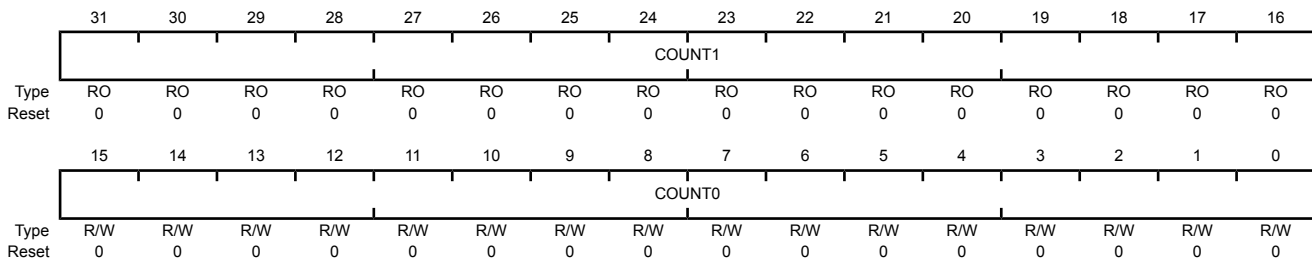
The COUNTn field is not a straight divider or count, but is instead calculated using the following formula:

$$\frac{1}{\left(\left\lfloor \frac{COUNTn}{2} \right\rfloor + 1\right)} \times 2$$

So, for example, a COUNTn of 0x0001 results in a clock rate of 1/2(system clock); a COUNTn of 0x0002 or 0x0003 results in a clock rate of 1/4(system clock).

EPI Main Baud Rate (EPIBAUD)

Base 0x400D.0000
 Offset 0x004
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|--------|--|
| 31:16 | COUNT1 | RO | 0x0000 | <p>Baud Rate Counter 1</p> <p>This bit field is only valid when the CSCFG field is 0x2 or 0x3 and the CSBAUD bit is set in the EPIHBnCFG2 register .</p> <p>This bit field contains a counter used to divide the system clock by the count. The maximum frequency for the external baud clock is 50 MHz.</p> <p>A count of 0 means the system clock is used as is.</p> |
| 15:0 | COUNT0 | R/W | 0x0000 | <p>Baud Rate Counter 0</p> <p>This bit field contains a counter used to divide the system clock by the count. The maximum frequency for the external baud clock is 50 MHz.</p> <p>A count of 0 means the system clock is used as is.</p> |

Register 3: EPI SDRAM Configuration (EPISDRAMCFG), offset 0x010

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPISDRAMCFG`, the `MODE` field must be 0x1.

The SDRAM Configuration register is used to specify several parameters for the SDRAM controller. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized.

The SDRAM interface designed to interface to x16 SDR SDRAMs of 64 MHz or higher, with the address and data pins overlapped (wire ORed on the board). See Table 11-2 on page 378 for pin assignments.

EPI SDRAM Configuration (EPISDRAMCFG)

Base 0x400D.0000

Offset 0x010

Type R/W, reset 0x42EE.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|----------|----|----|------|-------|----------|-----|-----|-----|-----|-----|------|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | FREQ | | reserved | | | RFSH | | | | | | | | | | |
| Type | R/W | R/W | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | SLEEP | reserved | | | | | | SIZE | | |
| Type | RO | RO | RO | RO | RO | RO | R/W | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | |
|-----------|-----------|------------|-------|--|-------|-----------|------------|-----|---|----|-----|----|----|-----|----|----|-----|----|-----|
| 31:30 | FREQ | R/W | 0x1 | <p>Frequency Range</p> <p>This field configures the frequency range of the system clock. This field must be configured correctly to ensure proper operation. This field does not affect the refresh counting, which is configured separately using the <code>RFSH</code> field (and is based on system clock rate and number of rows per bank). The ranges are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Low (MHz)</th> <th>High (MHz)</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>0</td> <td>15</td> </tr> <tr> <td>0x1</td> <td>15</td> <td>30</td> </tr> <tr> <td>0x2</td> <td>30</td> <td>50</td> </tr> <tr> <td>0x3</td> <td>50</td> <td>100</td> </tr> </tbody> </table> | Value | Low (MHz) | High (MHz) | 0x0 | 0 | 15 | 0x1 | 15 | 30 | 0x2 | 30 | 50 | 0x3 | 50 | 100 |
| Value | Low (MHz) | High (MHz) | | | | | | | | | | | | | | | | | |
| 0x0 | 0 | 15 | | | | | | | | | | | | | | | | | |
| 0x1 | 15 | 30 | | | | | | | | | | | | | | | | | |
| 0x2 | 30 | 50 | | | | | | | | | | | | | | | | | |
| 0x3 | 50 | 100 | | | | | | | | | | | | | | | | | |
| 29:27 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | |
| 26:16 | RFSH | R/W | 0x2EE | <p>Refresh Counter</p> <p>This field contains the refresh counter in system clocks. The reset value of 0x2EE provides a refresh period of 64 ms when using a 50 MHz clock.</p> | | | | | | | | | | | | | | | |
| 15:10 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 9 | SLEEP | R/W | 0 | Sleep Mode Value Description 1 The SDRAM is put into low power state, but is self-refreshed. 0 No effect. |
| 8:2 | 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. |
| 1:0 | SIZE | R/W | 0x0 | Size of SDRAM The value of this field affects address pins and behavior. Value Description 0x0 64Mb (8MB) 0x1 128Mb (16MB) 0x2 256Mb (32MB) 0x3 512Mb (64MB) |

Register 4: EPI Host-Bus 8 Configuration (EPIHB8CFG), offset 0x010

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPIHB8CFG`, the `MODE` field must be 0x2.

The Host Bus 8 Configuration register is activated when the HB8 mode is selected. The HB8 mode supports muxed address/data (overlay of lower 8 address and all 8 data pins), separated address/data, and address-less FIFO mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the HB8 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, CPLDs/FPGAs, and devices with an MCU/HostBus slave or 8-bit FIFO interface support.

Refer to Table 11-5 on page 383 for information on signal configuration controlled by this register and the `EPIHB8CFG2` register.

If less address pins are required, the corresponding GPIO's `AFSEL` bit (page 340) should not be enabled (so the EPI controller does not drive those pins, and they are available as standard GPIOs).

There is no direct chip enable (CE) model. Instead, CE can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally `CEn`, lower addresses are not used. For example, if pins `EPIOS27` and `EPIOS26` are used for Device 1 and 0 respectively, then address 0x6800.0000 accesses Device 0 (Device 1 has its `CEn` high), and 0x6400.0000 accesses Device 1 (Device 0 has its `CEn` high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as `CEn` because the address remains stable after the ALE strobe. The subsequent `WRn` or `RDn` signals write or read when ALE is low thus providing `CEn` functionality.

EPI Host-Bus 8 Configuration (EPIHB8CFG)

Base 0x400D.0000
Offset 0x010
Type R/W, reset 0x0000.FF00

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-------|-------|--------|--------|----------|----|------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | XFFEN | XFEEN | WRHIGH | RDHIGH | reserved | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | MAXWAIT | | | | | | | | WRWS | | RDWS | | reserved | | MODE | |
| 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 | RO | RO | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 23 | XFFEN | R/W | 0 | <p>External FIFO FULL Enable</p> <p>Value Description</p> <p>1 An external FIFO full signal can be used to control write cycles. If this bit is set and the external FIFO full signal is high, XFIFO writes are stalled.</p> <p>0 No effect.</p> |
| 22 | XFEEN | R/W | 0 | <p>External FIFO EMPTY Enable</p> <p>Value Description</p> <p>1 An external FIFO empty signal can be used to control read cycles. If this bit is set and the external FIFO empty signal is high, XFIFO reads are stalled.</p> <p>0 No effect.</p> |
| 21 | WRHIGH | R/W | 0 | <p>WRITE Strobe Polarity</p> <p>Value Description</p> <p>1 The WRITE strobe is WRn (active low).</p> <p>0 The WRITE strobe is WR (active high).</p> |
| 20 | RDHIGH | R/W | 0 | <p>READ Strobe Polarity</p> <p>Value Description</p> <p>1 The READ strobe is RDn (active low).</p> <p>0 The READ strobe is RD (active high).</p> |
| 19:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:8 | MAXWAIT | R/W | 0xFF | <p>Maximum Wait</p> <p>This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY).</p> <p>When this field is clear, the transaction is held off forever.</p> |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---|------|-------|---|-------|-------------|-----|--|-----|---|-----|--|-----|---|
| 7:6 | WRWS | R/W | 0x0 | <p>Write Wait States</p> <p>This field adds wait states to the data phase (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No wait states</td> </tr> <tr> <td>0x1</td> <td>1 wait state</td> </tr> <tr> <td>0x2</td> <td>2 wait states</td> </tr> <tr> <td>0x3</td> <td>3 wait states</td> </tr> </tbody> </table> <p>This field is used in conjunction with the EPIBAUD register.</p> | Value | Description | 0x0 | No wait states | 0x1 | 1 wait state | 0x2 | 2 wait states | 0x3 | 3 wait states |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | No wait states | | | | | | | | | | | | | |
| 0x1 | 1 wait state | | | | | | | | | | | | | |
| 0x2 | 2 wait states | | | | | | | | | | | | | |
| 0x3 | 3 wait states | | | | | | | | | | | | | |
| 5:4 | RDWS | R/W | 0x0 | <p>Read Wait States</p> <p>This field adds wait states to the data phase (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No wait states</td> </tr> <tr> <td>0x1</td> <td>1 wait state</td> </tr> <tr> <td>0x2</td> <td>2 wait states</td> </tr> <tr> <td>0x3</td> <td>3 wait states</td> </tr> </tbody> </table> <p>This field is used in conjunction with the EPIBAUD register.</p> | Value | Description | 0x0 | No wait states | 0x1 | 1 wait state | 0x2 | 2 wait states | 0x3 | 3 wait states |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | No wait states | | | | | | | | | | | | | |
| 0x1 | 1 wait state | | | | | | | | | | | | | |
| 0x2 | 2 wait states | | | | | | | | | | | | | |
| 0x3 | 3 wait states | | | | | | | | | | | | | |
| 3:2 | reserved | RO | 0x0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> | | | | | | | | | | |
| 1:0 | MODE | R/W | 0x0 | <p>Host Bus Sub-Mode</p> <p>This field determines which of four Host Bus 8 sub-modes to use. Sub-mode use is determined by the connected external peripheral. See Table 11-5 on page 383 for information on how this bit field affects the operation of the EPI signals.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>ADMUX – AD[7:0] Data and Address are muxed.</td> </tr> <tr> <td>0x1</td> <td>ADNONMUX – D[7:0] Data and address are separate.</td> </tr> <tr> <td>0x2</td> <td>SRAM This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing.</td> </tr> <tr> <td>0x3</td> <td>XFIFO – D[7:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.</td> </tr> </tbody> </table> | Value | Description | 0x0 | ADMUX – AD[7:0] Data and Address are muxed. | 0x1 | ADNONMUX – D[7:0] Data and address are separate. | 0x2 | SRAM This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. | 0x3 | XFIFO – D[7:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE. |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | ADMUX – AD[7:0] Data and Address are muxed. | | | | | | | | | | | | | |
| 0x1 | ADNONMUX – D[7:0] Data and address are separate. | | | | | | | | | | | | | |
| 0x2 | SRAM This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. | | | | | | | | | | | | | |
| 0x3 | XFIFO – D[7:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE. | | | | | | | | | | | | | |

Register 5: EPI Host-Bus 16 Configuration (EPIHB16CFG), offset 0x010

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPIHB16CFG`, the `MODE` field must be 0x3.

The Host Bus 16 sub-configuration register is activated when the HB16 mode is selected. The HB16 mode supports muxed address/data (overlay of lower 16 address and all 16 data pins), separated address/data, and address-less FIFO mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the HB16 mode is selected again, the values must be reinitialized.

This mode is intended to support SRAMs, Flash memory (read), FIFOs, and CPLDs/FPGAs, and devices with an MCU/HostBus slave or 16-bit FIFO interface support.

Refer to Table 11-6 on page 385 for information on signal configuration controlled by this register and the `EPIHB16CFG2` register.

If less address pins are required, the corresponding GPIO's `AFSEL` bit (page 340) should not be enabled (so the EPI controller does not drive those pins, and they are available as standard GPIOs).

There is no direct chip enable (CE) model. Instead, CE can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins `EPI0S27` and `EPI0S26` are used for Device 1 and 0 respectively, then address 0x6800.0000 accesses Device 0 (Device 1 has its CEn high), and 0x6400.0000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WRn or RDn signals write or read when ALE is low thus providing CEn functionality.

EPI Host-Bus 16 Configuration (EPIHB16CFG)

Base 0x400D.0000
Offset 0x010
Type R/W, reset 0x0000.FF00

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-------|-------|--------|--------|----------|------|------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | XFFEN | XFEEN | WRHIGH | RDHIGH | reserved | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | MAXWAIT | | | | | | | | WRWS | | RDWS | | reserved | BSEL | MODE | |
| 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 | RO | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 23 | XFFEN | R/W | 0 | External FIFO FULL Enable Value Description 1 An external FIFO full signal can be used to control write cycles. If this bit is set and the external FIFO full signal is high, XFIFO writes are stalled. 0 No effect. |
| 22 | XFEEN | R/W | 0 | External FIFO EMPTY Enable Value Description 1 An external FIFO empty signal can be used to control read cycles. If this bit is set and the external FIFO empty signal is high, XFIFO reads are stalled. 0 No effect. |
| 21 | WRHIGH | R/W | 0 | WRITE Strobe Polarity Value Description 1 The WRITE strobe is WRn (active low). 0 The WRITE strobe is WR (active high). |
| 20 | RDHIGH | R/W | 0 | READ Strobe Polarity Value Description 1 The READ strobe is RDn (active low). 0 The READ strobe is RD (active high). |
| 19:16 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:8 | MAXWAIT | R/W | 0xFF | Maximum Wait This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY). When this field is clear, the transaction is held off forever. |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---|------|-------|--|-------|-------------|-----|-----------------|-----|--------------------------------------|-----|---------------------|-----|---|
| 7:6 | WRWS | R/W | 0x0 | <p>Write Wait States</p> <p>This field adds wait states to the data phase (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No wait states</td> </tr> <tr> <td>0x1</td> <td>1 wait state</td> </tr> <tr> <td>0x2</td> <td>2 wait states</td> </tr> <tr> <td>0x3</td> <td>3 wait states</td> </tr> </tbody> </table> <p>This field is used in conjunction with the EPIBAUD register.</p> | Value | Description | 0x0 | No wait states | 0x1 | 1 wait state | 0x2 | 2 wait states | 0x3 | 3 wait states |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | No wait states | | | | | | | | | | | | | |
| 0x1 | 1 wait state | | | | | | | | | | | | | |
| 0x2 | 2 wait states | | | | | | | | | | | | | |
| 0x3 | 3 wait states | | | | | | | | | | | | | |
| 5:4 | RDWS | R/W | 0x0 | <p>Read Wait States</p> <p>This field adds wait states to the data phase (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No wait states</td> </tr> <tr> <td>0x1</td> <td>1 wait state</td> </tr> <tr> <td>0x2</td> <td>2 wait states</td> </tr> <tr> <td>0x3</td> <td>3 wait states</td> </tr> </tbody> </table> <p>This field is used in conjunction with the EPIBAUD register.</p> | Value | Description | 0x0 | No wait states | 0x1 | 1 wait state | 0x2 | 2 wait states | 0x3 | 3 wait states |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | No wait states | | | | | | | | | | | | | |
| 0x1 | 1 wait state | | | | | | | | | | | | | |
| 0x2 | 2 wait states | | | | | | | | | | | | | |
| 0x3 | 3 wait states | | | | | | | | | | | | | |
| 3 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> | | | | | | | | | | |
| 2 | BSEL | R/W | 0 | <p>Byte Select Configuration</p> <p>This bit enables byte select operation.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Byte Selects</td> </tr> <tr> <td></td> <td>Data is read and written as 16 bits.</td> </tr> <tr> <td>1</td> <td>Enable Byte Selects</td> </tr> <tr> <td></td> <td>Two EPI signals function as byte select signals to allow 8-bit transfers. See Table 11-6 on page 385 for details on which EPI signals are used.</td> </tr> </tbody> </table> | Value | Description | 0 | No Byte Selects | | Data is read and written as 16 bits. | 1 | Enable Byte Selects | | Two EPI signals function as byte select signals to allow 8-bit transfers. See Table 11-6 on page 385 for details on which EPI signals are used. |
| Value | Description | | | | | | | | | | | | | |
| 0 | No Byte Selects | | | | | | | | | | | | | |
| | Data is read and written as 16 bits. | | | | | | | | | | | | | |
| 1 | Enable Byte Selects | | | | | | | | | | | | | |
| | Two EPI signals function as byte select signals to allow 8-bit transfers. See Table 11-6 on page 385 for details on which EPI signals are used. | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 1:0 | MODE | R/W | 0x0 | <p>Host Bus Sub-Mode</p> <p>This field determines which of three Host Bus 16 sub-modes to use. Sub-mode use is determined by the connected external peripheral. See Table 11-6 on page 385 for information on how this bit field affects the operation of the EPI signals.</p> <p>Value Description</p> <p>0x0 ADMUX – AD[15:0] Data and Address are muxed.</p> <p>0x1 ADNONMUX – D[15:0] Data and address are separate. This mode is not practical in HB16 mode for normal peripherals because there are generally not enough address bits available.</p> <p>0x2 SRAM This mode is the same as ADNONMUX, but uses address switch for multiple reads instead of OEn strobing. This mode is not practical in HB16 mode for normal SRAMs because there are generally not enough address bits available.</p> <p>0x3 XFIFO – D[15:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.</p> |

Register 6: EPI General-Purpose Configuration (EPIGPCFG), offset 0x010

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPIGPCFG`, the `MODE` field must be 0x0.

The General-Purpose configuration register is used to control the size of control, data, and address pins, if used. This mode can be used for custom interfaces with FPGAs, CPLDs, and for digital data acquisition and actuator control. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the SDRAM mode is selected again, the register the values must be reinitialized.

This mode is designed for 3 general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs, with 3 sizes of data and optional address. Framing and clock-enable permit more optimized interfaces.
- General parallel GPIO. From 1 to 32 pins may be written or read, with the speed precisely controlled by the baud rate in the `EPIBAUD` register (when used with the NBRFIFO and/or the WFIFO) or by rate of accesses from software or μ DMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free running or gated), a framing signal (with frame size), a clock-enable input (to stretch transactions), READ and WRITE strobes, address of varying sizes, and data of varying sizes. Additionally, provisions are made for splitting address and data phases on the external interface.

EPI General-Purpose Configuration (EPIGPCFG)

Base 0x400D.0000
Offset 0x010
Type R/W, reset 0x0000.FF00

| | | | | | | | | | | | | | | | | |
|-------|---------|---------|----------|-------|--------|-------|--------|----------|-----|-------|-----|----------|--------|--------|----------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | CLKPIN | CLKGATE | reserved | RDYEN | FRMPIN | FRM50 | FRMCNT | | | | RW | reserved | WR2CYC | RD2CYC | reserved | |
| Type | R/W | R/W | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | R/W | R/W | 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 |
| | MAXWAIT | | | | | | | reserved | | ASIZE | | reserved | | DSIZE | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | R/W | R/W | RO | RO | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|-------------|
| 31 | CLKPIN | R/W | 0 | Clock Pin |

| Value | Description |
|-------|--------------------------------------|
| 1 | EPIOS31 functions as a clock output. |
| 0 | No clock output. |

The clock is generated from the `COUNT0` field in the `EPIBAUD` register (as is the system clock which is divided down from it).

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 30 | CLKGATE | R/W | 0 | <p>Clock Gated</p> <p>Value Description</p> <p>1 The clock is output only when there is data to write or read (current transaction); otherwise the clock is held low.</p> <p>0 The clock is free running.</p> <p>Note that <code>EPI0S27</code> is an <code>iRDY</code> signal if <code>RDYEN</code> is set. <code>CLKGATE</code> is ignored if <code>CLKPIN</code> is 0 or if the <code>COUNT0</code> field in the <code>EPIBAUD</code> register is cleared.</p> |
| 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 | RDYEN | R/W | 0 | <p>Ready Enable</p> <p>Value Description</p> <p>1 The external peripheral drives an <code>iRDY</code> signal into pin <code>EPI0S27</code>.</p> <p>0 The external peripheral does not drive an <code>iRDY</code> signal and is assumed to be ready always.</p> <p>The ready enable signal may only be used with a free-running clock (<code>CLKGATE=0</code>).</p> <p>The external <code>iRDY</code> signal is sampled on the rising edge of the clock. Setup and hold times must be met to ensure registration on the next rising clock edge.</p> <p>This bit is ignored if <code>CLKPIN</code> is 0 or <code>CLKGATE</code> is 1.</p> |
| 27 | FRMPIN | R/W | 0 | <p>Framing Pin</p> <p>Value Description</p> <p>1 A framing signal is output on <code>EPI0S30</code>.</p> <p>0 No framing signal.</p> <p>Framing has no impact on data itself, but forms a context for the external peripheral. When used with a free-running clock, the <code>FRAME</code> signal forms the valid signal. When used with a gated clock, it is usually used to form a frame size.</p> |
| 26 | FRM50 | R/W | 0 | <p>50/50 Frame</p> <p>Value Description</p> <p>1 The <code>FRAME</code> signal is output as 50/50 duty cycle using count (see <code>FRMCNT</code>).</p> <p>0 The <code>FRAME</code> signal is output as a single pulse, and then held low for the count.</p> <p>This bit is ignored if <code>FRMPIN</code> is 0.</p> |

| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|--|------|-------|---|---|--|---|--|
| 25:22 | FRMCNT | R/W | 0x0 | <p>Frame Count</p> <p>This field specifies the size of the frame in clocks. The frame counter is used to determine the frame size. The count is FRMCNT+1. So, a FRMCNT of 0 forms a pure transaction valid signal (held high during transactions, low otherwise).</p> <p>A FRMCNT of 0 with FRM50 set inverts the FRAME signal on each transaction. A FRMCNT of 1 means the FRAME signal is inverted every other transaction; a value of 15 means every sixteenth transaction.</p> <p>If FRM50 is set, the frame is held high for FRMCNT+1 transactions, then held low for that many transactions, and so on.</p> <p>If FRM50 is clear, the frame is pulsed high for one clock and then low for FRMCNT clocks.</p> <p>This field is ignored if FRMPIN is 0.</p> | | | | |
| 21 | RW | R/W | 0 | <p>Read and Write</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>READ and WRITE strobes are asserted on EPI0S29 and EPI0S28. READ is asserted high on the rising edge of the clock when a read is being performed. WRITE is asserted high on the rising edge of the clock when a write is being performed</td> </tr> <tr> <td>0</td> <td>READ and WRITE strobes are not output.</td> </tr> </table> <p>This bit is forced to 1 when RD2CYC and/or WR2CYC is 1.</p> | 1 | READ and WRITE strobes are asserted on EPI0S29 and EPI0S28. READ is asserted high on the rising edge of the clock when a read is being performed. WRITE is asserted high on the rising edge of the clock when a write is being performed | 0 | READ and WRITE strobes are not output. |
| 1 | READ and WRITE strobes are asserted on EPI0S29 and EPI0S28. READ is asserted high on the rising edge of the clock when a read is being performed. WRITE is asserted high on the rising edge of the clock when a write is being performed | | | | | | | |
| 0 | READ and WRITE strobes are not output. | | | | | | | |
| 20 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> | | | | |
| 19 | WR2CYC | R/W | 0 | <p>2-Cycle Writes</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>Writes are two cycles long, with address on one cycle (with the WRITE strobe asserted) and data written on the following cycle (with WRITE strobe de-asserted). The next address (if any) is in the cycle following.</td> </tr> <tr> <td>0</td> <td>Data is output on the same cycle as the address.</td> </tr> </table> <p>When this bit is set, then the RW bit is forced to be set.</p> | 1 | Writes are two cycles long, with address on one cycle (with the WRITE strobe asserted) and data written on the following cycle (with WRITE strobe de-asserted). The next address (if any) is in the cycle following. | 0 | Data is output on the same cycle as the address. |
| 1 | Writes are two cycles long, with address on one cycle (with the WRITE strobe asserted) and data written on the following cycle (with WRITE strobe de-asserted). The next address (if any) is in the cycle following. | | | | | | | |
| 0 | Data is output on the same cycle as the address. | | | | | | | |
| 18 | RD2CYC | R/W | 0 | <p>2-Cycle Reads</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>Reads are two cycles, with address on one cycle (with the READ strobe asserted) and data captured on the following cycle (with READ strobe de-asserted). The next address (if any) is in the cycle following.</td> </tr> <tr> <td>0</td> <td>Data is captured on the cycle with READ strobe asserted.</td> </tr> </table> <p>When this bit is set, then the RW bit is forced to be set.</p> | 1 | Reads are two cycles, with address on one cycle (with the READ strobe asserted) and data captured on the following cycle (with READ strobe de-asserted). The next address (if any) is in the cycle following. | 0 | Data is captured on the cycle with READ strobe asserted. |
| 1 | Reads are two cycles, with address on one cycle (with the READ strobe asserted) and data captured on the following cycle (with READ strobe de-asserted). The next address (if any) is in the cycle following. | | | | | | | |
| 0 | Data is captured on the cycle with READ strobe asserted. | | | | | | | |
| 17:16 | reserved | RO | 0x0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|--|------|-------|---|-------|-------------|-----|--|-----|--|-----|---|-----|--|
| 15:8 | MAXWAIT | R/W | 0xFF | <p>Maximum Wait</p> <p>This field defines the maximum number of external clocks to wait while an external clock-enable (see <code>RDYEN</code>) is holding off a transaction. If this field is 0, the transaction is held forever. If the maximum wait of 255 clocks (<code>MAXWAIT=0xFF</code>) is exceeded, an error interrupt occurs and the transaction is aborted/ignored.</p> | | | | | | | | | | |
| 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 | ASIZE | R/W | 0x0 | <p>Address Bus Size</p> <p>This field defines the size of the address bus (starting at <code>EPIOS8</code>, <code>EPIOS16</code>, or <code>EPIOS24</code>, depending on size). Subsets of these numbers can be created by clearing the <code>AFSEL</code> bit for the corresponding GPIOs. Also, if <code>RDYEN</code> is 1, then the address sizes are 1 smaller (3, 11, 19).</p> <p>The values are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No address</td> </tr> <tr> <td>0x1</td> <td>4 Bits Wide (<code>EPIOS24</code> to <code>EPIOS27</code>)</td> </tr> <tr> <td>0x2</td> <td>12 Bits Wide (<code>EPIOS16</code> to <code>EPIOS27</code>). This size cannot be used with 24-bit data.</td> </tr> <tr> <td>0x3</td> <td>20 Bits Wide (<code>EPIOS8</code> to <code>EPIOS27</code>). This size cannot be used with data sizes other than 8.</td> </tr> </tbody> </table> | Value | Description | 0x0 | No address | 0x1 | 4 Bits Wide (<code>EPIOS24</code> to <code>EPIOS27</code>) | 0x2 | 12 Bits Wide (<code>EPIOS16</code> to <code>EPIOS27</code>). This size cannot be used with 24-bit data. | 0x3 | 20 Bits Wide (<code>EPIOS8</code> to <code>EPIOS27</code>). This size cannot be used with data sizes other than 8. |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | No address | | | | | | | | | | | | | |
| 0x1 | 4 Bits Wide (<code>EPIOS24</code> to <code>EPIOS27</code>) | | | | | | | | | | | | | |
| 0x2 | 12 Bits Wide (<code>EPIOS16</code> to <code>EPIOS27</code>). This size cannot be used with 24-bit data. | | | | | | | | | | | | | |
| 0x3 | 20 Bits Wide (<code>EPIOS8</code> to <code>EPIOS27</code>). This size cannot be used with data sizes other than 8. | | | | | | | | | | | | | |
| 3:2 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 1:0 | DSIZE | R/W | 0x0 | <p>Size of Data Bus</p> <p>This field defines the size of the data bus (starting at <code>EPIOS0</code>). Subsets of these numbers can be created by clearing the <code>AFSEL</code> bit for the corresponding GPIOs. Note that size 32 may not be used with clock, frame, address, or other control.</p> <p>The values are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>8 Bits Wide (<code>EPIOS0</code> to <code>EPIOS7</code>)</td> </tr> <tr> <td>0x1</td> <td>16 Bits Wide (<code>EPIOS0</code> to <code>EPIOS15</code>)</td> </tr> <tr> <td>0x2</td> <td>24 Bits Wide (<code>EPIOS0</code> to <code>EPIOS23</code>)</td> </tr> <tr> <td>0x3</td> <td>32 Bits Wide (<code>EPIOS0</code> to <code>EPIOS31</code>). This size may not be used with a clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction.</td> </tr> </tbody> </table> | Value | Description | 0x0 | 8 Bits Wide (<code>EPIOS0</code> to <code>EPIOS7</code>) | 0x1 | 16 Bits Wide (<code>EPIOS0</code> to <code>EPIOS15</code>) | 0x2 | 24 Bits Wide (<code>EPIOS0</code> to <code>EPIOS23</code>) | 0x3 | 32 Bits Wide (<code>EPIOS0</code> to <code>EPIOS31</code>). This size may not be used with a clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction. |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | 8 Bits Wide (<code>EPIOS0</code> to <code>EPIOS7</code>) | | | | | | | | | | | | | |
| 0x1 | 16 Bits Wide (<code>EPIOS0</code> to <code>EPIOS15</code>) | | | | | | | | | | | | | |
| 0x2 | 24 Bits Wide (<code>EPIOS0</code> to <code>EPIOS23</code>) | | | | | | | | | | | | | |
| 0x3 | 32 Bits Wide (<code>EPIOS0</code> to <code>EPIOS31</code>). This size may not be used with a clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction. | | | | | | | | | | | | | |

Register 7: EPI SDRAM Configuration 2 (EPISDRAMCFG2), offset 0x014

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPISDRAMCFG2`, the `MODE` field must be 0x1.

The SDRAM Configuration 2 register is used to control when data is captured on read cycles. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized.

EPI SDRAM Configuration 2 (EPISDRAMCFG2)

Base 0x400D.0000
Offset 0x014
Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | RCM | reserved | | | | | | | | | | | | | | |
| Type | R/W | 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 | | | | | | | | | | | | | | | |
| 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 | RCM | R/W | 0 | Read Capture Mode |
| | | | | Value Description |
| | | | | 0 Read data is captured on the rising edge of the SDRAM clock (EPI0S31). |
| | | | | 1 Read data is captured on the rising edge of the feedback SDRAM clock (EPI0S31). This mode adds one clock of internal latency on the first read access (not SDRAM latency). |
| 30:0 | 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 8: EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2), offset 0x014

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPIHB8CFG2`, the `MODE` field must be 0x2.

This register is used to configure operation while in Host-Bus 8 mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the Host-Bus 8 mode is selected again, the values must be reinitialized.

EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2)

Base 0x400D.0000

Offset 0x014

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----|--------|-------|-----|----------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | WORD | reserved | | | | CSBAUD | CSCFG | | reserved | | | | | | | |
| Type | R/W | RO | RO | RO | RO | R/W | R/W | R/W | 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 | | | | | | | | | | | | | | | |
| 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 | WORD | R/W | 0 | <p>Word Access Mode</p> <p>By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8].</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Word Access mode is disabled.</td> </tr> <tr> <td>1</td> <td>Word Access mode is enabled.</td> </tr> </table> | Value | Description | 0 | Word Access mode is disabled. | 1 | Word Access mode is enabled. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Word Access mode is disabled. | | | | | | | | | | | | | |
| 1 | Word Access mode is enabled. | | | | | | | | | | | | | |
| 30:27 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 26 | CSBAUD | R/W | 0 | <p>Chip Select Baud Rate</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Same Baud Rate</td> </tr> <tr> <td colspan="2">Both CS0n and CS1n use the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register.</td> </tr> <tr> <td>1</td> <td>Different Baud Rates</td> </tr> <tr> <td colspan="2">CS0n uses the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. CSn1 uses the baud rate defined by the <code>COUNT1</code> field in the <code>EPIBAUD</code> register.</td> </tr> </table> | Value | Description | 0 | Same Baud Rate | Both CS0n and CS1n use the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. | | 1 | Different Baud Rates | CS0n uses the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. CSn1 uses the baud rate defined by the <code>COUNT1</code> field in the <code>EPIBAUD</code> register. | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Same Baud Rate | | | | | | | | | | | | | |
| Both CS0n and CS1n use the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. | | | | | | | | | | | | | | |
| 1 | Different Baud Rates | | | | | | | | | | | | | |
| CS0n uses the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. CSn1 uses the baud rate defined by the <code>COUNT1</code> field in the <code>EPIBAUD</code> register. | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 25:24 | CSCFG | R/W | 0x0 | <p>Chip Select Configuration</p> <p>Value Description</p> <p>0x0 ALE Configuration</p> <p>EPIOS30 is used as an address latch (ALE). When using this mode, the address and data should be muxed (HB8MODE field in the EPIHB8CFG register should be configured to 0x0). If needed, the address can be latched by external logic.</p> <p>0x1 CSn Configuration</p> <p>EPIOS30 is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (HB8MODE field in the EPIHB8CFG register should be configured to 0x1). In this mode, the WR signal (EPIOS29) and the RD signal (EPIOS28) are used to latch the address when CSn is low.</p> <p>0x2 Dual CSn Configuration</p> <p>EPIOS30 is used as CS0n and EPIOS27 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.</p> <p>0x3 ALE with Dual CSn Configuration</p> <p>EPIOS30 is used as address latch (ALE), EPIOS27 is used as CS0n, and EPIOS26 is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.</p> |
| 23:0 | reserved | RO | 0x00.0000 | 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 9: EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2), offset 0x014

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPIHB16CFG2`, the `MODE` field must be 0x3.

This register is used to configure operation while in Host-Bus 16 mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the Host-Bus 16 mode is selected again, the values must be reinitialized.

EPI Host-Bus 16 Configuration 2 (EPIHB16CFG2)

Base 0x400D.0000

Offset 0x014

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----|--------|-------|-----|----|----------|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | WORD | reserved | | | | CSBAUD | CSCFG | | | reserved | | | | | | |
| Type | R/W | RO | RO | RO | RO | R/W | R/W | R/W | 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 | | | | | | | | | | | | | | | |
| 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 | WORD | R/W | 0 | <p>Word Access Mode</p> <p>By default, the EPI controller uses data bits [15:0] for Host-Bus 16 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:16].</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Word Access mode is disabled.</td> </tr> <tr> <td>1</td> <td>Word Access mode is enabled.</td> </tr> </table> | Value | Description | 0 | Word Access mode is disabled. | 1 | Word Access mode is enabled. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Word Access mode is disabled. | | | | | | | | | | | | | |
| 1 | Word Access mode is enabled. | | | | | | | | | | | | | |
| 30:27 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 26 | CSBAUD | R/W | 0 | <p>Chip Select Baud Rate</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Same Baud Rate</td> </tr> <tr> <td colspan="2">Both CS0n and CS1n use the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register.</td> </tr> <tr> <td>1</td> <td>Different Baud Rates</td> </tr> <tr> <td colspan="2">CS0n uses the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. CS1n uses the baud rate defined by the <code>COUNT1</code> field in the <code>EPIBAUD</code> register.</td> </tr> </table> | Value | Description | 0 | Same Baud Rate | Both CS0n and CS1n use the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. | | 1 | Different Baud Rates | CS0n uses the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. CS1n uses the baud rate defined by the <code>COUNT1</code> field in the <code>EPIBAUD</code> register. | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Same Baud Rate | | | | | | | | | | | | | |
| Both CS0n and CS1n use the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. | | | | | | | | | | | | | | |
| 1 | Different Baud Rates | | | | | | | | | | | | | |
| CS0n uses the baud rate for the external bus that is defined by the <code>COUNT0</code> field in the <code>EPIBAUD</code> register. CS1n uses the baud rate defined by the <code>COUNT1</code> field in the <code>EPIBAUD</code> register. | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|--|
| 25:24 | CSCFG | R/W | 0x0 | <p>Chip Select Configuration</p> <p>This field controls the chip select options, including an ALE format and a chip select format.</p> <p>Value Description</p> <p>0x0 ALE Configuration</p> <p><i>EPI0S30</i> is used as an address latch (ALE). When using this mode, the address and data should be muxed (<i>HB16MODE</i> field in the EPIHB16CFG register should be configured to 0x0). If needed, the address can be latched by external logic.</p> <p>0x1 CSn Configuration</p> <p><i>EPI0S30</i> is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (<i>HB16MODE</i> field in the EPIHB16CFG register should be configured to 0x1). In this mode, the WR signal (<i>EPI0S29</i>) and the RD signal (<i>EPI0S28</i>) are used to latch the address when CSn is low.</p> <p>0x2 Dual CSn Configuration</p> <p><i>EPI0S30</i> is used as CS0n and <i>EPI0S27</i> is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map. This configuration can be used for a RAM bank split between 2 devices as well as when using both an external RAM and an external peripheral.</p> <p>0x3 ALE with Dual CSn Configuration</p> <p><i>EPI0S30</i> is used as address latch (ALE), <i>EPI0S27</i> is used as CS0n, and <i>EPI0S26</i> is used as CS1n. Whether CS0n or CS1n is asserted is determined by the most significant address bit for a respective external address map.</p> |
| 23:0 | reserved | RO | 0x00.0000 | 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 10: EPI General-Purpose Configuration 2 (EPIGPCFG2), offset 0x014

Important: The `MODE` field in the `EPICFG` register determines which configuration register is accessed for offsets 0x010 and 0x014.

To access `EPIGPCFG2`, the `MODE` field must be 0x0.

This register is used to configure operation while in General-Purpose mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the General-Purpose mode is selected again, the values must be reinitialized.

EPI General-Purpose Configuration 2 (EPIGPCFG2)

Base 0x400D.0000

Offset 0x014

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | WORD | reserved | | | | | | | | | | | | | | |
| Type | R/W | 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 | | | | | | | | | | | | | | | |
| 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 | WORD | R/W | 0x0 | <p>Word Access Mode</p> <p>By default, the EPI controller uses data bits [7:0] when the <code>DSIZE</code> field in the <code>EPIGPCFG</code> register is 0x0; data bits [15:0] when the <code>DSIZE</code> field is 0x1; data bits [23:0] when the <code>DSIZE</code> field is 0x2; and data bits [31:0] when the <code>DSIZE</code> field is 0x3.</p> <p>When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8] for <code>DSIZE</code>=0x0 and bits [31:16] for <code>DSIZE</code>=0x1. For <code>DSIZE</code>=0x2 or 0x3, this bit must be clear.</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Word Access mode is disabled.</td> </tr> <tr> <td>1</td> <td>Word Access mode is enabled.</td> </tr> </table> | Value | Description | 0 | Word Access mode is disabled. | 1 | Word Access mode is enabled. |
| Value | Description | | | | | | | | | |
| 0 | Word Access mode is disabled. | | | | | | | | | |
| 1 | Word Access mode is enabled. | | | | | | | | | |
| 30:0 | 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 11: EPI Address Map (EPIADDRMAP), offset 0x01C

This register enables address mapping. The EPI controller can directly address memory and peripherals. In addition, the EPI controller supports address mapping to allow indirect accesses in the External RAM and External Peripheral areas.

If the external device is a peripheral, including a FIFO or a directly addressable device, the `EPSZ` and `EPADR` bit fields should be configured for the address space. If the external device is SDRAM, SRAM, NOR or NAND Flash memory, the `ERADR` and `ERSZ` bit fields should be configured for the address space.

If one of the Dual-Chip-Selects modes are used (`CSCCFG=0x2` or `0x3` in the `EPIHBnCFG2` register), both chip selects can share the peripheral or the memory space, or one chip select can use the peripheral space and the other can use the memory space. If the `EPADR` field is not `0x0` and the `ERADR` field is `0x0`, then the address specified by `EPADR` is used for both chip selects, with `CS0n` being asserted when the MSB of the address range is 0 and `CS1n` being asserted when the MSB of the address range is 1. If the `ERADR` field is not `0x0` and the `EPADR` field is `0x0`, then the address specified by `ERADR` is used for both chip selects, with the MSB performing the same delineation. If both the `EPADR` and the `ERADR` are not `0x0`, then `CS0n` is asserted for the address range defined by `EPADR` and `CS1n` is asserted for the address range defined by `ERADR`.

EPI Address Map (EPIADDRMAP)

Base 0x400D.0000
 Offset 0x01C
 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 | | | | | | | | EPSZ | | EPADR | | ERSZ | | ERADR | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---|------|-----------|---|-------|-------------|-----|--|-----|--|-----|--|-----|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 7:6 | EPSZ | R/W | 0x0 | External Peripheral Size This field selects the size of the external peripheral. If the size of the external peripheral is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused): <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0x0</td> <td>256 bytes; lower address range: 0x00 to 0xFF</td> </tr> <tr> <td>0x1</td> <td>64 KB; lower address range: 0x0000 to 0xFFFF</td> </tr> <tr> <td>0x2</td> <td>16 MB; lower address range: 0x00.0000 to 0xFF.FFFF</td> </tr> <tr> <td>0x3</td> <td>256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF</td> </tr> </table> | Value | Description | 0x0 | 256 bytes; lower address range: 0x00 to 0xFF | 0x1 | 64 KB; lower address range: 0x0000 to 0xFFFF | 0x2 | 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF | 0x3 | 256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | 256 bytes; lower address range: 0x00 to 0xFF | | | | | | | | | | | | | |
| 0x1 | 64 KB; lower address range: 0x0000 to 0xFFFF | | | | | | | | | | | | | |
| 0x2 | 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF | | | | | | | | | | | | | |
| 0x3 | 256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|---|------|-------|---|-------|-------------|-----|--|-----|--|-----|--|-----|---|
| 5:4 | EPADR | R/W | 0x0 | <p>External Peripheral Address</p> <p>This field selects address mapping for the external peripheral area:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Not mapped</td> </tr> <tr> <td>0x1</td> <td>At 0xA000.0000</td> </tr> <tr> <td>0x2</td> <td>At 0xC000.0000</td> </tr> <tr> <td>0x3</td> <td>reserved</td> </tr> </tbody> </table> | Value | Description | 0x0 | Not mapped | 0x1 | At 0xA000.0000 | 0x2 | At 0xC000.0000 | 0x3 | reserved |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Not mapped | | | | | | | | | | | | | |
| 0x1 | At 0xA000.0000 | | | | | | | | | | | | | |
| 0x2 | At 0xC000.0000 | | | | | | | | | | | | | |
| 0x3 | reserved | | | | | | | | | | | | | |
| 3:2 | ERSZ | R/W | 0x0 | <p>External RAM Size</p> <p>This field selects the size of mapped RAM. If the size of the external memory is larger, a bus fault occurs. If the size of the external memory is smaller, it wraps (upper address bits unused):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>256 bytes; lower address range: 0x00 to 0xFF</td> </tr> <tr> <td>0x1</td> <td>64 KB; lower address range: 0x0000 to 0xFFFF</td> </tr> <tr> <td>0x2</td> <td>16 MB; lower address range: 0x00.0000 to 0xFF.FFFF</td> </tr> <tr> <td>0x3</td> <td>256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF</td> </tr> </tbody> </table> | Value | Description | 0x0 | 256 bytes; lower address range: 0x00 to 0xFF | 0x1 | 64 KB; lower address range: 0x0000 to 0xFFFF | 0x2 | 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF | 0x3 | 256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | 256 bytes; lower address range: 0x00 to 0xFF | | | | | | | | | | | | | |
| 0x1 | 64 KB; lower address range: 0x0000 to 0xFFFF | | | | | | | | | | | | | |
| 0x2 | 16 MB; lower address range: 0x00.0000 to 0xFF.FFFF | | | | | | | | | | | | | |
| 0x3 | 256 MB; lower address range: 0x000.0000 to 0xFFF.FFFF | | | | | | | | | | | | | |
| 1:0 | ERADR | R/W | 0x0 | <p>External RAM Address</p> <p>Selects address mapping for external RAM area:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Not mapped</td> </tr> <tr> <td>0x1</td> <td>At 0x6000.0000</td> </tr> <tr> <td>0x2</td> <td>At 0x8000.0000</td> </tr> <tr> <td>0x3</td> <td>reserved</td> </tr> </tbody> </table> | Value | Description | 0x0 | Not mapped | 0x1 | At 0x6000.0000 | 0x2 | At 0x8000.0000 | 0x3 | reserved |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Not mapped | | | | | | | | | | | | | |
| 0x1 | At 0x6000.0000 | | | | | | | | | | | | | |
| 0x2 | At 0x8000.0000 | | | | | | | | | | | | | |
| 0x3 | reserved | | | | | | | | | | | | | |

Register 12: EPI Read Size 0 (EPIRSIZE0), offset 0x020

Register 13: EPI Read Size 1 (EPIRSIZE1), offset 0x030

This register selects the size of transactions when performing non-blocking reads with the **EPIRPSTD** registers. This size affects how the external address is incremented.

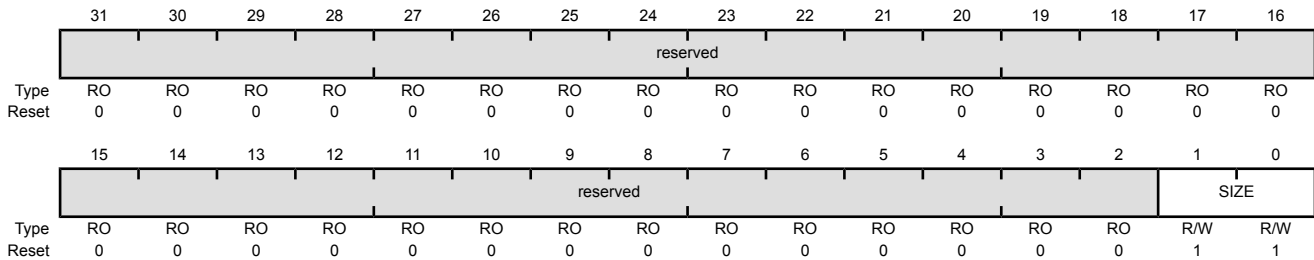
The **SIZE** field must match the external data width as configured in the **EPIHBnCFG** or **EPIGPCFG** register.

SDRAM mode uses a 16-bit data interface. If **SIZE** is 0x1, data is returned on the least significant bits (D[7:0]), and the remaining bits D[31:8] are all zeros, therefore the data on bits D[15:8] is lost. If **SIZE** is 0x2, data is returned on the least significant bits (D[15:0]), and the remaining bits D[31:16] are all zeros.

Note that changing this register while a read is active has an unpredictable effect.

EPI Read Size 0 (EPIRSIZE0)

Base 0x400D.0000
 Offset 0x020
 Type R/W, reset 0x0000.0003



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1:0 | SIZE | R/W | 0x3 | Current Size |
| | | | | Value Description |
| | | | | 0x0 reserved |
| | | | | 0x1 Byte (8 bits) |
| | | | | 0x2 Half-word (16 bits) |
| | | | | 0x3 Word (32 bits) |

Register 14: EPI Read Address 0 (EPIRADDR0), offset 0x024**Register 15: EPI Read Address 1 (EPIRADDR1), offset 0x034**

This register holds the current address value. When performing non-blocking reads via the **EPIRSTD** registers, this register's value forms the address (when used by the mode). That is, when a **EPIRSTD** register is written with a non-0 value, this register is used as the first address. After each read, it is incremented by the size specified by the corresponding **EPIRSIZE** register. Thus at the end of a read, this register contains the next address for the next read. For example, if the last read was 0x20, and the size is word, then the register contains 0x24. When a non-blocking read is cancelled, this register contains the next address that would have been read had it not been cancelled. For example, if reading by bytes and 0x103 had been read but not 0x104, this register contains 0x104. In this manner, the system can determine the number of values in the NBRFIFO to drain.

Note that changing this register while a read is active has an unpredictable effect due to race condition.

EPI Read Address 0 (EPIRADDR0)

Base 0x400D.0000

Offset 0x024

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | ADDR | | | | | | | | | | | | |
| Type | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ADDR | | | | | | | | | | | | | | | |
| 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 |
|-----------|----------|------|------------|---|
| 31:29 | 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. |
| 28:0 | ADDR | R/W | 0x000.0000 | Current Address Next address to read. |

Register 16: EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028

Register 17: EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038

This register sets up a non-blocking read via the external interface. A non-blocking read is started by writing to this register with the count (other than 0). Clearing this register terminates an active non-blocking read as well as cancelling any that are pending. This register should always be cleared before writing a value other than 0; failure to do so can cause improper operation.

The first address is based on the corresponding **EPIRADDR** register. The address register is incremented by the size specified by the **EPIRSIZE** register after each read. If the size is less than a word, only the least significant bits of data are filled into the NBRFIFO; the most significant bits are cleared.

Note that all three registers may be written using one STM instruction, such as with a structure copy in C/C++.

The data may be read from the **EPIREADFIFO** register after the read cycle is completed. The interrupt mechanism is normally used to trigger the FIFO reads via ISR or μ DMA.

If the countdown has not reached 0 and the NBRFIFO is full, the external interface waits until a NBRFIFO entry becomes available to continue.

Note: if a blocking read or write is performed through the address mapped area (at 0x6000.0000 through 0xCFFF.FFFF), any current non-blocking read is paused (at the next safe boundary), and the blocking request is inserted. After completion of any blocking reads or writes, the non-blocking reads continue from where they were paused.

The other way to read data is via the address mapped locations (see the **EPIADDRMAP** register), but this method is blocking (core or μ DMA waits until result is returned).

To cancel a non-blocking read, clear this register. To make sure that all values read are drained from the NBRFIFO, the **EPISTAT** register must be consulted to be certain that bits **NBRBUSY** and **ACTIVE** are cleared. One of these registers should not be cleared until either the other **EPIRPSTDx** register becomes active or the external interface is not busy. At that point, the corresponding **EPIRADDR** register indicates how many values were read.

EPI Non-Blocking Read Data 0 (EPIRPSTD0)

Base 0x400D.0000
 Offset 0x028
 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 | | | POSTCNT | | | | | | | | | | | | |
| Type | RO | RO | RO | 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 |
|-----------|----------|------|----------|---|
| 31:13 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 12:0 | POSTCNT | R/W | 0x000 | <p>Post Count</p> <p>A write of a non-zero value starts a read operation for that count. Note that it is the software's responsibility to handle address wraparound.</p> <p>Reading this register provides the current count.</p> <p>A write of 0 cancels a non-blocking read (whether active now or pending).</p> <p>Prior to writing a non-zero value, this register must first be cleared.</p> |

Register 18: EPI Status (EPISTAT), offset 0x060

This register indicates which non-blocking read register is currently active; it also indicates whether the external interface is busy performing a write or non-blocking read (it cannot be performing a blocking read, as the bus would be blocked and as a result, this register could not be accessed).

This register is useful for determining which non-blocking read register is active when both are loaded with values and when implementing sequencing or sharing.

This register is also useful when canceling non-blocking reads, as it shows how many values were read by the canceled side.

EPI Status (EPISTAT)

Base 0x400D.0000
 Offset 0x060
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|-------|--------|---------|---------|-------|---------|----------|----|----|--------|
| | 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 | | | | | | CELOW | XFFULL | XFEMPTY | INITSEQ | WBUSY | NBRBUSY | reserved | | | ACTIVE |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:10 | reserved | RO | 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 | CELOW | RO | 0 | <p>Clock Enable Low</p> <p>This bit provides information on the clock status when in general-purpose mode and the RDYEN bit is set.</p> <p>Value Description</p> <p>1 The external device is gating the clock (iRDY is low). Attempts to read or write in this situation are stalled until the clock is enabled or the counter times out as specified by the MAXWAIT field.</p> <p>0 The external device is not gating the clock.</p> |
| 8 | XFFULL | RO | 0 | <p>External FIFO Full</p> <p>This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the XFFEN bit set in the EPIHBnCFG register. The EPIOS26 signal reflects the status of this bit.</p> <p>Value Description</p> <p>1 The XFIFO is signaling as full (the FIFO full signal is high). Attempts to write in this case are stalled until the XFIFO full signal goes low or the counter times out as specified by the MAXWAIT field.</p> <p>0 The external device is not gating the clock.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | XFEMPTY | RO | 0 | <p>External FIFO Empty</p> <p>This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus n mode with the <i>XFEEEN</i> bit set in the EPIHBnCFG register. The <i>EPIOS27</i> signal reflects the status of this bit.</p> <p>Value Description</p> <p>1 The XFIFO is signaling as empty (the FIFO empty signal is high). Attempts to read in this case are stalled until the XFIFO empty signal goes low or the counter times out as specified by the <i>MAXWAIT</i> field.</p> <p>0 The external device is not gating the clock.</p> |
| 6 | INITSEQ | RO | 0 | <p>Initialization Sequence</p> <p>Value Description</p> <p>1 The SDRAM interface is running through the wakeup period (greater than 100 μs). If an attempt is made to read or write the SDRAM during this period, the access is held off until the wakeup period is complete.</p> <p>0 The SDRAM interface is not in the wakeup period.</p> |
| 5 | WBUSY | RO | 0 | <p>Write Busy</p> <p>Value Description</p> <p>1 The external interface is performing a write.</p> <p>0 The external interface is not performing a write.</p> |
| 4 | NBRBUSY | RO | 0 | <p>Non-Blocking Read Busy</p> <p>Value Description</p> <p>1 The external interface is performing a non-blocking read, or if the non-blocking read is paused due to a write.</p> <p>0 The external interface is not performing a non-blocking read.</p> |
| 3:1 | reserved | RO | 0x0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 0 | ACTIVE | RO | 0 | <p>Register Active</p> <p>Value Description</p> <p>1 The EPIRPSTD1 register is active.</p> <p>0 If <i>NBRBUSY</i> is set, the EPIRPSTD0 register is active. If the <i>NBRBUSY</i> bit is clear, then neither EPIRPSTDx register is active.</p> |

Register 19: EPI Read FIFO Count (EPIRFIFOCNT), offset 0x06C

This register returns the number of values in the NBRFIFO (the data in the NBRFIFO can be read via the **EPIREADFIFO** register). A race is possible, but that only means that more values may come in after this register has been read.

EPI Read FIFO Count (EPIRFIFOCNT)

Base 0x400D.0000

Offset 0x06C

Type RO, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-------|----|----|
| | 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 | | | | | | | | | | | | | COUNT | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | - | - |

| 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 | COUNT | RO | - | FIFO Count Number of filled entries in the NBRFIFO. |

Register 20: EPI Read FIFO (EPIREADFIFO), offset 0x070

Register 21: EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074

Register 22: EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078

Register 23: EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C

Register 24: EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080

Register 25: EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084

Register 26: EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088

Register 27: EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C

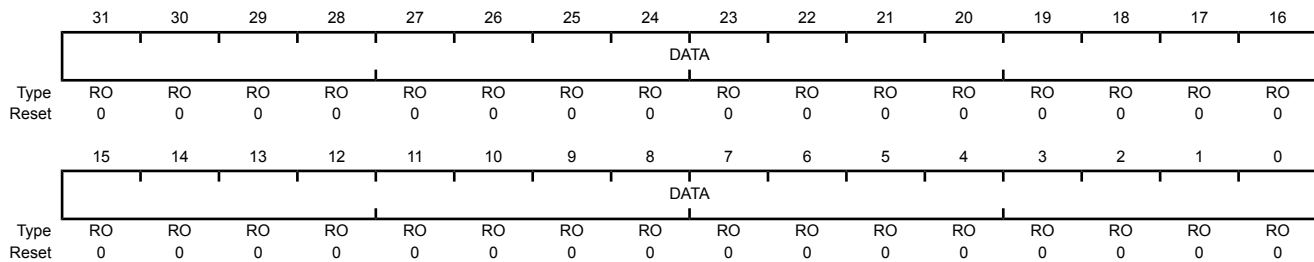
This register returns the contents of the NBRFIFO or 0 if the NBRFIFO is empty. Each read returns the data that is at the top of the NBRFIFO, and then empties that value from the NBRFIFO. The alias registers can be used with the LDMIA instruction for more efficient operation (for up to 8 registers). See *ARM® Cortex™-M3 Technical Reference Manual* for more information on the LDMIA instruction.

EPI Read FIFO (EPIREADFIFO)

Base 0x400D.0000

Offset 0x070

Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------------|-------------|
| 31:0 | DATA | RO | 0x0000.0000 | Reads Data |

This field contains the data that is at the top of the NBRFIFO. After being read, the NBRFIFO entry is removed.

Register 28: EPI FIFO Level Selects (EPIFIFOLVL), offset 0x200

This register allows selection of the FIFO levels which trigger an interrupt to the interrupt controller or, more efficiently, a DMA request to the μ DMA. The NBRFIFO select triggers on fullness such that it triggers on match or above (more full). The WFIFO triggers on emptiness such that it triggers on match or below (less entries).

It should be noted that the FIFO triggers are not identical to other such FIFOs in Stellaris[®] peripherals. In particular, empty and full triggers are provided to avoid wait states when using blocking operations.

The settings in this register are only meaningful if the μ DMA is active or the interrupt is enabled.

Additionally, this register allows protection against writes stalling and notification of performing blocking reads which stall for extra time due to preceding writes. The two functions behave in a non-orthogonal way because read and write are not orthogonal.

The write error bit configures the system such that an attempted write to an already full WFIFO abandons the write and signals an error interrupt to prevent accidental latencies due to stalling writes.

The read error bit configures the system such that after a read has been stalled due to any preceding writes in the WFIFO, the error interrupt is generated. Note that the excess stall is not prevented, but an interrupt is generated after the fact to notify that it has happened.

EPI FIFO Level Selects (EPIFIFOLVL)

Base 0x400D.0000
 Offset 0x200
 Type R/W, reset 0x0000.0033

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|--------|----|-----|----------|--------|----|-----|-------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | WFERR | RSERR |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | 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 | | | | | | | WRFIFO | | | reserved | RDFIFO | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 | 1 | 1 | 0 | 0 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:18 | 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. |
| 17 | WFERR | R/W | 0 | Write Full Error |
| | | | | Value Description |
| | | | | 1 This bit enables the Write Full error interrupt (WTFULL in the EPIIC register) to be generated when a write is attempted and the WFIFO is full. The write stalls until a WFIFO entry becomes available. |
| | | | | 0 The Write Full error interrupt is disabled. Writes are stalled when the WFIFO is full until a space becomes available but an error is not generated. Note that the Cortex-M3 write buffer may hide that stall if no other memory transactions are attempted during that time. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 16 | RSERR | R/W | 0 | <p>Read Stall Error</p> <p>Value Description</p> <p>1 This bit enables the Read Stalled error interrupt (<code>RSTALL</code> in the EPIIC register) to be generated when a read is attempted and the WFIFO is not empty. The read is still stalled during the time the WFIFO drains, but this error notifies the application that this excess delay has occurred.</p> <p>0 The Read Stalled error interrupt is disabled. Reads behave as normal and are stalled until any preceding writes have completed and the read has returned a result.</p> <p>Note that the configuration of this bit has no effect on non-blocking reads.</p> |
| 15:7 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:4 | WRFIFO | R/W | 0x3 | <p>Write FIFO</p> <p>This field configures the trigger point for the WFIFO.</p> <p>Value Description</p> <p>0x0 Trigger when there are 1 to 4 spaces available in the WFIFO.</p> <p>0x1 reserved</p> <p>0x2 Trigger when there are 1 to 3 spaces available in the WFIFO.</p> <p>0x3 Trigger when there are 1 to 2 spaces available in the WFIFO.</p> <p>0x4 Trigger when there is 1 space available in the WFIFO.</p> <p>0x5-0x7 reserved</p> |
| 3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2:0 | RDFIFO | R/W | 0x3 | <p>Read FIFO</p> <p>This field configures the trigger point for the NBRFIFO.</p> <p>Value Description</p> <p>0x0 reserved</p> <p>0x1 Trigger when there are 1 or more entries in the NBRFIFO.</p> <p>0x2 Trigger when there are 2 or more entries in the NBRFIFO.</p> <p>0x3 Trigger when there are 4 or more entries in the NBRFIFO.</p> <p>0x4 Trigger when there are 6 or more entries in the NBRFIFO.</p> <p>0x5 Trigger when there are 7 or more entries in the NBRFIFO.</p> <p>0x6 Trigger when there are 8 entries in the NBRFIFO.</p> <p>0x7 reserved</p> |

Register 29: EPI Write FIFO Count (EPIWFIFOCNT), offset 0x204

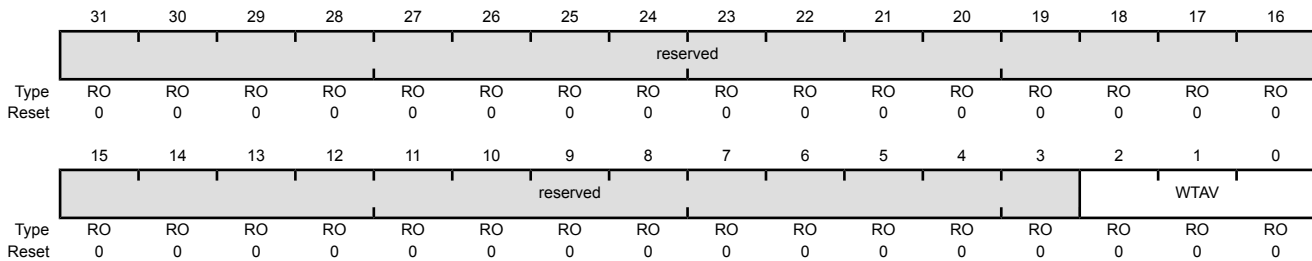
This register contains the number of slots currently available in the WFIFO. This register may be used for polled writes to avoid stalling and for blocking reads to avoid excess stalling (due to undrained writes). An example use for writes may be:

```
for (idx = 0; idx < cnt; idx++) {
while (EPIWFIFOCNT == 0) ;
*ext_ram = *mydata++;
}
```

The above code ensures that writes to the address mapped location do not occur unless the WFIFO has room. Although polling makes the code wait (spinning in the loop), it does not prevent interrupts being serviced due to bus stalling.

EPI Write FIFO Count (EPIWFIFOCNT)

Base 0x400D.0000
 Offset 0x204
 Type RO, reset 0x0000.0000



| 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 | WTAV | RO | 0x0 | Available Write Transactions The number of write transactions available in the WFIFO. When clear, a write is stalled waiting for a slot to become free (from a preceding write completing). |

Register 30: EPI Interrupt Mask (EPIIM), offset 0x210

This register is the interrupt mask set or clear register. For each interrupt source (read, write, and error), a mask value of 1 allows the interrupt source to trigger an interrupt to the interrupt controller; a mask value of 0 prevents the interrupt source from triggering an interrupt.

Note that interrupt masking has no effect on μ DMA, which operates off the raw source of the read and write interrupts.

EPI Interrupt Mask (EPIIM)

Base 0x400D.0000

Offset 0x210

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 | | | | | | | | | | | | | WRIM | RDIM | ERRIM | |
| Type | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | WRIM | R/W | 0 | Write Interrupt Mask Value Description 1 WRRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller. 0 WRRIS in the EPIRIS register is masked and does not cause an interrupt. |
| 1 | RDIM | R/W | 0 | Read Interrupt Mask Value Description 1 RDRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller. 0 RDRIS in the EPIRIS register is masked and does not cause an interrupt. |
| 0 | ERRIM | R/W | 0 | Error Interrupt Mask Value Description 1 ERRIS in the EPIRIS register is not masked and can trigger an interrupt to the interrupt controller. 0 ERRIS in the EPIRIS register is masked and does not cause an interrupt. |

Register 31: EPI Raw Interrupt Status (EPIRIS), offset 0x214

This register is the raw interrupt status register. On a read, it gives the current state of each interrupt source. A write has no effect.

Note that raw status for read and write is set or cleared based on FIFO fullness as controlled by **EPIFIFOLVL**.

Raw status for error is held until the error is cleared by writing to the **EPIIC** register.

EPI Raw Interrupt Status (EPIRIS)

Base 0x400D.0000
 Offset 0x214
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | WRRIS | RDRIS | ERRIS |
| 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 |
|-----------|----------|------|------------|---|
| 31:3 | 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 | WRRIS | RO | 0 | Write Raw Interrupt Status Value Description 1 The number of available entries in the WFIFO is within the range specified by the trigger level (the <code>WRFIFO</code> field in the EPIFIFOLVL register). 0 The number of available entries in the WFIFO is above the range specified by the trigger level. This bit is cleared when the level in the WFIFO is above the trigger point programmed by the <code>WRFIFO</code> field. |
| 1 | RDRIS | RO | 0 | Read Raw Interrupt Status Value Description 1 The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the <code>RDFIFO</code> field in the EPIFIFOLVL register). 0 The number of valid entries in the NBRFIFO is below the range specified by the trigger level. This bit is cleared when the level in the NBRFIFO is below the trigger point programmed by the <code>RDFIFO</code> field. |

| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|--|------|-------|---|---|--|---|----------------------------|
| 0 | ERRRIS | RO | 0 | <p>Error Raw Interrupt Status</p> <p>The error interrupt occurs in the following situations:</p> <ul style="list-style-type: none"> ■ WFIFO Full. For a full WFIFO to generate an error interrupt, the <code>WFERR</code> bit in the EPIFIFOLVL register must be set. ■ Read Stalled. For a stalled read to generate an error interrupt, the <code>RSERR</code> bit in the EPIFIFOLVL register must be set. ■ Timeout. If the <code>MAXWAIT</code> field in the EPIGPCFG register is configured to a value other than 0, a timeout error occurs when <code>iRDY</code> or <code>XFIFO</code> not-ready signals hold a transaction for more than the count in the <code>MAXWAIT</code> field. <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>A WFIFO Full, a Read Stalled, or a Timeout error has occurred.</td> </tr> <tr> <td>0</td> <td>An error has not occurred.</td> </tr> </table> <p>To determine which error occurred, read the status of the EPI Error Interrupt Status and Clear (EPIEISC) register. This bit is cleared by writing a 1 to the bit in the EPIEISC register that caused the interrupt.</p> | 1 | A WFIFO Full, a Read Stalled, or a Timeout error has occurred. | 0 | An error has not occurred. |
| 1 | A WFIFO Full, a Read Stalled, or a Timeout error has occurred. | | | | | | | |
| 0 | An error has not occurred. | | | | | | | |

Register 32: EPI Masked Interrupt Status (EPIMIS), offset 0x218

This register is the masked interrupt status register. On read, it gives the current state of each interrupt source (read, write, and error) after being masked via the **EPIIM** register. A write has no effect.

The values returned are the ANDing of the **EPIIM** and **EPIRIS** registers. If a bit is set in this register, the interrupt is sent to the interrupt controller.

EPI Masked Interrupt Status (EPIMIS)

Base 0x400D.0000
 Offset 0x218
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|--------|
| | 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 | | | | | | | | | | | | | WRMIS | RDMIS | ERRMIS |
| 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 |
|-----------|----------|------|------------|---|
| 31:3 | 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 | WRMIS | RO | 0 | Write Masked Interrupt Status Value Description 1 The number of available entries in the WFIFO is within the range specified by the trigger level (the WRFIFO field in the EPIFIFOLVL register) and the WRIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller. 0 The number of available entries in the WFIFO is above the range specified by the trigger level or the interrupt is masked. |
| 1 | RDMIS | RO | 0 | Read Masked Interrupt Status Value Description 1 The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the RDFIFO field in the EPIFIFOLVL register) and the RDIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller. 0 The number of valid entries in the NBRFIFO is below the range specified by the trigger level or the interrupt is masked. |
| 0 | ERRMIS | RO | 0 | Error Masked Interrupt Status Value Description 1 A WFIFO Full, a Read Stalled, or a Timeout error has occurred and the ERIM bit in the EPIIM register is set, triggering an interrupt to the interrupt controller. 0 An error has not occurred or the interrupt is masked. |

Register 33: EPI Error Interrupt Status and Clear (EPIEISC), offset 0x21C

This register is used to clear a pending error interrupt. If any of these bits are set, the `ERRRIS` bit in the `EPIRIS` register is set, and an EPI controller error is sent to the interrupt controller if the `ERIM` bit in the `EPIIM` register is set. Clearing any defined bit has no effect; setting a bit clears the error source and the raw error returns to 0. Note that writing to this register and reading back immediately (pipelined by the processor) returns the old register contents. One cycle is needed between write and read.

EPI Error Interrupt Status and Clear (EPIEISC)

Base 0x400D.0000

Offset 0x21C

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 | | | | | | | | | | | | | WTFULL | RSTALL | TOUT | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | WTFULL | R/W1C | 0 | Write FIFO Full Error Value Description 1 The <code>WFERR</code> bit is enabled and a write is stalled due to the <code>WFIFO</code> being full. 0 The <code>WFERR</code> bit is not enabled or no writes are stalled. Writing a 1 to this bit clears it and the <code>WFERR</code> bit in the <code>EPIFIFOLVL</code> register. |
| 1 | RSTALL | R/W1C | 0 | Read Stalled Error Value Description 1 The <code>RSERR</code> bit is enabled and a pending read is stalled due to writes in the <code>WFIFO</code> . 0 The <code>RSERR</code> bit is not enabled or no pending reads are stalled. Writing a 1 to this bit clears it and the <code>RSERR</code> bit in the <code>EPIFIFOLVL</code> register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|-------|-------|--|
| 0 | TOUT | R/W1C | 0 | <p>Timeout Error</p> <p>This bit is the timeout error source. The timeout error occurs when the iRDY or XFIFO not-ready signals hold a transaction for more than the count in the <code>MAXWAIT</code> field (when not 0).</p> <p>Value Description</p> <p>1 A timeout error has occurred.</p> <p>0 No timeout error has occurred.</p> <p>Writing a 1 to bit this clears it.</p> |

12 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 (Timer 0, Timer 1, Timer 2, and Timer 3). Each GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger μ DMA transfers.

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

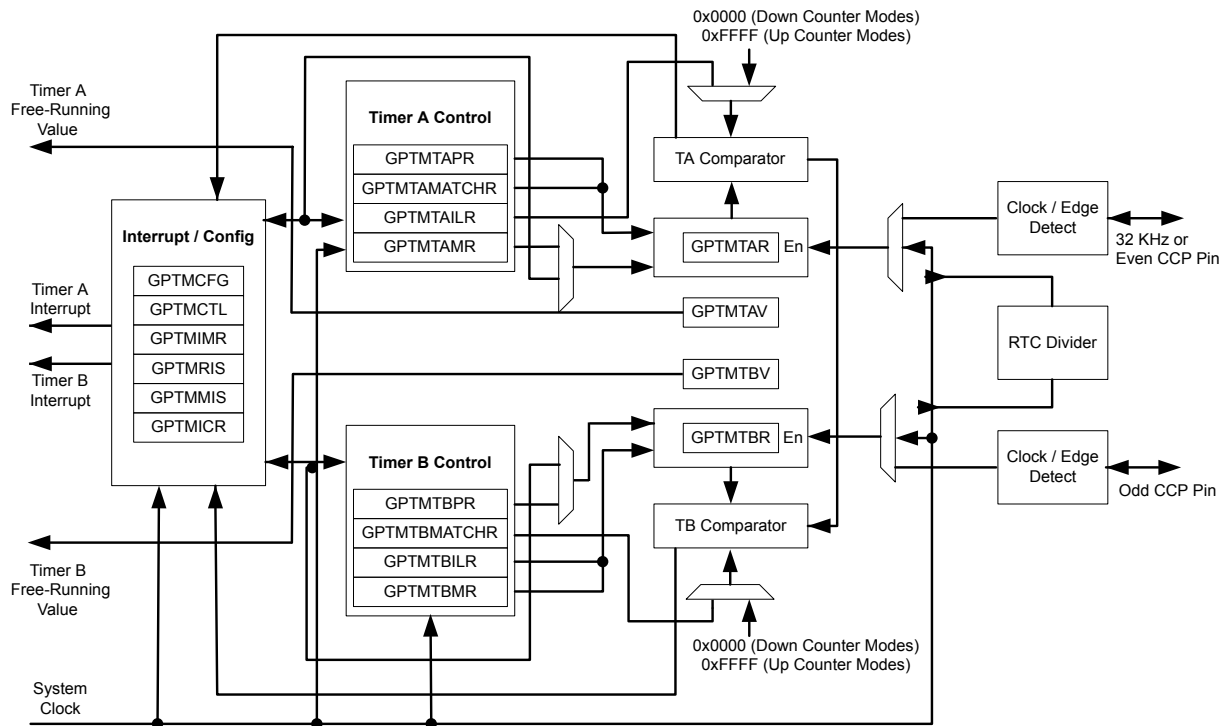
The GPT Module is one timing resource available on the Stellaris® microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 74).

The General-Purpose Timer Module (GPTM) contains four GPTM blocks with the following functional options:

- Count up or down
- 16- or 32-bit programmable one-shot timer
- 16- or 32-bit programmable periodic timer
- 16-bit general-purpose timer with an 8-bit prescaler
- 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
- Eight Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- ADC event trigger
- User-enabled stalling when the controller asserts CPU Halt flag during debug (excluding RTC mode)
- 16-bit input-edge count- or time-capture modes
- 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Dedicated channel for each timer
 - Burst request generated on timer interrupt

12.1 Block Diagram

Figure 12-1. GPTM Module Block Diagram



Note: In Figure 12-1 on page 434, the specific Capture Compare PWM (CCP) pins available depend on the Stellaris® device. See Table 12-1 on page 434 for the available CCP pins and their timer assignments

Table 12-1. Available CCP Pins

| Timer | 16-Bit Up/Down Counter | Even CCP Pin | Odd CCP Pin |
|---------|------------------------|--------------|-------------|
| Timer 0 | Timer A | CCP0 | - |
| | Timer B | - | CCP1 |
| Timer 1 | Timer A | CCP2 | - |
| | Timer B | - | CCP3 |
| Timer 2 | Timer A | CCP4 | - |
| | Timer B | - | CCP5 |
| Timer 3 | Timer A | CCP6 | - |
| | Timer B | - | CCP7 |

12.2 Signal Description

Table 12-2 on page 435 lists the external signals of the GP Timer module and describes the function of each. The GP Timer signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these GP Timer signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the GP Timer function.

The number in parentheses is the encoding that must be programmed into the PMC_n field in the **GPIO Port Control (GPIOCTL)** register (page 358) to assign the GP Timer signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 318.

Table 12-2. Signals for General-Purpose Timers

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|------------------------|
| CCP0 | 13 | PD3 (4) | I/O | TTL | Capture/Compare/PWM 0. |
| | 22 | PC7 (4) | | | |
| | 23 | PC6 (6) | | | |
| | 39 | PJ2 (9) | | | |
| | 42 | PF4 (1) | | | |
| | 66 | PB0 (1) | | | |
| | 72 | PB2 (5) | | | |
| | 91 | PB5 (4) | | | |
| 97 | PD4 (1) | | | | |
| CCP1 | 24 | PC5 (1) | I/O | TTL | Capture/Compare/PWM 1. |
| | 25 | PC4 (9) | | | |
| | 34 | PA6 (2) | | | |
| | 67 | PB1 (4) | | | |
| | 90 | PB6 (1) | | | |
| | 96 | PE3 (1) | | | |
| 100 | PD7 (3) | | | | |
| CCP2 | 6 | PE4 (6) | I/O | TTL | Capture/Compare/PWM 2. |
| | 11 | PD1 (10) | | | |
| | 25 | PC4 (5) | | | |
| | 41 | PF5 (1) | | | |
| | 67 | PB1 (1) | | | |
| | 75 | PE1 (4) | | | |
| | 91 | PB5 (6) | | | |
| | 95 | PE2 (5) | | | |
| 98 | PD5 (1) | | | | |
| CCP3 | 6 | PE4 (1) | I/O | TTL | Capture/Compare/PWM 3. |
| | 23 | PC6 (1) | | | |
| | 24 | PC5 (5) | | | |
| | 35 | PA7 (7) | | | |
| | 61 | PF1 (10) | | | |
| | 72 | PB2 (4) | | | |
| | 74 | PE0 (3) | | | |
| 97 | PD4 (2) | | | | |
| CCP4 | 22 | PC7 (1) | I/O | TTL | Capture/Compare/PWM 4. |
| | 25 | PC4 (6) | | | |
| | 35 | PA7 (2) | | | |
| | 95 | PE2 (1) | | | |
| | 98 | PD5 (2) | | | |
| CCP5 | 5 | PE5 (1) | I/O | TTL | Capture/Compare/PWM 5. |
| | 12 | PD2 (4) | | | |
| | 25 | PC4 (1) | | | |
| | 36 | PG7 (8) | | | |
| | 90 | PB6 (6) | | | |
| | 91 | PB5 (2) | | | |
| CCP6 | 10 | PD0 (6) | I/O | TTL | Capture/Compare/PWM 6. |
| | 12 | PD2 (2) | | | |
| | 75 | PE1 (5) | | | |
| | 86 | PH0 (1) | | | |
| | 91 | PB5 (3) | | | |

Table 12-2. Signals for General-Purpose Timers (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|------------------------|
| CCP7 | 11 | PD1 (6) | I/O | TTL | Capture/Compare/PWM 7. |
| | 13 | PD3 (2) | | | |
| | 85 | PH1 (1) | | | |
| | 90 | PB6 (2) | | | |
| | 96 | PE3 (5) | | | |

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

12.3 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as Timer A and Timer B), two 16-bit match registers, 2 16-bit shadow 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 448), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 449), and the **GPTM Timer B Mode (GPTMTBMR)** register (see page 451). 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.

12.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 466) and the **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 467) and shadow registers: the **GPTM Timer A Value (GPTMTAV)** register (see page 474) and the **GPTM Timer B Value (GPTMTBV)** register (see page 475). The prescale counters are initialized to 0x00: the **GPTM Timer A Prescale (GPTMTAPR)** register (see page 470) and the **GPTM Timer B Prescale (GPTMTBPR)** register (see page 471).

12.3.2 32-Bit Timer Operating Modes

This section describes the three GPTM 32-bit timer modes (One-Shot, Periodic, and RTC) and their configurations.

The GPTM is placed into 32-bit mode by writing a 0x0 (One-Shot/Periodic 32-bit timer mode) or a 0x1 (RTC mode) to the **GPTMCFG** bit field in the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain GPTM registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 466
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 467
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 472
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 473
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 474
- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 475

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is:

```
GPTMTBILR[15:0]:GPTMTAILR[15:0]
```

Likewise, a 32-bit read access to **GPTMTAR** returns the value:

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

A 32-bit read access to **GPTMTAV** returns the value:

```
GPTMTBV[15:0]:GPTMTAV[15:0]
```

12.3.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the Timer A and Timer B registers are configured as a 32-bit up or down counter. The selection of one-shot or periodic mode is determined by the value written to the **TAMR** field of the **GPTM Timer A Mode (GPTMTAMR)** register (see page 449); there is no need to write to the **GPTM Timer B Mode (GPTMTBMR)** register.

When software sets the **TAEN** bit in the **GPTM Control (GPTMCTL)** register (see page 453), the timer begins counting up or down from its preloaded value. Alternatively, if the **TnWOT** bit is set in the **GPTMTnMR** register, once the **TnEN** bit is set, the timer waits for a trigger to begin counting (see “Wait-for-Trigger Mode” on page 442).

Once the time-out event (0x0000.0000 when counting down, 0xFFFF.FFFF when counting up) 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. If the **TnSNAPS** bit in the **GPTMTnMR** register is set, the actual free-running value of the timer at the time-out event is loaded into the **GPTMTAR** register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the time-out event. The GPTM sets the **TATORIS** bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 458), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 464). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 456), the GPTM also sets the **TATOMIS** bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 461). By setting the **TAMIE** bit in the **GPTMTAMR** register, an interrupt can also be generated when the Timer A value equals the value loaded into the **GPTM Timer A Match (GPTMTAMATCH)** register. This interrupt has the same status, masking, and clearing functions as the time-out interrupt. The ADC trigger is enabled by setting the **TAOTE** bit in **GPTMCTL**. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel. See “Channel Configuration” on page 262.

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 until the bit is cleared.

12.3.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the Timer A and Timer B registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time after reset, the counter is loaded with a value of 0x0000.0001. All subsequent load values must be written to the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 466).

The input clock on the CCP0, CCP2, or CCP4 signal 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 in the 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, the GPTM asserts the RTCRIS bit in GPTMRIS and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). If the RTC interrupt is enabled in GPTIMR, the GPTM also sets the RTCMIS bit in GPTMISR and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in GPTMICR.

In addition to generating interrupts, a μ DMA trigger can be generated. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel. See “Channel Configuration” on page 262.

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.

12.3.3 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 448). This section describes each of the GPTM 16-bit modes of operation. Timer A and Timer B have identical modes, so a single description is given using an n to reference both.

12.3.3.1 16-Bit One-Shot/Periodic Timer Mode

In 16-bit one-shot and periodic timer modes, the timer is configured as a 16-bit up or 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 Timer n Prescale (GPTMTnPR) register.

When software sets the TnEN bit in the GPTMCTL register, the timer begins counting up or down from its preloaded value. Alternatively, if the TnWOT bit is set in the GPTMTnMR register, once the TnEN bit is set, the timer waits for a trigger to begin counting (see “Wait-for-Trigger Mode” on page 442).

Once the time-out event (0x0000 when counting down, 0xFFFF when counting up) 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. If the TnSNAPS bit in the GPTMTnMR register is set, the actual free-running value of the timer at the time-out event is loaded into the GPTMTAR register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the timer generates interrupts and triggers when it reaches the time-out event. 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 GPTIMR, the GPTM also sets the TnTOMIS bit in GPTMISR and generates a controller interrupt. By setting the TnMIE bit in the GPTMTnMR register, an interrupt can also be generated when the timer value equals the value loaded into the GPTM Timer n Match (GPTMTnMATCH) register. This interrupt has the same status, masking, and clearing functions as the time-out interrupt. The ADC trigger is enabled by setting the TnOTE bit in the GPTMCTL register. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel. See “Channel Configuration” on page 262.

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 until the bit is cleared. The following example shows a variety of configurations for a 16-bit free-running timer while using the prescaler. All values assume an 80-MHz clock with $Tc=12.5$ ns (clock period).

Table 12-3. 16-Bit Timer With Prescaler Configurations

| Prescale | #Clock (Tc) ^a | Max Time | Units |
|----------|------------------------------|----------|-------|
| 00000000 | 1 | 0.8192 | mS |
| 00000001 | 2 | 1.6385 | mS |
| 00000010 | 3 | 2.4576 | mS |
| ----- | -- | -- | -- |
| 11111100 | 254 | 208.0768 | mS |
| 11111110 | 255 | 208.896 | mS |
| 11111111 | 256 | 209.7152 | mS |

a. Tc is the clock period.

12.3.3.2 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.

In Edge-Count mode, 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. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge-Count mode, the $TnCMR$ bit of the **GPTMTnMR** register must be cleared. The type of edge that the timer counts is determined by the $TnEVENT$ fields of the **GPTMCTL** register. During initialization, the **GPTM Timer n 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. The optional prescaler is loaded into the **GPTM Timer n Prescale (GPTMTnPR)** register.

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).

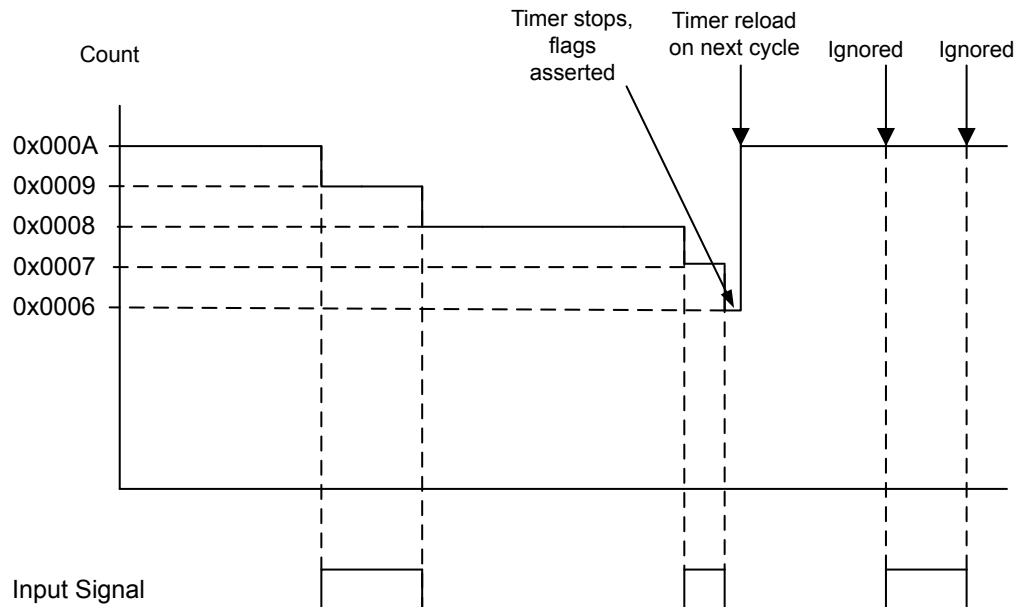
In addition to generating interrupts, a μ DMA trigger can be generated. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel. See “Channel Configuration” on page 262.

The counter is then reloaded using the value in **GPTMTnILR**, and stopped because the GPTM automatically clears the $TnEN$ bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until $TnEN$ is re-enabled by software. The **GPTMTnV** contains the free-running timer value and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

Figure 12-2 on page 440 shows how Input Edge-Count mode works. In this case, the timer start value is set to **GPTMnILR** = 0x000A and the match value is set to **GPTMnMATCHR** = 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 **GPTMnMR** register.

Figure 12-2. 16-Bit Input Edge-Count Mode Example



12.3.3.3 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.

In Edge-Time mode, the timer is configured as a 16-bit free-running down-counter with an optional 8-bit prescaler that effectively extends the counting range of the timer to 24 bits. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR** register (or 0xFFFF at reset). This mode allows for event capture of either rising or falling edges, but not 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 **GPTMCnTL** register. The optional prescaler is loaded into the **GPTM Timer n Prescale (GPTMTnPR)** 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 microcontroller. The GPTM then asserts the **CnERIS** bit (and the **CnEMIS** bit, if the interrupt is not masked). The **GPTMTnV** is the free-running value of the timer and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

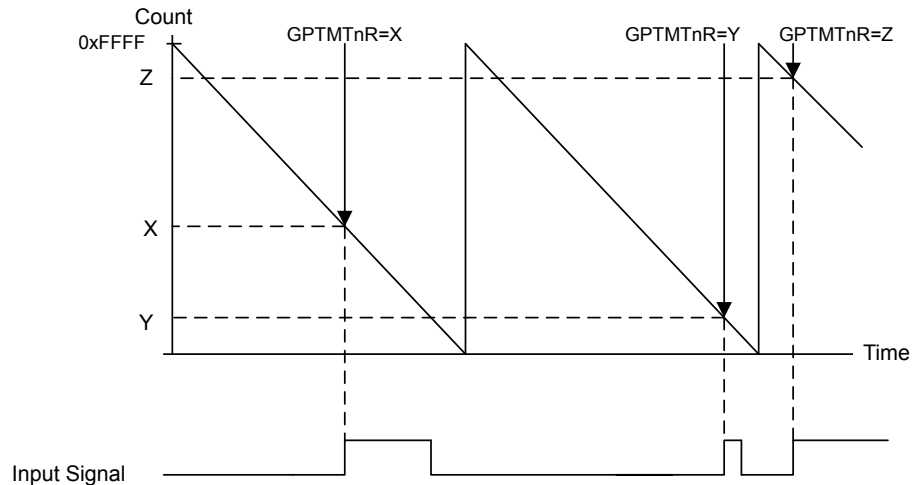
In addition to generating interrupts, a μ DMA trigger can be generated. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel. See “Channel Configuration” on page 262.

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 **GPTMnILR** register.

Figure 12-3 on page 441 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 12-3. 16-Bit Input Edge-Time Mode Example



12.3.3.4 16-Bit PWM Mode

Note: The prescaler is not available in 16-Bit PWM mode.

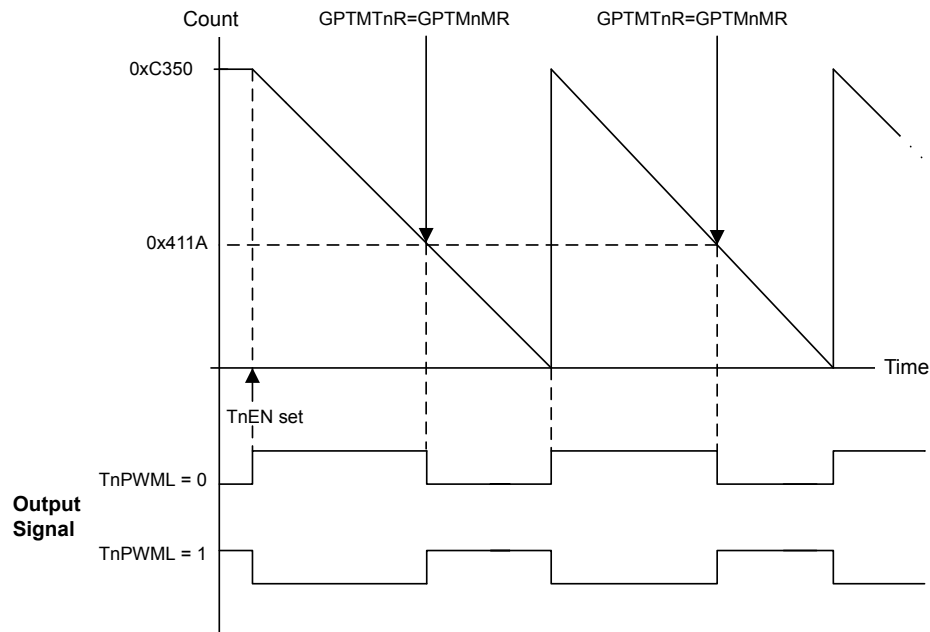
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the T_nAMS bit to 0x1, the T_nCMR bit to 0x0, and the T_nMR field to 0x2.

When software writes the T_nEN 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 T_nEN 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 Timer n Match Register (GPTMnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the T_nPWML bit in the **GPTMCTL** register.

Figure 12-4 on page 442 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and $T_nPWML = 0$ (duty cycle would be 33% for the $T_nPWML = 1$ configuration). For this example, the start value is **GPTMnIRL**=0xC350 and the match value is **GPTMnMR**=0x411A.

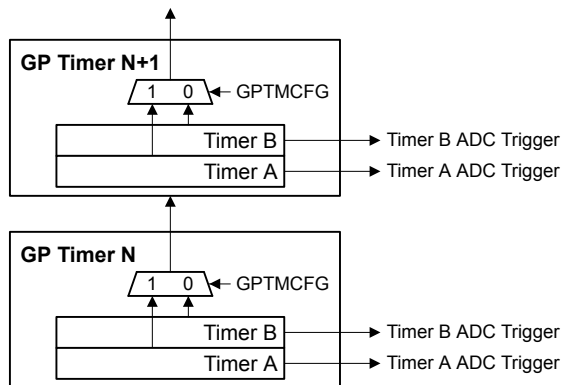
Figure 12-4. 16-Bit PWM Mode Example



12.3.3.5 Wait-for-Trigger Mode

The Wait-for-Trigger mode allows daisy chaining of the timer modules such that once configured, a single timer can initiate multiple timing events using the Timer triggers. Wait-for-Trigger mode is enabled by setting the $TnWOT$ bit in the **GPTMTnMR** register. When the $TnWOT$ bit is set, Timer N+1 does not begin counting until the timer in the previous position in the daisy chain (Timer N) reaches its time-out event. The daisy chain is configured such that GPTM1 always follows GPTM0, GPTM2 follows GPTM1, and so on. If Timer A is in 32-bit mode (controlled by the $GPTMCFG$ bit in the **GPTMCFG** register), it triggers Timer A in the next module. If Timer A is in 16-bit mode, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Care must be taken that the $TAWOT$ bit is never set in GPTM0. Figure 12-5 on page 442 shows how the $GPTMCFG$ bit affects the daisy chain. This function is valid for both one-shot and periodic modes.

Figure 12-5. Timer Daisy Chain



12.3.4 DMA Operation

The timers each have a dedicated μ DMA channel and can provide a request signal to the μ DMA controller. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the μ DMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic timeout at 10 ms. Configure the μ DMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the μ DMA controller transfers 8 items, until all 256 items have been transferred.

No other special steps are needed to enable Timers for μ DMA operation. Refer to “Micro Direct Memory Access (μ DMA)” on page 258 for more details about programming the μ DMA controller.

12.4 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 (see page 175). If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 184). To find out which GPIO port to enable, refer to Table 24-4 on page 1028. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the CCP signals to the appropriate pins (see page 358 and Table 24-5 on page 1035).

This section shows module initialization and configuration examples for each of the supported timer modes.

12.4.1 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 `0x0000.0000`.
3. Configure the `TAMR` field in the **GPTM Timer A Mode Register (GPTMTAMR)**:
 - a. Write a value of `0x1` for One-Shot mode.
 - b. Write a value of `0x2` for Periodic mode.
4. Optionally configure the `TASNAPS`, `TAWOT`, `TAMTE`, and `TACDIR` bits in the **GPTMTAMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. Load the start value into the **GPTM Timer A Interval Load Register (GPTMTAILR)**.
6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the `TAEN` bit in the **GPTMCTL** register to enable the timer and start counting.
8. Poll the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

If the `TAMIE` bit in the **GPTMTAMR** register is set, the `RTCRES` bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

12.4.2 32-Bit Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on the `CCP0`, `CCP2`, or `CCP4` signal. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the `TAEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of `0x0000.0001`.
3. Write the match value to the **GPTM Timer A Match Register (GPTMTAMATCHR)**.
4. Set/clear the `RTCEN` bit in the **GPTM Control Register (GPTMCTL)** as needed.
5. If interrupts are required, set the `RTCIM` bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the `TAEN` bit in the **GPTMCTL** register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTAMATCHR** register, the counter is re-loaded with `0x0000.0000` and begins counting. If an interrupt is enabled, it does not have to be cleared.

12.4.3 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 `0x0000.0004`.
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. Optionally configure the `TnSNAPS`, `TnWOT`, `TnMTE` and `TnCDIR` bits in the **GPTMTnMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the start value into the **GPTM Timer Interval Load Register (GPTMTnILR)**.
7. If interrupts are required, set the appropriate bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
8. Set the `TnEN` bit in the **GPTM Control Register (GPTMCTL)** to enable the timer and start counting.

9. Poll the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

If the $TnMIE$ bit in the **GPTMTnMR** register is set, the $RTCRES$ bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

12.4.4 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 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the $TnCMR$ field to 0x0 and the $TnMR$ field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the $TnEVENT$ field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. Load the event count into the **GPTM Timer n Match (GPTMTnMATCHR)** register.
8. If interrupts are required, set the $CnMIM$ bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
9. Set the $TnEN$ bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
10. Poll the $CnMRIS$ bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the $CnMCINT$ bit of the **GPTM Interrupt Clear (GPTMICR)** register.

In Input Edge-Count Mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the $TnEN$ bit is cleared and repeat step 4 on page 445 through step 9 on page 445.

12.4.5 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 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the $TnCMR$ field to 0x1 and the $TnMR$ field to 0x3.
4. Configure the type of event that the timer captures by writing the $TnEVENT$ field of the **GPTM Control (GPTMCTL)** register.

5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. If interrupts are required, set the `CnEIM` bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the `TnEN` bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
9. Poll the `CnERIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `CnECINT` bit of the **GPTM Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM Timer n (GPTMTnR)** register.

In Input Edge Timing mode, the timer continues running after an edge event has been detected, but the timer interval can be changed at any time by writing the **GPTMTnILR** register. The change takes effect at the next cycle after the write.

12.4.6 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 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the `TnAMS` bit to 0x1, the `TnCMR` bit to 0x0, and the `TnMR` field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the match value.
7. Set the `TnEN` bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

In PWM Timing mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

12.5 Register Map

Table 12-4 on page 447 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 GP Timer module clock must be enabled before the registers can be programmed (see page 175).

Table 12-4. Timers Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|--------------|------|-------------|------------------------------|----------|
| 0x000 | GPTMCFG | R/W | 0x0000.0000 | GPTM Configuration | 448 |
| 0x004 | GPTMTAMR | R/W | 0x0000.0000 | GPTM Timer A Mode | 449 |
| 0x008 | GPTMTBMR | R/W | 0x0000.0000 | GPTM Timer B Mode | 451 |
| 0x00C | GPTMCTL | R/W | 0x0000.0000 | GPTM Control | 453 |
| 0x018 | GPTMIMR | R/W | 0x0000.0000 | GPTM Interrupt Mask | 456 |
| 0x01C | GPTMRIS | RO | 0x0000.0000 | GPTM Raw Interrupt Status | 458 |
| 0x020 | GPTMMIS | RO | 0x0000.0000 | GPTM Masked Interrupt Status | 461 |
| 0x024 | GPTMICR | W1C | 0x0000.0000 | GPTM Interrupt Clear | 464 |
| 0x028 | GPTMTAILR | R/W | 0xFFFF.FFFF | GPTM Timer A Interval Load | 466 |
| 0x02C | GPTMTBILR | R/W | 0x0000.FFFF | GPTM Timer B Interval Load | 467 |
| 0x030 | GPTMTAMATCHR | R/W | 0xFFFF.FFFF | GPTM Timer A Match | 468 |
| 0x034 | GPTMTBMATCHR | R/W | 0x0000.FFFF | GPTM Timer B Match | 469 |
| 0x038 | GPTMTAPR | R/W | 0x0000.0000 | GPTM Timer A Prescale | 470 |
| 0x03C | GPTMTBPR | R/W | 0x0000.0000 | GPTM Timer B Prescale | 471 |
| 0x048 | GPTMTAR | RO | 0xFFFF.FFFF | GPTM Timer A | 472 |
| 0x04C | GPTMTBR | RO | 0x0000.FFFF | GPTM Timer B | 473 |
| 0x050 | GPTMTAV | RO | 0xFFFF.FFFF | GPTM Timer A Value | 474 |
| 0x054 | GPTMTBV | RO | 0x0000.FFFF | GPTM Timer B Value | 475 |

12.6 Register Descriptions

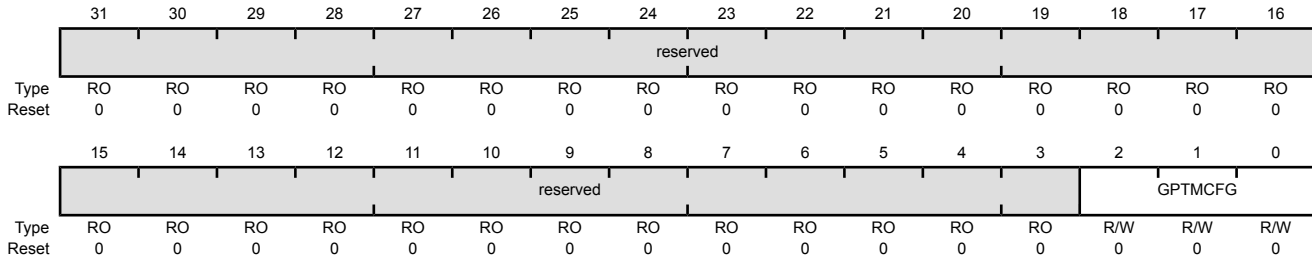
The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

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 | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2:0 | GPTMCFG | R/W | 0x0 | GPTM Configuration |

The GPTMCFG values are defined as follows:

| Value | Description |
|---------|---|
| 0x0 | 32-bit timer configuration. |
| 0x1 | 32-bit real-time clock (RTC) counter configuration. |
| 0x2-0x3 | Reserved |
| 0x4-0x7 | 16-bit timer configuration, function is controlled by bits 1:0 of GPTMTAMR and GPTMTBMR . |

Register 2: GPTM Timer A Mode (GPTMTAMR), offset 0x004

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the **TAAMS** bit, clear the **TACMR** bit, and configure the **TAMR** field to 0x2.

In 16-bit timer configuration, **TAMR** controls the 16-bit timer modes for Timer A. In 32-bit timer configuration, this register controls the mode, and the contents of **GPTMTBMR** are ignored.

GPTM Timer A 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|---------|-------|-------|--------|-------|-------|------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | TASNAPS | TAWOT | TAMIE | TACDIR | TAAMS | TACMR | TAMR | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7 | TASNAPS | R/W | 0 | GPTM Timer A Snap-Shot Mode Value Description 0 Snap-shot mode is disabled. 1 If Timer A is configured in the periodic mode, the actual free-running value of Timer A is loaded at the time-out event into the GPTM Timer A (GPTMTAR) register. |
| 6 | TAWOT | R/W | 0 | GPTM Timer A Wait-on-Trigger Value Description 0 Timer A begins counting as soon as it is enabled. 1 If Timer A is enabled (TAEN is set in the GPTMCTL register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see Figure 12-5 on page 442. This function is valid for both one-shot and periodic modes. |

This bit must be clear for GP Timer Module 0, Timer A.

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 5 | TAMIE | R/W | 0 | <p>GPTM Timer A Match Interrupt Enable</p> <p>Value Description</p> <p>0 The match interrupt is disabled.</p> <p>1 An interrupt is generated when the match value in the GPTMTAMATCHR register is reached in the one-shot and periodic modes.</p> |
| 4 | TACDIR | R/W | 0 | <p>GPTM Timer A Count Direction</p> <p>Value Description</p> <p>0 The timer counts down.</p> <p>1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0000.</p> |
| 3 | TAAMS | R/W | 0 | <p>GPTM Timer A Alternate Mode Select</p> <p>The TAAMS values are defined as follows:</p> <p>Value Description</p> <p>0 Capture mode is enabled.</p> <p>1 PWM mode is enabled.</p> <p>Note: To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x2.</p> |
| 2 | TACMR | R/W | 0 | <p>GPTM Timer A Capture Mode</p> <p>The TACMR values are defined as follows:</p> <p>Value Description</p> <p>0 Edge-Count mode</p> <p>1 Edge-Time mode</p> |
| 1:0 | TAMR | R/W | 0x0 | <p>GPTM Timer A Mode</p> <p>The TAMR values are defined as follows:</p> <p>Value Description</p> <p>0x0 Reserved</p> <p>0x1 One-Shot Timer mode</p> <p>0x2 Periodic Timer mode</p> <p>0x3 Capture mode</p> <p>The Timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register (16-or 32-bit).</p> |

Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the **TBAMS** bit, clear the **TBCMR** bit, and configure the **TBMR** field to 0x2.

In 16-bit timer configuration, these bits control the 16-bit timer modes for Timer B. In 32-bit timer configuration, this register's contents are ignored, and **GPTMTAMR** is used.

GPTM Timer B 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|---------|-------|-------|--------|-------|-------|------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | TBSNAPS | TBWOT | TBMIE | TBCDIR | TBAMS | TBCMR | TBMR | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|--|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7 | TBSNAPS | R/W | 0 | GPTM Timer B Snap-Shot Mode Value Description 0 Snap-shot mode is disabled. 1 If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the GPTM Timer B (GPTMTBR) register. |
| 6 | TBWOT | R/W | 0 | GPTM Timer B Wait-on-Trigger Value Description 0 Timer B begins counting as soon as it is enabled. 1 If Timer B is enabled (TBEN is set in the GPTMCTL register), Timer B does not begin counting until it receives an it receives a trigger from the timer in the previous position in the daisy chain. See Figure 12-5 on page 442. This function is valid for both one-shot and periodic modes. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 5 | TBMIE | R/W | 0 | <p>GPTM Timer B Match Interrupt Enable</p> <p>Value Description</p> <p>0 The match interrupt is disabled.</p> <p>1 An interrupt is generated when the match value in the GPTMTBMATCHR register is reached in the one-shot and periodic modes.</p> |
| 4 | TBCDIR | R/W | 0 | <p>GPTM Timer B Count Direction</p> <p>Value Description</p> <p>0 The timer counts down.</p> <p>1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0000.</p> |
| 3 | TBAMS | R/W | 0 | <p>GPTM Timer B Alternate Mode Select</p> <p>The TBAMS values are defined as follows:</p> <p>Value Description</p> <p>0 Capture mode is enabled.</p> <p>1 PWM mode is enabled.</p> <p>Note: To enable PWM mode, you must also clear the TBCMR bit and set the TBMR field to 0x2.</p> |
| 2 | TBCMR | R/W | 0 | <p>GPTM Timer B Capture Mode</p> <p>The TBCMR values are defined as follows:</p> <p>Value Description</p> <p>0 Edge-Count mode</p> <p>1 Edge-Time mode</p> |
| 1:0 | TBMR | R/W | 0x0 | <p>GPTM Timer B Mode</p> <p>The TBMR values are defined as follows:</p> <p>Value Description</p> <p>0x0 Reserved</p> <p>0x1 One-Shot Timer mode</p> <p>0x2 Periodic Timer mode</p> <p>0x3 Capture mode</p> <p>The timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register.</p> |

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

| | | | | | | | | | | | | | | | | |
|-------|----------|--------|-------|----------|---------|---------|------|----------|--------|-------|-------|---------|---------|------|-----|-----|
| | 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 | TBPWML | TBOTE | reserved | TBEVENT | TBSTALL | TBEN | reserved | TAPWML | TAOTE | RTCEN | TAEVENT | TASTALL | TAEN | | |
| Type | RO | R/W | R/W | RO | R/W | R/W | R/W | R/W | 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 | | | | | | |
|-----------|---|------|--------|--|-------|-------------|---|---|---|--|
| 31:15 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | |
| 14 | TBPWML | R/W | 0 | GPTM Timer B PWM Output Level The TBPWML values are defined as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Output is unaffected.</td> </tr> <tr> <td>1</td> <td>Output is inverted.</td> </tr> </tbody> </table> | Value | Description | 0 | Output is unaffected. | 1 | Output is inverted. |
| Value | Description | | | | | | | | | |
| 0 | Output is unaffected. | | | | | | | | | |
| 1 | Output is inverted. | | | | | | | | | |
| 13 | TBOTE | R/W | 0 | GPTM Timer B Output Trigger Enable The TBOTE values are defined as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output Timer B ADC trigger is disabled.</td> </tr> <tr> <td>1</td> <td>The output Timer B ADC trigger is enabled.</td> </tr> </tbody> </table> <p>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 530).</p> | Value | Description | 0 | The output Timer B ADC trigger is disabled. | 1 | The output Timer B ADC trigger is enabled. |
| Value | Description | | | | | | | | | |
| 0 | The output Timer B ADC trigger is disabled. | | | | | | | | | |
| 1 | The output Timer B ADC trigger is enabled. | | | | | | | | | |
| 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 | <p>GPTM Timer B Event Mode</p> <p>The TBEVENT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table> | Value | Description | 0x0 | Positive edge | 0x1 | Negative edge | 0x2 | Reserved | 0x3 | Both edges |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Positive edge | | | | | | | | | | | | | |
| 0x1 | Negative edge | | | | | | | | | | | | | |
| 0x2 | Reserved | | | | | | | | | | | | | |
| 0x3 | Both edges | | | | | | | | | | | | | |
| 9 | TBSTALL | R/W | 0 | <p>GPTM Timer B Stall Enable</p> <p>The TBSTALL values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer B stalling is disabled.</td> </tr> <tr> <td>1</td> <td>Timer B stalling is enabled.</td> </tr> </tbody> </table> | Value | Description | 0 | Timer B stalling is disabled. | 1 | Timer B stalling is enabled. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Timer B stalling is disabled. | | | | | | | | | | | | | |
| 1 | Timer B stalling is enabled. | | | | | | | | | | | | | |
| 8 | TBEN | R/W | 0 | <p>GPTM Timer B Enable</p> <p>The TBEN values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer B is disabled.</td> </tr> <tr> <td>1</td> <td>Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.</td> </tr> </tbody> </table> | Value | Description | 0 | Timer B is disabled. | 1 | Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Timer B is disabled. | | | | | | | | | | | | | |
| 1 | Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register. | | | | | | | | | | | | | |
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 6 | TAPWML | R/W | 0 | <p>GPTM Timer A PWM Output Level</p> <p>The TAPWML values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Output is unaffected.</td> </tr> <tr> <td>1</td> <td>Output is inverted.</td> </tr> </tbody> </table> | Value | Description | 0 | Output is unaffected. | 1 | Output is inverted. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Output is unaffected. | | | | | | | | | | | | | |
| 1 | Output is inverted. | | | | | | | | | | | | | |
| 5 | TAOTE | R/W | 0 | <p>GPTM Timer A Output Trigger Enable</p> <p>The TAOTE values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output Timer A ADC trigger is disabled.</td> </tr> <tr> <td>1</td> <td>The output Timer A ADC trigger is enabled.</td> </tr> </tbody> </table> <p>In addition, the ADC must be enabled and the timer selected as a trigger source with the EM_n bit in the ADCEMUX register (see page 530).</p> | Value | Description | 0 | The output Timer A ADC trigger is disabled. | 1 | The output Timer A ADC trigger is enabled. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | The output Timer A ADC trigger is disabled. | | | | | | | | | | | | | |
| 1 | The output Timer A ADC trigger is enabled. | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|--|------|-------|---|-------|-------------|-----|-------------------------------|-----|--|-----|----------|-----|------------|
| 4 | RTCEN | R/W | 0 | <p>GPTM RTC Enable</p> <p>The <code>RTCEN</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>RTC counting is disabled.</td> </tr> <tr> <td>1</td> <td>RTC counting is enabled.</td> </tr> </tbody> </table> | Value | Description | 0 | RTC counting is disabled. | 1 | RTC counting is enabled. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | RTC counting is disabled. | | | | | | | | | | | | | |
| 1 | RTC counting is enabled. | | | | | | | | | | | | | |
| 3:2 | TAEVENT | R/W | 0x0 | <p>GPTM Timer A Event Mode</p> <p>The <code>TAEVENT</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table> | Value | Description | 0x0 | Positive edge | 0x1 | Negative edge | 0x2 | Reserved | 0x3 | Both edges |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Positive edge | | | | | | | | | | | | | |
| 0x1 | Negative edge | | | | | | | | | | | | | |
| 0x2 | Reserved | | | | | | | | | | | | | |
| 0x3 | Both edges | | | | | | | | | | | | | |
| 1 | TASTALL | R/W | 0 | <p>GPTM Timer A Stall Enable</p> <p>The <code>TASTALL</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer A stalling is disabled.</td> </tr> <tr> <td>1</td> <td>Timer A stalling is enabled.</td> </tr> </tbody> </table> | Value | Description | 0 | Timer A stalling is disabled. | 1 | Timer A stalling is enabled. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Timer A stalling is disabled. | | | | | | | | | | | | | |
| 1 | Timer A stalling is enabled. | | | | | | | | | | | | | |
| 0 | TAEN | R/W | 0 | <p>GPTM Timer A Enable</p> <p>The <code>TAEN</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer A is disabled.</td> </tr> <tr> <td>1</td> <td>Timer A is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register.</td> </tr> </tbody> </table> | Value | Description | 0 | Timer A is disabled. | 1 | Timer A is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Timer A is disabled. | | | | | | | | | | | | | |
| 1 | Timer A is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register. | | | | | | | | | | | | | |

Register 5: GPTM Interrupt Mask (GPTMIMR), offset 0x018

This register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x018
 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 | | | | TBMIM | CBEIM | CBMIM | TBTOIM | reserved | | | | TAMIM | RTCIM | CAEIM | CAMIM | TATOIM |
| Type | RO | RO | RO | RO | R/W | R/W | R/W | R/W | 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 |
|-----------|----------|------|----------|---|
| 31:12 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 11 | TBMIM | R/W | 0 | GPTM Timer B Mode Match Interrupt Mask The TBMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled. |
| 10 | CBEIM | R/W | 0 | GPTM Capture B 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 Capture B Match Interrupt Mask The CBMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 8 | TBTOIM | R/W | 0 | GPTM Timer B Time-Out Interrupt Mask The TBTOIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled. |
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | TAMIM | R/W | 0 | GPTM Timer A Mode Match Interrupt Mask The TAMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled. |
| 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 Capture A 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 Capture A 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 Timer A 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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|--------|--------|--------|---------|----------|----|----|-----|--------|--------|--------|--------|---------|
| | 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 | | | | TBMRIS | CBERIS | CBMRIS | TBTORIS | reserved | | | | TAMRIS | RTCRIS | CAERIS | CAMRIS | TATORIS |
| Type | RO | RO | RO | RO | R/W | RO | RO | RO | RO | RO | RO | 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 |
|-----------|----------|------|----------|---|
| 31:12 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 11 | TBMRIS | R/W | 0 | GPTM Timer B Mode Match Raw Interrupt Value Description 1 The TBMIE bit is set in the GPTMTBMR register, and the match value in the GPTMTBMATCHR register has been reached when in the one-shot and periodic modes. 0 The match value has not been reached. This bit is cleared by writing a 1 to the TBMCINT bit in the GPTMICR register. |
| 10 | CBERIS | RO | 0 | GPTM Capture B Event Raw Interrupt Value Description 1 The Capture B event has occurred. 0 The Capture B event has not occurred. This bit is cleared by writing a 1 to the CBECINT bit in the GPTMICR register. |
| 9 | CBMRIS | RO | 0 | GPTM Capture B Match Raw Interrupt Value Description 1 The Capture B match has occurred. 0 The Capture B match has not occurred. This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 8 | TBTORIS | RO | 0 | <p>GPTM Timer B Time-Out Raw Interrupt</p> <p>Value Description</p> <p>1 Timer B has timed out.</p> <p>0 Timer B has not timed out.</p> <p>This bit is cleared by writing a 1 to the <code>TBTOCINT</code> bit in the GPTMICR register.</p> |
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | TAMRIS | R/W | 0 | <p>GPTM Timer A Mode Match Raw Interrupt</p> <p>Value Description</p> <p>1 The <code>TAMIE</code> bit is set in the GPTMTAMR register, and the match value in the GPTMTAMATCHR register has been reached when in the one-shot and periodic modes.</p> <p>0 The match value has not been reached.</p> <p>This bit is cleared by writing a 1 to the <code>TAMCINT</code> bit in the GPTMICR register.</p> |
| 3 | RTCRIS | RO | 0 | <p>GPTM RTC Raw Interrupt</p> <p>Value Description</p> <p>1 The RTC event has occurred.</p> <p>0 The RTC event has not occurred.</p> <p>This bit is cleared by writing a 1 to the <code>RTCCINT</code> bit in the GPTMICR register.</p> |
| 2 | CAERIS | RO | 0 | <p>GPTM Capture A Event Raw Interrupt</p> <p>Value Description</p> <p>1 The Capture A event has occurred.</p> <p>0 The Capture A event has not occurred.</p> <p>This bit is cleared by writing a 1 to the <code>CAECINT</code> bit in the GPTMICR register.</p> |
| 1 | CAMRIS | RO | 0 | <p>GPTM Capture A Match Raw Interrupt</p> <p>Value Description</p> <p>1 The Capture A match has occurred.</p> <p>0 The Capture A match has not occurred.</p> <p>This bit is cleared by writing a 1 to the <code>CAMCINT</code> bit in the GPTMICR register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 0 | TATORIS | RO | 0 | GPTM Timer A Time-Out Raw Interrupt |
| | | | | Value Description |
| | | | | 1 Timer A has timed out. |
| | | | | 0 Timer A has not timed out. |
| | | | | This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR register. |

Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

GPTM Masked Interrupt Status (GPTMMIS)

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 | | | | | | | | | | | | | | | | |
| 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 | | | | TBMMIS | CBEMIS | CBMMIS | TBTOMIS | reserved | | | | TAMMIS | RTCMIS | CAEMIS | CAMMIS | TATOMIS |
| Type | RO | RO | RO | RO | R/W | RO | RO | RO | RO | RO | RO | 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 |
|-----------|----------|------|----------|--|
| 31:12 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 11 | TBMMIS | R/W | 0 | GPTM Timer B Mode Match Masked Interrupt Value Description 1 An unmasked Timer B Mode Match interrupt has occurred. 0 A Timer B Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the <code>TBMCINT</code> bit in the GPTMICR register. |
| 10 | CBEMIS | RO | 0 | GPTM Capture B Event Masked Interrupt Value Description 1 An unmasked Capture B event interrupt has occurred. 0 A Capture B event interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the <code>CBECINT</code> bit in the GPTMICR register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 9 | CBMMIS | RO | 0 | <p>GPTM Capture B Match Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Capture B Match interrupt has occurred.</p> <p>0 A Capture B Mode Match interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR register.</p> |
| 8 | TBTOMIS | RO | 0 | <p>GPTM Timer B Time-Out Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Timer B Time-Out interrupt has occurred.</p> <p>0 A Timer B Time-Out interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the TBTOCINT bit in the GPTMICR register.</p> |
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | TAMMIS | R/W | 0 | <p>GPTM Timer A Mode Match Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Timer A Mode Match interrupt has occurred.</p> <p>0 A Timer A Mode Match interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the TAMCINT bit in the GPTMICR register.</p> |
| 3 | RTCMIS | RO | 0 | <p>GPTM RTC Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked RTC event interrupt has occurred.</p> <p>0 An RTC event interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the RTCCINT bit in the GPTMICR register.</p> |
| 2 | CAEMIS | RO | 0 | <p>GPTM Capture A Event Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Capture A event interrupt has occurred.</p> <p>0 A Capture A event interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the CAECINT bit in the GPTMICR register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 1 | CAMMIS | RO | 0 | <p>GPTM Capture A Match Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Capture A Match interrupt has occurred.</p> <p>0 A Capture A Mode Match interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR register.</p> |
| 0 | TATOMIS | RO | 0 | <p>GPTM Timer A Time-Out Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Timer A Time-Out interrupt has occurred.</p> <p>0 A Timer A Time-Out interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR register.</p> |

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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|---------|---------|---------|----------|----------|----|----|---------|---------|---------|---------|----------|
| | 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 | | | | TBMCINT | CBECINT | CBMCINT | TBTOCINT | reserved | | | TAMCINT | RTCCINT | CAECINT | CAMCINT | TATOCINT |
| Type | RO | RO | RO | RO | W1C | W1C | W1C | W1C | RO | RO | RO | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:12 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 11 | TBMCINT | W1C | 0 | GPTM Timer B Mode Match Interrupt Clear Writing a 1 to this bit clears the TBMRIS bit in the GPTMRIS register and the TBMMIS bit in the GPTMMIS register. |
| 10 | CBECINT | W1C | 0 | GPTM Capture B Event Interrupt Clear Writing a 1 to this bit clears the CBERIS bit in the GPTMRIS register and the CBEMIS bit in the GPTMMIS register. |
| 9 | CBMCINT | W1C | 0 | GPTM Capture B Match Interrupt Clear Writing a 1 to this bit clears the CBMRIS bit in the GPTMRIS register and the CBMMIS bit in the GPTMMIS register. |
| 8 | TBTOCINT | W1C | 0 | GPTM Timer B Time-Out Interrupt Clear Writing a 1 to this bit clears the TBTORIS bit in the GPTMRIS register and the TBTOMIS bit in the GPTMMIS register. |
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | TAMCINT | W1C | 0 | GPTM Timer A Mode Match Interrupt Clear Writing a 1 to this bit clears the TAMRIS bit in the GPTMRIS register and the TAMMIS bit in the GPTMMIS register. |
| 3 | RTCCINT | W1C | 0 | GPTM RTC Interrupt Clear Writing a 1 to this bit clears the RTCRIS bit in the GPTMRIS register and the RTCMIS bit in the GPTMMIS register. |

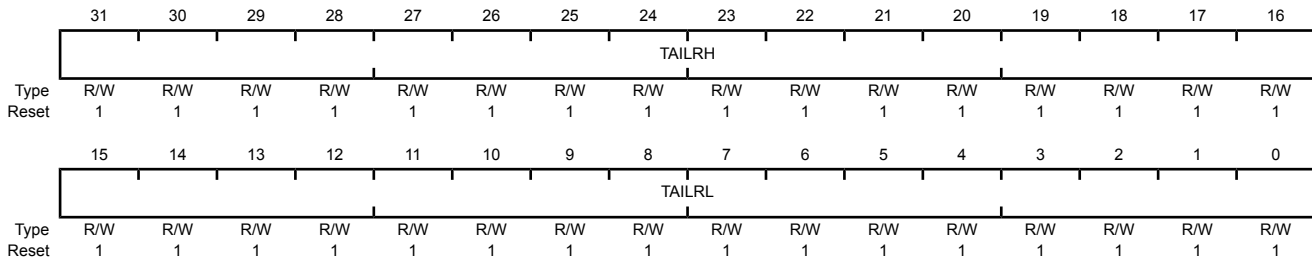
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 2 | CAECINT | W1C | 0 | GPTM Capture A Event Interrupt Clear Writing a 1 to this bit clears the CAERIS bit in the GPTMRIS register and the CAEMIS bit in the GPTMMIS register. |
| 1 | CAMCINT | W1C | 0 | GPTM Capture A Match Interrupt Clear Writing a 1 to this bit clears the CAMRIS bit in the GPTMRIS register and the CAMMIS bit in the GPTMMIS register. |
| 0 | TATOCINT | W1C | 0 | GPTM Timer A Time-Out Raw Interrupt Writing a 1 to this bit clears the TATORIS bit in the GPTMRIS register and the TATOMIS bit in the GPTMMIS register. |

Register 9: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028

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 Timer B 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 Timer A 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 Timer A Interval Load Register High When configured for 32-bit mode via the GPTMCFG register, the GPTM Timer B 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 Timer A Interval Load Register Low For both 16- and 32-bit modes, writing this field loads the counter for Timer A. A read returns the current value of GPTMTAILR . |

Register 10: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C

This register is used to load the starting count value into Timer B. When the GPTM is configured to a 32-bit mode, **GPTMTBILR** returns the current value of Timer B and ignores writes.

GPTM Timer B 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

| | 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 |
| | TBILRL | | | | | | | | | | | | | | | |
| 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 |
|-----------|----------|------|--------|---|
| 31:16 | 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 Timer B Interval Load Register |

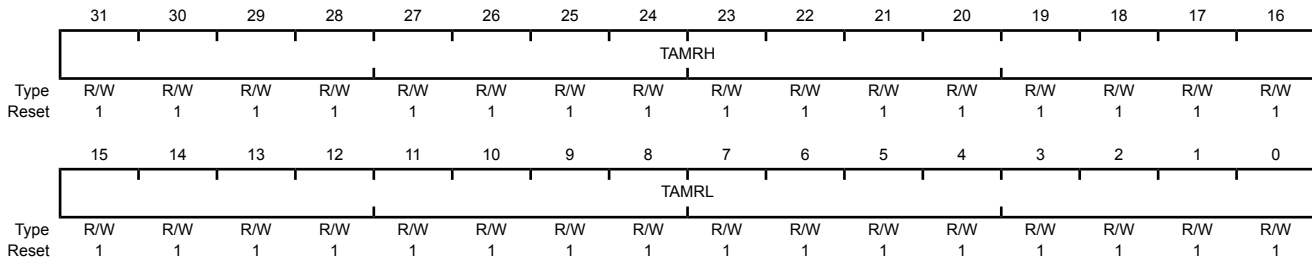
When the GPTM is not configured as a 32-bit timer, a write to this field updates **GPTMTBILR**. In 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

Register 11: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode. In 16-bit Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

GPTM Timer A 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 Timer A Match Register High When the timer is configured for 32-bit 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 Timer A Match Register Low When the timer is configured for 32-bit mode via the GPTMCFG register, this value is compared to the lower half of GPTMTAR , to determine match events. When the timer is configured for 16-bit mode via the GPTMCFG register, this value is compared to GPTMTAR to determine match events. When configured for 16-bit 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. When configured for PWM mode, this value along with GPTMTAILR , determines the duty cycle of the output PWM signal. |

Register 12: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode. In 16-bit Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

GPTM Timer B 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

| | 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 |
| | TBMRL | | | | | | | | | | | | | | | |
| 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 |

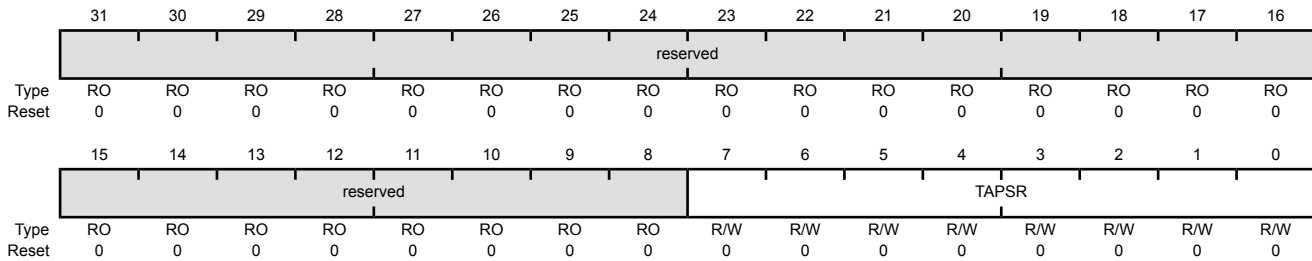
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TBMRL | R/W | 0xFFFF | GPTM Timer B Match Register Low When the timer is configured for 16-bit mode via the GPTMCFG register, this value is compared to GPTMTBR to determine match events. When configured for 16-bit 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. When configured for PWM mode, this value along with GPTMTBILR , determines the duty cycle of the output PWM signal. |

Register 13: GPTM Timer A Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the 16-bit timers.

GPTM Timer A 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 | 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 | TAPSR | R/W | 0x00 | GPTM Timer A Prescale |

The register loads this value on a write. A read returns the current value of the register.

Refer to Table 12-3 on page 439 for more details and an example.

Register 14: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C

This register allows software to extend the range of the 16-bit timers.

GPTM Timer B 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-------|-----|-----|-----|-----|-----|-----|-----|
| | 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 | | | | | | | | TBPSR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TBPSR | R/W | 0x00 | GPTM Timer B Prescale The register loads this value on a write. A read returns the current value of this register. |

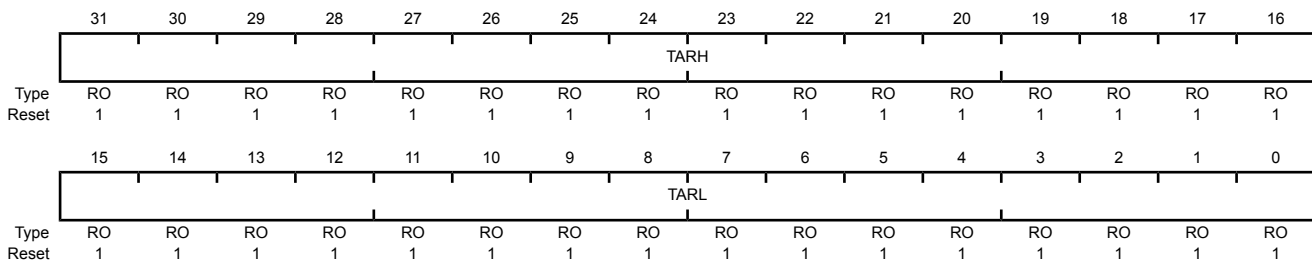
Refer to Table 12-3 on page 439 for more details and an example.

Register 15: GPTM Timer A (GPTMTAR), offset 0x048

This register shows the current value of the Timer A counter in all cases except for Input Edge-Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM Timer A (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 Timer A Register High If the GPTMCFG is in a 32-bit mode, Timer B value is read. If the GPTMCFG is in a 16-bit mode, this is read as zero. |
| 15:0 | TARL | RO | 0xFFFF | GPTM Timer A Register Low A read returns the current value of the GPTM Timer A Count Register , except in Input Edge-Count mode, when it returns the timestamp from the last edge event. |

Register 16: GPTM Timer B (GPTMTBR), offset 0x04C

This register shows the current value of the Timer B counter in all cases except for Input Edge-Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM Timer B (GPTMTBR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x04C
 Type RO, reset 0x0000.FFFF

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 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 |
| | TBRL | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | 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 Timer B |

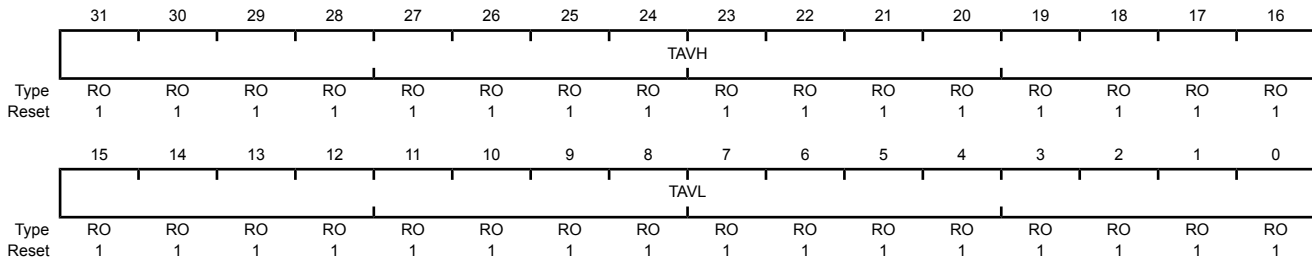
A read returns the current value of the **GPTM Timer B Count Register**, except in Input Edge-Count mode, when it returns the timestamp from the last edge event.

Register 17: GPTM Timer A Value (GPTMTAV), offset 0x050

This register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry.

GPTM Timer A Value (GPTMTAV)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x050
 Type RO, reset 0xFFFF.FFFF



| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|--------|---|
| 31:16 | TAVH | RO | 0xFFFF | GPTM Timer A Value High If the GPTMCFG is configured for 32-bit mode, the Timer B value is read. If the GPTMCFG is configured for 16-bit mode, this is read as zero. |
| 15:0 | TAVL | RO | 0xFFFF | GPTM Timer A Register Low A read returns the current value of Timer A. |

Register 18: GPTM Timer B Value (GPTMTBV), offset 0x054

This register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry.

GPTM Timer B Value (GPTMTBV)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x054
 Type RO, reset 0x0000.FFFF

| | 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 |
| | TBVL | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | 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 | TBVL | RO | 0xFFFF | GPTM Timer B Register A read returns the current value of Timer B. |

13 Watchdog Timers

A watchdog timer can generate a nonmaskable interrupt (NMI) 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 LM3S9B90 microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other is clocked by the PIOSC (Watchdog Timer 1). The two modules are identical except that WDT1 is in a different clock domain, and therefore requires synchronizers. As a result, WDT1 has a bit defined in the **Watchdog Timer Control (WDTCTL)** register to indicate when a write to a WDT1 register is complete. Software can use this bit to ensure that the previous access has completed before starting the next access.

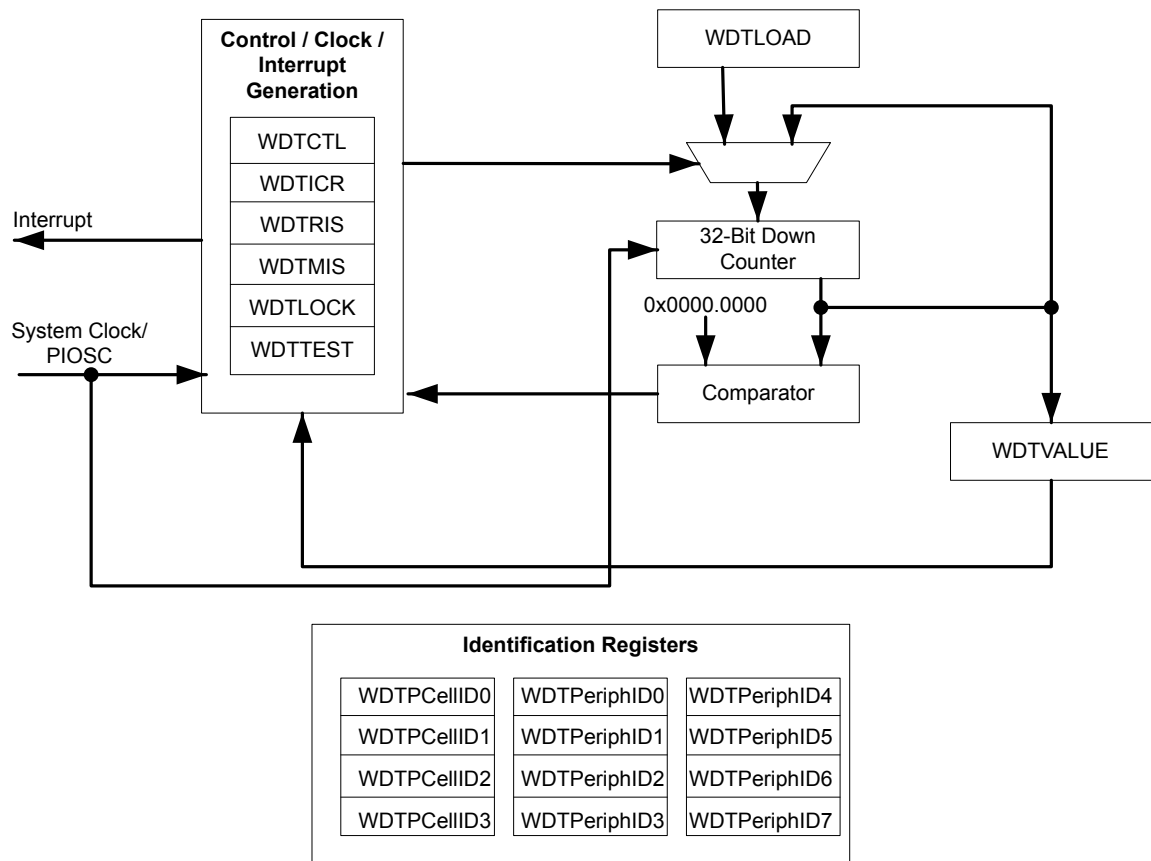
The Stellaris[®] LM3S9B90 controller has two Watchdog Timer modules with the following features:

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

The Watchdog Timer can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

13.1 Block Diagram

Figure 13-1. WDT Module Block Diagram



13.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the `RESEN` bit in the **WDTCTL** register, the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

13.2.1 Register Access Timing

Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The **WRC** bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for **WRC=1** prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock.

13.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the **WDT** bit in the **RCGC0** register, see page 166.

The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If WDT1, wait for the **WRC** bit in the **WDTCTL** register to be set.
3. If the Watchdog is configured to trigger system resets, set the **RESEN** bit in the **WDTCTL** register.
4. If WDT1, wait for the **WRC** bit in the **WDTCTL** register to be set.
5. Set the **INTEN** bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of 0x1ACC.E551.

13.4 Register Map

Table 13-1 on page 479 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address:

- WDT0: 0x4000.0000
- WDT1: 0x4000.1000

Note that the Watchdog Timer module clock must be enabled before the registers can be programmed (see page 166).

Table 13-1. Watchdog Timers Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|--|--------------------------------------|----------|
| 0x000 | WDTLOAD | R/W | 0xFFFF.FFFF | Watchdog Load | 480 |
| 0x004 | WDTVALUE | RO | 0xFFFF.FFFF | Watchdog Value | 481 |
| 0x008 | WDTCTL | R/W | 0x0000.0000 (WDT0) 0x8000.0000 (WDT1) | Watchdog Control | 482 |
| 0x00C | WDTICR | WO | - | Watchdog Interrupt Clear | 484 |
| 0x010 | WDTRIS | RO | 0x0000.0000 | Watchdog Raw Interrupt Status | 485 |
| 0x014 | WDTMIS | RO | 0x0000.0000 | Watchdog Masked Interrupt Status | 486 |
| 0x418 | WDTTEST | R/W | 0x0000.0000 | Watchdog Test | 487 |
| 0xC00 | WDTLOCK | R/W | 0x0000.0000 | Watchdog Lock | 488 |
| 0xFD0 | WDTPeriphID4 | RO | 0x0000.0000 | Watchdog Peripheral Identification 4 | 489 |
| 0xFD4 | WDTPeriphID5 | RO | 0x0000.0000 | Watchdog Peripheral Identification 5 | 490 |
| 0xFD8 | WDTPeriphID6 | RO | 0x0000.0000 | Watchdog Peripheral Identification 6 | 491 |
| 0xFDC | WDTPeriphID7 | RO | 0x0000.0000 | Watchdog Peripheral Identification 7 | 492 |
| 0xFE0 | WDTPeriphID0 | RO | 0x0000.0005 | Watchdog Peripheral Identification 0 | 493 |
| 0xFE4 | WDTPeriphID1 | RO | 0x0000.0018 | Watchdog Peripheral Identification 1 | 494 |
| 0xFE8 | WDTPeriphID2 | RO | 0x0000.0018 | Watchdog Peripheral Identification 2 | 495 |
| 0xFEC | WDTPeriphID3 | RO | 0x0000.0001 | Watchdog Peripheral Identification 3 | 496 |
| 0xFF0 | WDTPrimeCellID0 | RO | 0x0000.000D | Watchdog PrimeCell Identification 0 | 497 |
| 0xFF4 | WDTPrimeCellID1 | RO | 0x0000.00F0 | Watchdog PrimeCell Identification 1 | 498 |
| 0xFF8 | WDTPrimeCellID2 | RO | 0x0000.0006 | Watchdog PrimeCell Identification 2 | 499 |
| 0xFFC | WDTPrimeCellID3 | RO | 0x0000.00B1 | Watchdog PrimeCell Identification 3 | 500 |

13.5 Register Descriptions

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

Register 1: Watchdog Load (WDTLOAD), offset 0x000

This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

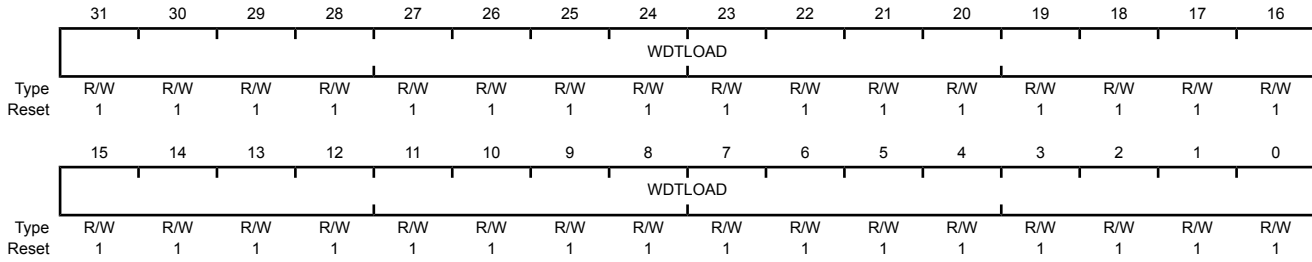
Watchdog Load (WDTLOAD)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x000

Type R/W, reset 0xFFFF.FFFF



| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------------|---------------------|
| 31:0 | WDTLOAD | R/W | 0xFFFF.FFFF | Watchdog Load Value |

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x004

Type RO, reset 0xFFFF.FFFF

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | WDTVALUE | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | WDTVALUE | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

Important: Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The `WRC` bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for `WRC=1` prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock and therefore does not have a `WRC` bit.

Watchdog Control (WDTCTL)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x008

Type R/W, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1)

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | WRC | reserved | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | RESEN | INTEN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-------------|
|-----------|------|------|-------|-------------|

| | | | | |
|----|-----|----|---|----------------|
| 31 | WRC | RO | 1 | Write Complete |
|----|-----|----|---|----------------|

The `WRC` values are defined as follows:

| Value | Description |
|-------|-------------|
|-------|-------------|

| | |
|---|---|
| 0 | A write access to one of the WDT1 registers is in progress. |
|---|---|

| | |
|---|---|
| 1 | A write access is not in progress, and WDT1 registers can be read or written. |
|---|---|

Note: This bit is reserved for WDT0 and has a reset value of 0.

| | | | | |
|------|----------|----|-----------|---|
| 30:2 | reserved | RO | 0x000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
|------|----------|----|-----------|---|

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|--|
| 1 | RESEN | R/W | 0 | Watchdog Reset Enable 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.

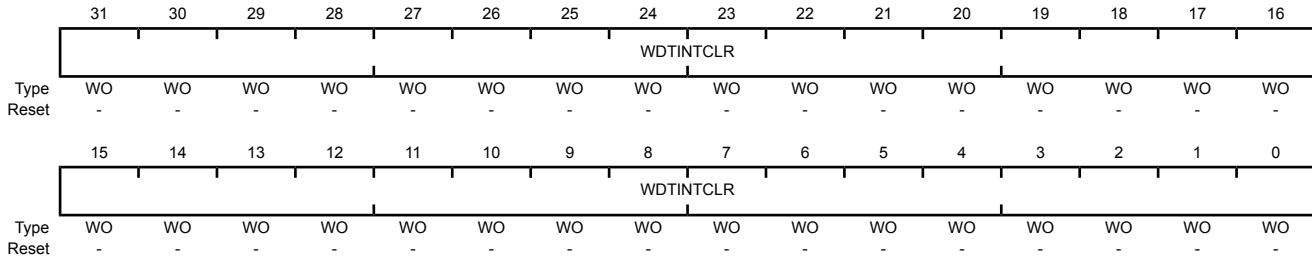
Watchdog Interrupt Clear (WDTICR)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x00C

Type WO, reset -



| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------|------|-------|--------------------------|
| 31:0 | WDTINTCLR | WO | - | Watchdog Interrupt Clear |

Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

Watchdog Raw Interrupt Status (WDTRIS)

WDT0 base: 0x4000.0000
 WDT1 base: 0x4000.1000
 Offset 0x010
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|
| | 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 | | | | | | | | | | | | | | | WDTRIS |
| 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 |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | WDTRIS | RO | 0 | Watchdog Raw Interrupt Status |

Value Description

| | |
|---|---|
| 1 | A watchdog time-out event has occurred. |
| 0 | The watchdog has not timed out. |

Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

Watchdog Masked Interrupt Status (WDTMIS)

WDT0 base: 0x4000.0000
 WDT1 base: 0x4000.1000
 Offset 0x014
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|
| | 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 | | | | | | | | | | | | | | | WDTMIS |
| 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 |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | WDTMIS | RO | 0 | Watchdog Masked Interrupt Status |
| | | | | Value Description |
| | | | | 1 A watchdog time-out event has been signalled to the interrupt controller. |
| | | | | 0 The watchdog has not timed out or the watchdog timer interrupt is masked. |

Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Watchdog Test (WDTTEST)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x418

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|-------|----------|----|----|----|----|----|----|----|
| | 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 | | | | | | | STALL | reserved | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | R/W | 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 |

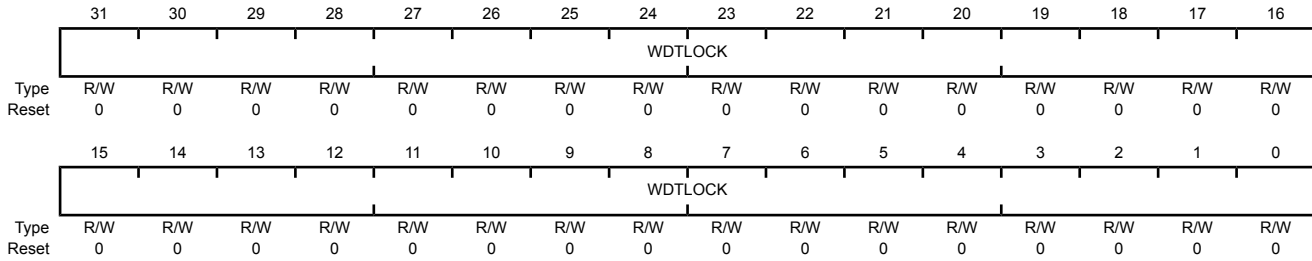
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:9 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | STALL | R/W | 0 | Watchdog Stall Enable Value Description 1 If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting. 0 The watchdog timer continues counting if the microcontroller is stopped with a debugger. |
| 7:0 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Watchdog Lock (WDTLOCK)

WDT0 base: 0x4000.0000
 WDT1 base: 0x4000.1000
 Offset 0xC00
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------------|---------------|
| 31:0 | WDTLOCK | R/W | 0x0000.0000 | 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)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFD0

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | PID4 | | | | | | | |
| 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 |

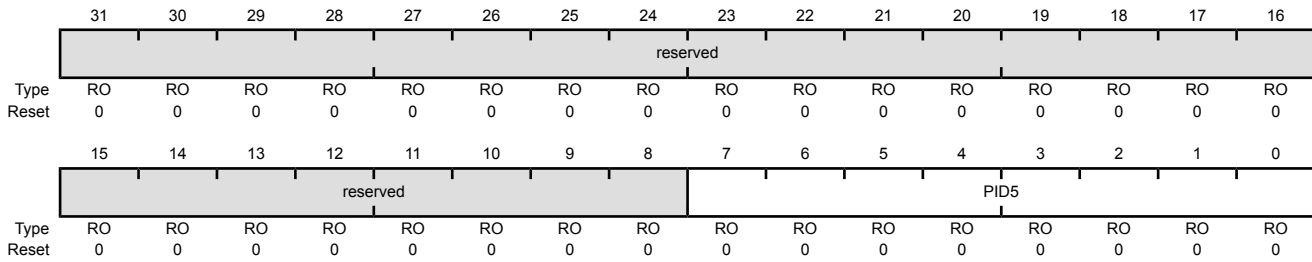
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | WDT Peripheral ID Register [7:0] |

Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 5 (WDTPeriphID5)

WDT0 base: 0x4000.0000
 WDT1 base: 0x4000.1000
 Offset 0xFD4
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | WDT Peripheral ID Register [15:8] |

Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 6 (WDTPeriphID6)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFD8

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID6 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | WDT Peripheral ID Register [23:16] |

Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 7 (WDTPeriphID7)

WDT0 base: 0x4000.0000
 WDT1 base: 0x4000.1000
 Offset 0xFDC
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID7 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | WDT Peripheral ID Register [31:24] |

Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 0 (WDTPeriphID0)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFE0

Type RO, reset 0x0000.0005

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID0 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x05 | Watchdog Peripheral ID Register [7:0] |

Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 1 (WDTPeriphID1)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFE4

Type RO, reset 0x0000.0018

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID1 | | | | | | | |
| 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 | 1 | 1 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x18 | Watchdog Peripheral ID Register [15:8] |

Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 2 (WDTPeriphID2)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFE8

Type RO, reset 0x0000.0018

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID2 | | | | | | | |
| 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 | 1 | 1 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | Watchdog Peripheral ID Register [23:16] |

Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Peripheral Identification 3 (WDTPeriphID3)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFEC

Type RO, reset 0x0000.0001

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID3 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | Watchdog Peripheral ID Register [31:24] |

Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 0 (WDTPCellID0)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFF0

Type RO, reset 0x0000.000D

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID0 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 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. |
| 7:0 | CID0 | RO | 0x0D | Watchdog PrimeCell ID Register [7:0] |

Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 1 (WDTPCellID1)

WDT0 base: 0x4000.0000
 WDT1 base: 0x4000.1000
 Offset 0xFF4
 Type RO, reset 0x0000.00F0

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | CID1 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | Watchdog PrimeCell ID Register [15:8] |

Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 2 (WDTPCellID2)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFF8

Type RO, reset 0x0000.0006

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID2 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x06 | Watchdog PrimeCell ID Register [23:16] |

Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog PrimeCell Identification 3 (WDTPCellID3)

WDT0 base: 0x4000.0000
 WDT1 base: 0x4000.1000
 Offset 0xFFC
 Type RO, reset 0x0000.00B1

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID3 | | | | | | | |
| 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 | 1 | 1 | 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. |
| 7:0 | CID3 | RO | 0xB1 | Watchdog PrimeCell ID Register [31:24] |

14 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. Two identical converter units are included, which share sixteen input channels. The two converter units may be sampled in the same processor clock or out of phase with each other.

The Stellaris® ADC module features 10-bit conversion resolution and supports sixteen input channels, plus an internal temperature sensor. The ADC module contains four programmable sequencers allowing the sampling of multiple analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. A digital comparator function is included which allows the conversion value to be diverted to a digital comparator module. The digital comparator module provides digital comparator. The comparator module measures the ADC conversion value against two user-defined values to determine the operational range of the signal.

The Stellaris® LM3S9B90 microcontroller provides two ADC modules with the following features:

- Sixteen 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 sequencers 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
- Digital comparison unit providing 16 digital comparators
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (μDMA)
 - Dedicated channel for each sample sequencer
 - Burst request asserted when interrupt is triggered

14.1 Block Diagram

The Stellaris[®] microcontroller contains two identical Analog-to-Digital Converter units. These two modules, ADC0 and ADC1, share the same sixteen analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers. Figure 14-1 on page 502 shows how the two modules are connected to analog inputs and the system bus.

Figure 14-1. Implementation of Two ADC Blocks

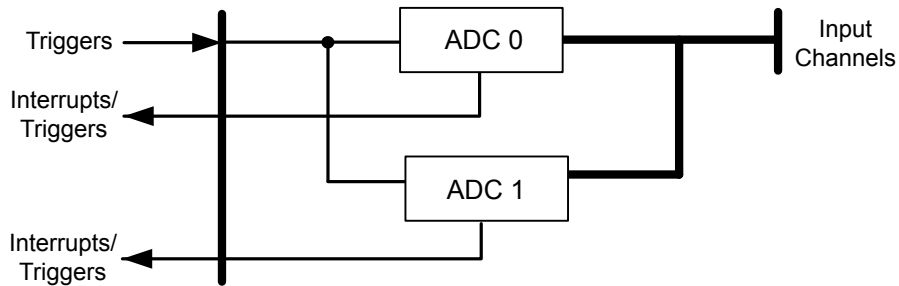
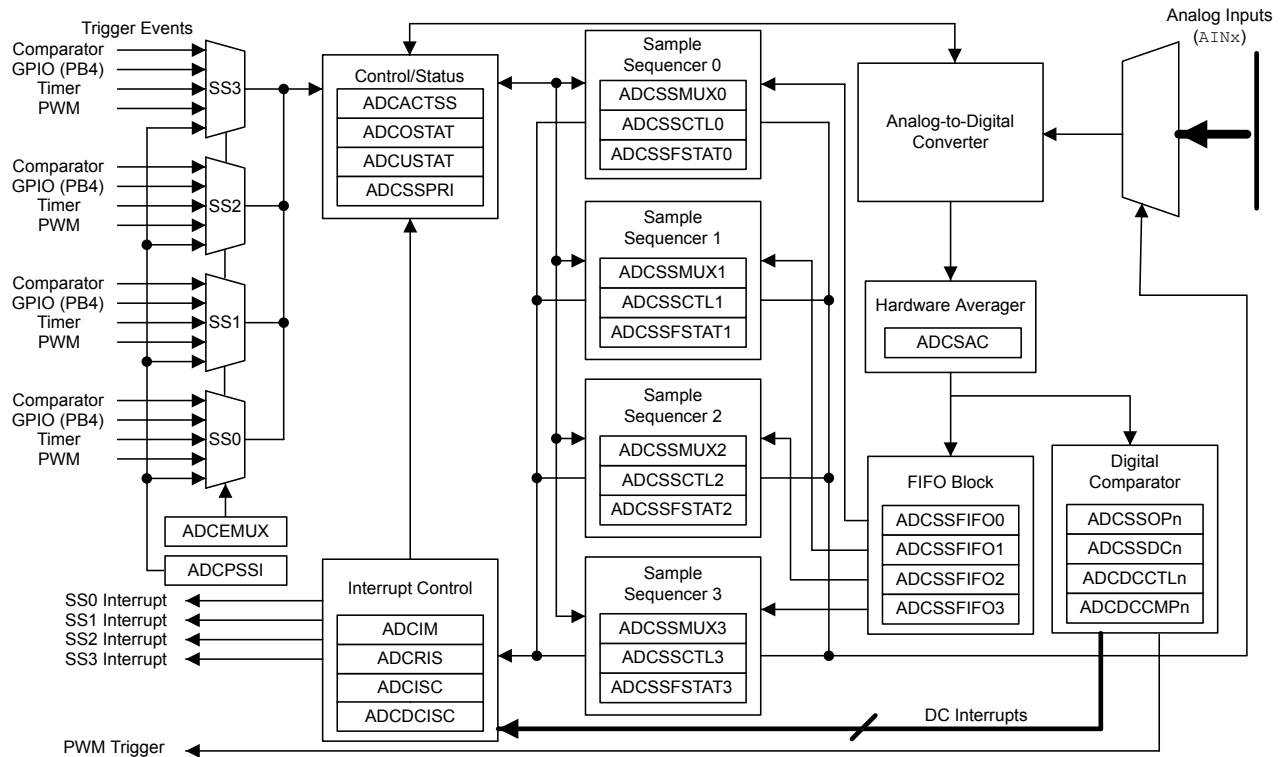


Figure 14-2 on page 502 provides details on the internal configuration of the ADC controls and data registers.

Figure 14-2. ADC Module Block Diagram



14.2 Signal Description

Table 14-1 on page 503 lists the external signals of the ADC module and describes the function of each. The ADC signals are analog functions for some GPIO signals. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the ADC signals. Note that when a pin is used as an ADC input, the appropriate bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register must be set to disable the analog isolation circuit, and the appropriate bit in the **GPIO Digital Enable (GPIODEN)** register must be clear to disable digital function. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318.

Table 14-1. Signals for ADC

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|---|
| AIN0 | 1 | PE7 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | 2 | PE6 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | 5 | PE5 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | 6 | PE4 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | 100 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | 99 | PD6 | I | Analog | Analog-to-digital converter input 5. |
| AIN6 | 98 | PD5 | I | Analog | Analog-to-digital converter input 6. |
| AIN7 | 97 | PD4 | I | Analog | Analog-to-digital converter input 7. |
| AIN8 | 96 | PE3 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | 95 | PE2 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | 92 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | 91 | PB5 | I | Analog | Analog-to-digital converter input 11. |
| AIN12 | 13 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | 12 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | 11 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | 10 | PD0 | I | Analog | Analog-to-digital converter input 15. |
| VREFA | 90 | PB6 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AIN _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-2 on page 1039. |

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

14.3 Functional Description

The Stellaris® ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence. The μ DMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

14.3.1 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 14-2 on page 504 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 14-2. Samples and FIFO Depth of Sequencers

| Sequencer | Number of Samples | Depth of FIFO |
|-----------|-------------------|---------------|
| SS3 | 1 | 1 |
| SS2 | 4 | 4 |
| SS1 | 4 | 4 |
| SS0 | 8 | 8 |

For a given sample sequence, each sample is defined by 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. Sampling is then initiated by setting the **SSn** bit in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. In addition, sample sequences may be initiated on multiple ADC modules simultaneously using the **GSYNC** and **SYNCWAIT** bits in the **ADCPSSI** register during the configuration of each ADC module. For more information on using these bits, refer to page 535.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence is allowed. In the **ADCSSCTLn** register, the **IE_n** 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 (ADCSSFIFO_n)** 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 (ADCSSFSTAT_n)** registers along with **FULL** and **EMPTY** status flags. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

14.3.2 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
- Comparator configuration

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured for 16-MHz operation automatically by hardware when the system **XTAL** is selected.

14.3.2.1 Interrupts

The register configurations of the sample sequencers and digital comparators dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the `MASK` bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of the various interrupt signals; and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding `IN` bit in **ADCISC**. Digital comparator interrupts are cleared by writing a 1 to the **ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)** register.

14.3.2.2 DMA Operation

The ADC module provides a request signal to the μ DMA controller for each sample sequencer. Each sample sequencer has a dedicated μ DMA channel. The request signal is a burst type and is asserted whenever an interrupt is enabled in a sample sequence (`IE` bit in the **ADCSSCTLn** register is set). Single requests are not supported.

The arbitration size of the μ DMA transfer must be a power of 2, and the associated `IE` bits in the **ADDSSCTLn** register must be set. For example, if the μ DMA channel of SS0 has an arbitration size of four, the `IE3` bit (4th sample) and the `IE7` bit (8th sample) must be set. Thus the μ DMA request occurs every time 4 samples have been acquired. No other special steps are needed to enable the ADC module for μ DMA operation.

Refer to the “Micro Direct Memory Access (μ DMA)” on page 258 for more details about programming the μ DMA controller.

14.3.2.3 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the **ADC Sample Sequencer Priority (ADCSSPRI)** register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active sample sequencer units with the same priority do not provide consistent results, so software must ensure that all active sample sequencer units have a unique priority value.

14.3.2.4 Sampling Events

Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), analog comparators, an external signal on GPIO `PB4`, a GP Timer, and continuous sampling. 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 continuous sampling trigger. If a sequencer's priority is too high, it is possible to starve other lower priority sequencers.

14.3.2.5 External Voltage Reference

An external reference voltage may be provided to serve as the maximum conversion value reference. The `VREF` bit in the **ADC Control (ADCCTL)** register specifies whether to use the internal or external reference. The `VREFA` specification defines the useful range for the external voltage reference, see Table 26-2 on page 1039. Ground is always used as the reference level for the minimum conversion value. Care must be taken to supply a reference voltage of acceptable quality.

14.3.3 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 537). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

14.3.4 Analog-to-Digital Converter

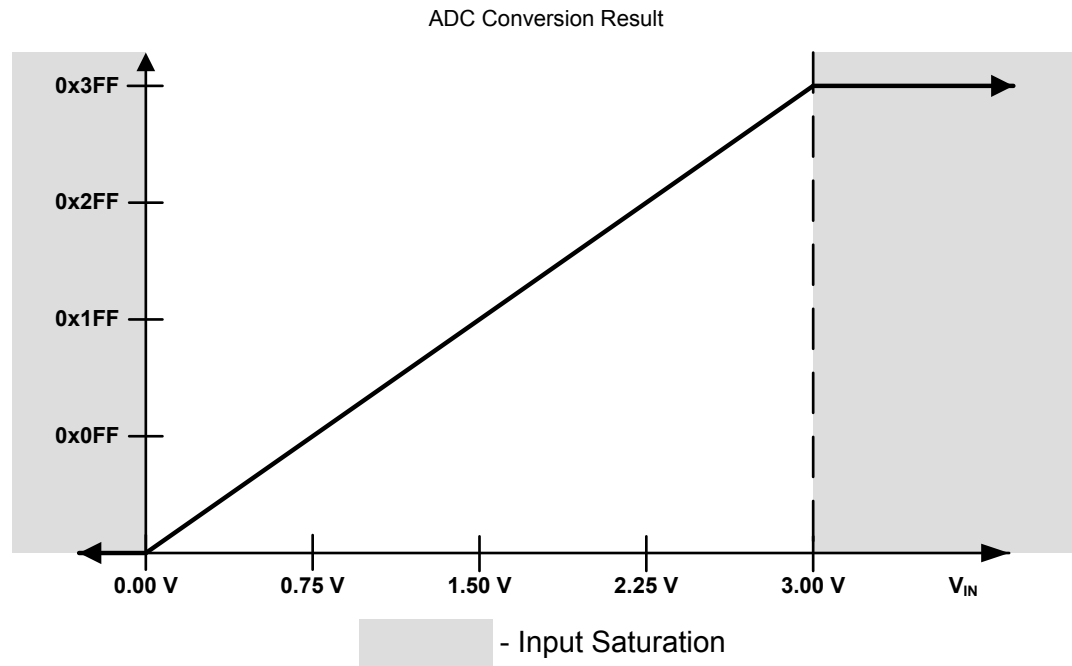
The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 10-bit, low-power, high-precision conversion value. The successive-approximation algorithm uses a current mode D/A converter to achieve lower settling time, resulting in higher conversion speeds for the A/D converter. In addition, built-in sample-and-hold circuitry with offset-calibration circuitry improves conversion accuracy. The ADC must be run from the PLL or a 14- to 18-MHz clock source.

The ADC operates from the 3.3-V analog and 1.2-V digital power supply. Integrated shutdown modes are available to reduce power consumption when ADC conversions are not required. The analog inputs are connected to the ADC through custom pads and specially balanced input paths to minimize the distortion on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in “Analog-to-Digital Converter” on page 1053.

14.3.4.1 Internal Voltage Reference

The band-gap circuitry generates an internal 3.0 V reference that can be used by the ADC to produce a conversion value from the selected analog input. The range of this conversion value is from 0x000 to 0x3FF. In single-ended-input mode, the 0x000 value corresponds to an analog input voltage of 0.0 V; the 0x3FF value corresponds to an analog input voltage of 3.0 V. This configuration results in a resolution of approximately 2.9 mV per ADC code. While the analog input pads can handle voltages beyond this range, the ADC conversions saturate in under-voltage and over-voltage cases. Figure 14-3 on page 507 shows the ADC conversion function of the analog inputs.

Figure 14-3. Internal Voltage Conversion Result

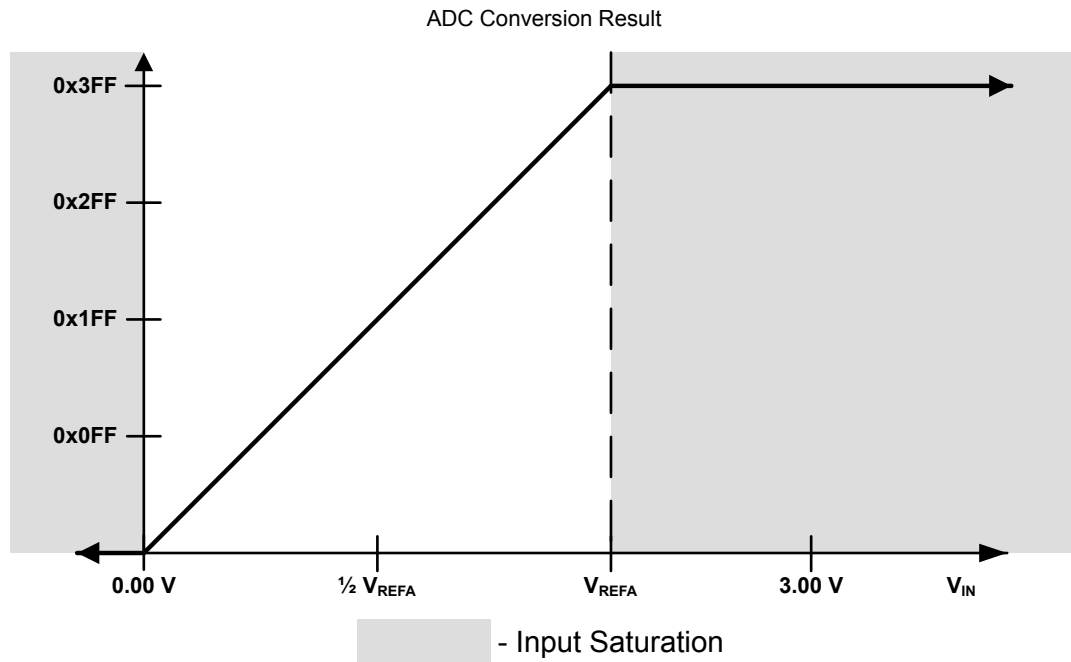


14.3.4.2 External Voltage Reference

The ADC can use an external voltage reference to produce the conversion value from the selected analog input by setting the V_{REF} bit in the **ADC Control (ADCCTL)** register. While the range of the conversion value remains the same (0x000 to 0x3FF), the analog voltage associated with the 0x3FF value corresponds to the value of the external voltage reference, resulting in a smaller voltage resolution per ADC code. Analog input voltages above the external voltage reference saturate to 0x3FF while those below 0.0 V continue to saturate at 0x000. Figure 14-4 on page 508 shows the ADC conversion function of the analog inputs when using an external voltage reference.

The external voltage reference can be more accurate than the internal reference by using a high-precision source or trimming the source.

Figure 14-4. External Voltage Conversion Result



14.3.5 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 D_n bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** register. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 14-3 on page 508). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 14-3. Differential Sampling Pairs

| Differential Pair | Analog Inputs |
|-------------------|---------------|
| 0 | 0 and 1 |
| 1 | 2 and 3 |
| 2 | 4 and 5 |
| 3 | 6 and 7 |
| 4 | 8 and 9 |
| 5 | 10 and 11 |
| 6 | 12 and 13 |
| 7 | 14 and 15 |

The voltage sampled in differential mode is the difference between the odd and even channels:

$$\Delta V \text{ (differential voltage)} = V_{IN_EVEN} \text{ (even channels)} - V_{IN_ODD} \text{ (odd channels)}, \text{ therefore:}$$

- 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 14-5 on page 509 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 14-6 on page 510 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 14-7 on page 510 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 14-5. Differential Sampling Range, $V_{IN_ODD} = 1.5$ V

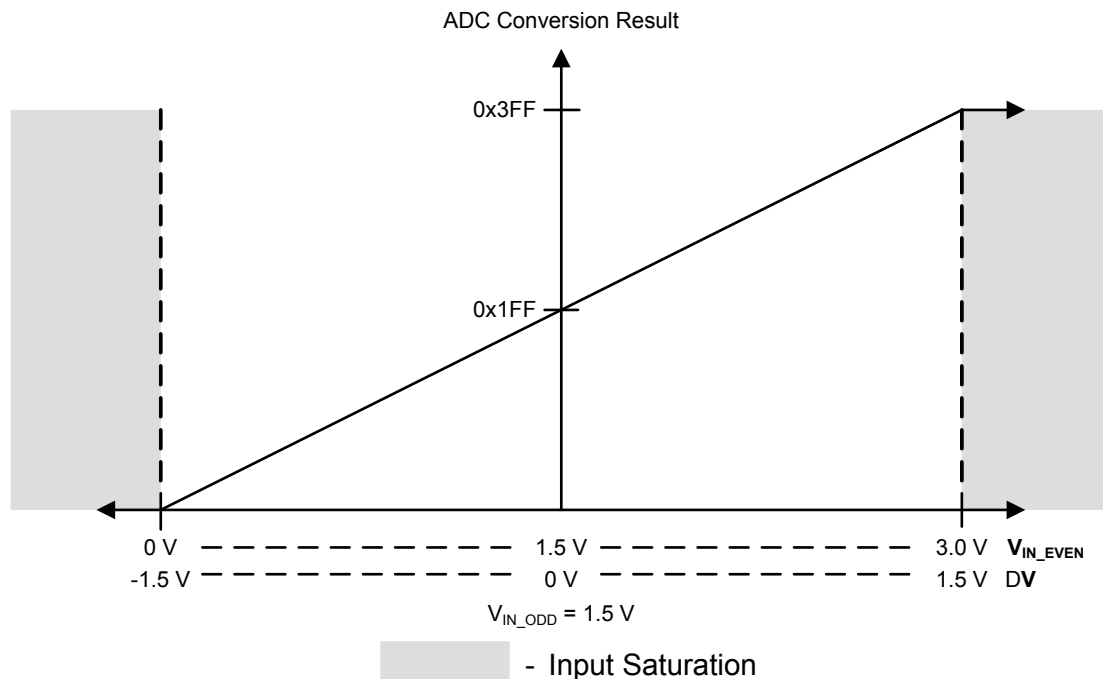


Figure 14-6. Differential Sampling Range, $V_{IN_ODD} = 0.75\text{ V}$

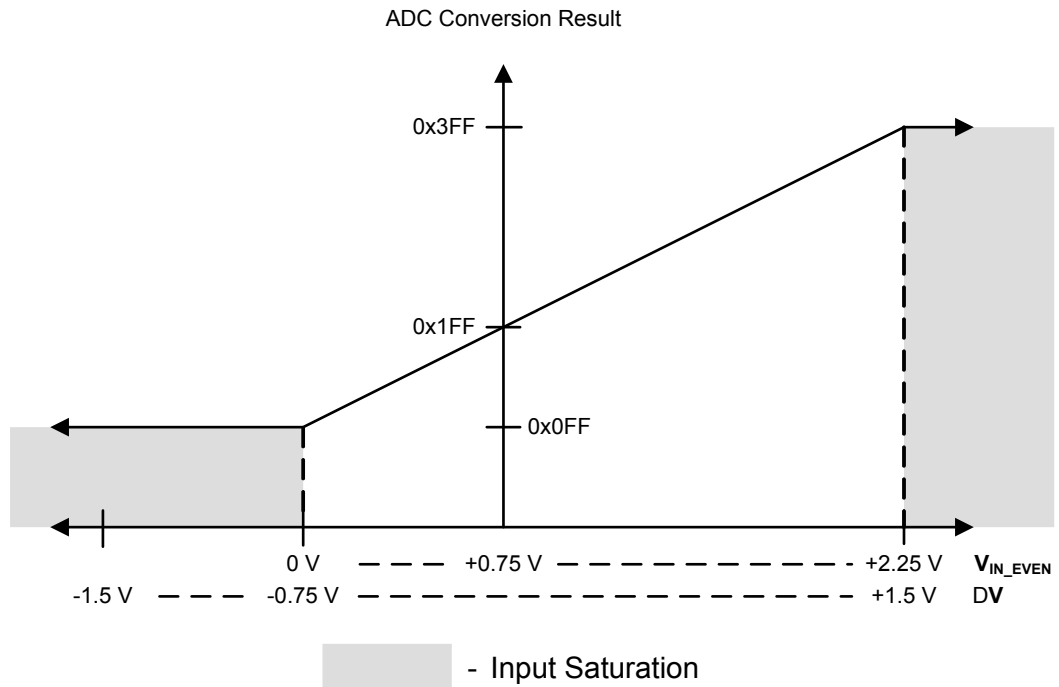
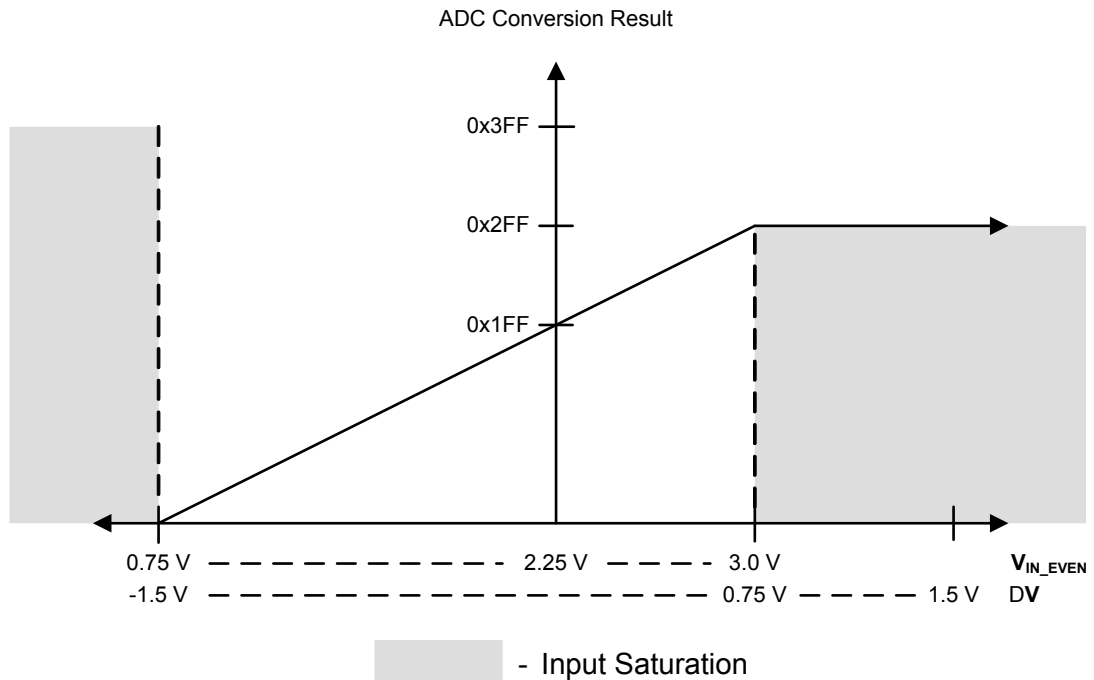


Figure 14-7. Differential Sampling Range, $V_{IN_ODD} = 2.25\text{ V}$



14.3.6 Internal Temperature Sensor

The temperature sensor serves two primary purposes: 1) to notify the system that internal temperature is too high or low for reliable operation and 2) to provide temperature measurements for calibration of the Hibernate module RTC trim value.

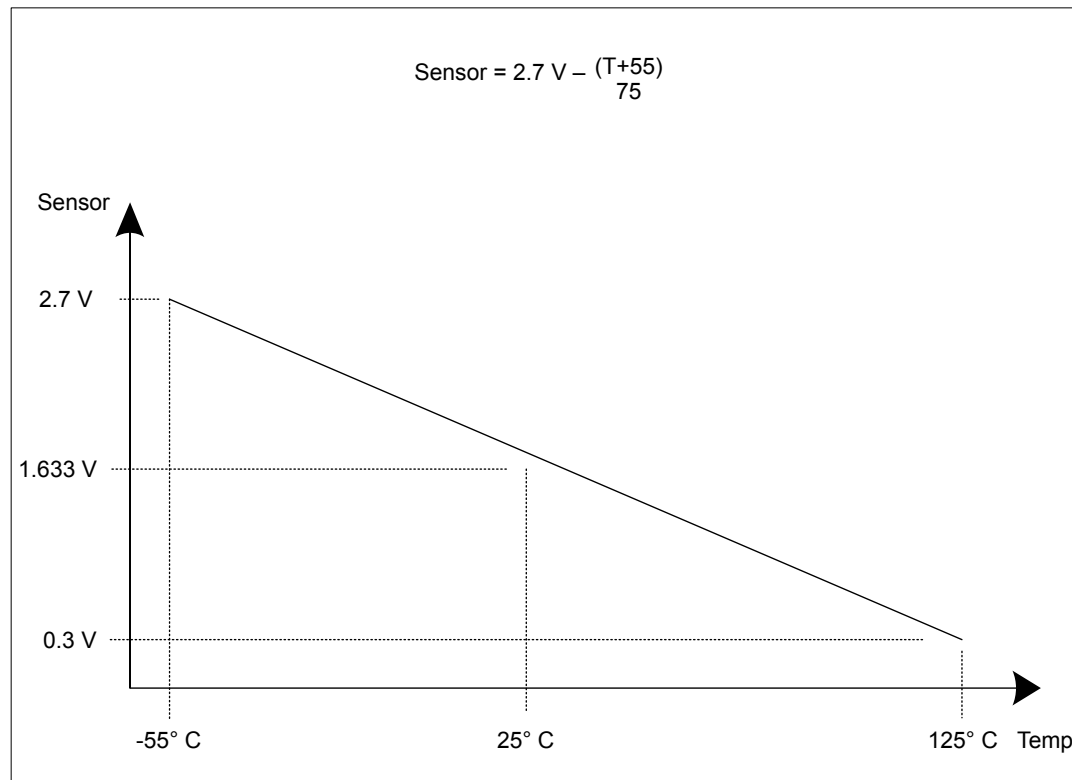
The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC. In addition, the temperature sensor has a second power-down input in the 3.3 V domain which provides control by the Hibernation module.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. 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 14-8 on page 511.

Figure 14-8. Internal Temperature Sensor Characteristic



The temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (in °C) based on the ADC reading:

$$\text{Temperature} = 147.5 - ((225 \times \text{ADC}) / 1023)$$

14.3.7 Digital Comparator Unit

An ADC is commonly used to sample an external signal and to monitor its value to ensure that it remains in a given range. To automate this monitoring procedure and reduce the amount of processor overhead that is required, digital comparators are provided. Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the **ADC Digital**

Comparator Range (ADCDCMPn) registers. If the observed signal moves out of the acceptable range, a processor interrupt can be generated. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be applied to three separate regions (low band, mid band, high band) as defined by the user.

14.3.7.1 Output Functions

ADC conversions can either be stored in the ADC Sample Sequence FIFOs or compared using the digital comparator resources as defined by the S_nDCOP bits in the **ADC Sample Sequence n Operation (ADCSSOPn)** register. These selected ADC conversions are used by their respective digital comparator to monitor the external signal. Each comparator has two possible output functions: processor interrupts and triggers.

Each function has its own state machine to track the monitored signal. Even though the interrupt and trigger functions can be enabled individually or both at the same time, the same conversion data is used by each function to determine if the right conditions have been met to assert the associated output.

Interrupts

The digital comparator interrupt function is enabled by setting the CIE bit in the **ADC Digital Comparator Control (ADCDCCTLn)** register. This bit enables the interrupt function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, and the $DCONSS_x$ bit is set in the **ADCIM** register, an interrupt is sent to the interrupt controller.

Triggers

The digital comparator trigger function is enabled by setting the CTE bit in the **ADCDCCTLn** register. This bit enables the trigger function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, the corresponding digital comparator trigger to the PWM module is asserted

14.3.7.2 Operational Modes

Four operational modes are provided to support a broad range of applications and multiple possible signaling requirements: Always, Once, Hysteresis Always, and Hysteresis Once. The operational mode is selected using the CIM or CTM field in the **ADCDCCTLn** register.

Always Mode

In the Always operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria. The result is a string of assertions on the interrupt or trigger while the conversions are within the appropriate range.

Once Mode

In the Once operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria, and the previous ADC conversion value did not. The result is a single assertion of the interrupt or trigger when the conversions are within the appropriate range.

Hysteresis-Always Mode

The Hysteresis-Always operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Always mode, the associated interrupt or trigger is asserted in the following cases: 1) the ADC conversion value meets its comparison criteria or 2)

a previous ADC conversion value has met the comparison criteria, and the hysteresis condition has not been cleared by entering the opposite region. The result is a string of assertions on the interrupt or trigger that continue until the opposite region is entered.

Hysteresis-Once Mode

The Hysteresis-Once operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Once mode, the associated interrupt or trigger is asserted only when the ADC conversion value meets its comparison criteria, the hysteresis condition is clear, and the previous ADC conversion did not meet the comparison criteria. The result is a single assertion on the interrupt or trigger.

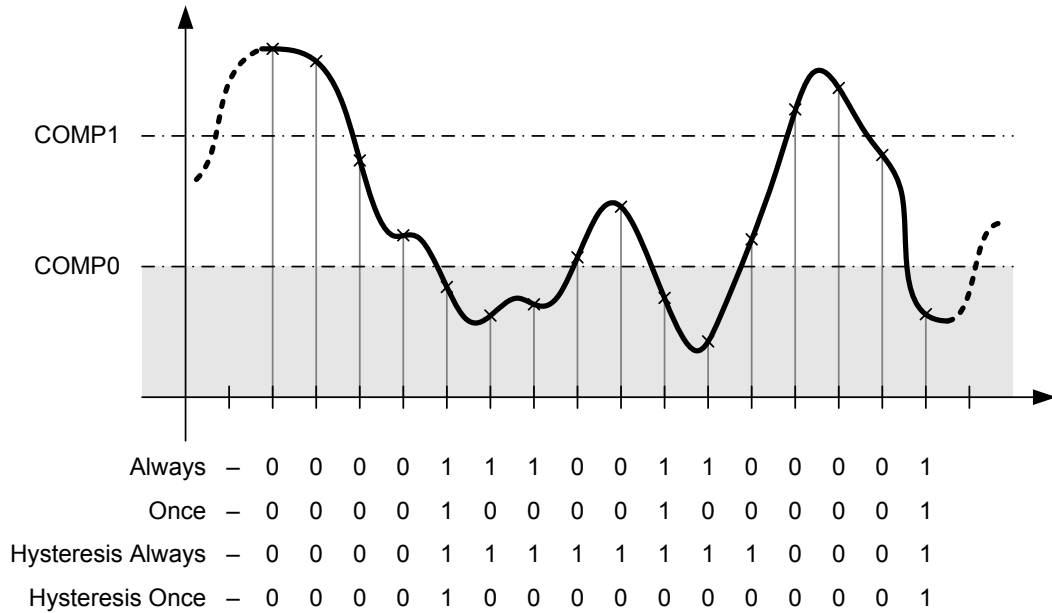
14.3.7.3 Function Ranges

The two comparison values, `COMP0` and `COMP1`, in the **ADC Digital Comparator Range (ADCDCMPn)** register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than or equal to `COMP0`), mid-band (greater than `COMP0` but less than or equal to `COMP1`), and high-band (greater than `COMP1`) regions. `COMP0` and `COMP1` may be programmed to the same value, effectively creating two regions, but `COMP1` must always be greater than or equal to the value of `COMP0`. A `COMP1` value that is less than `COMP0` generates unpredictable results.

Low-Band Operation

To operate in the low-band region, either the `CIC` field or the `CTC` field in the **ADCDCCTLn** register must be programmed to `0x0`. This setting causes interrupts or triggers to be generated in the low-band region as defined by the programmed operational mode. An example of the state of the interrupt/trigger signal in the low-band region for each of the operational modes is shown in Figure 14-9 on page 514. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

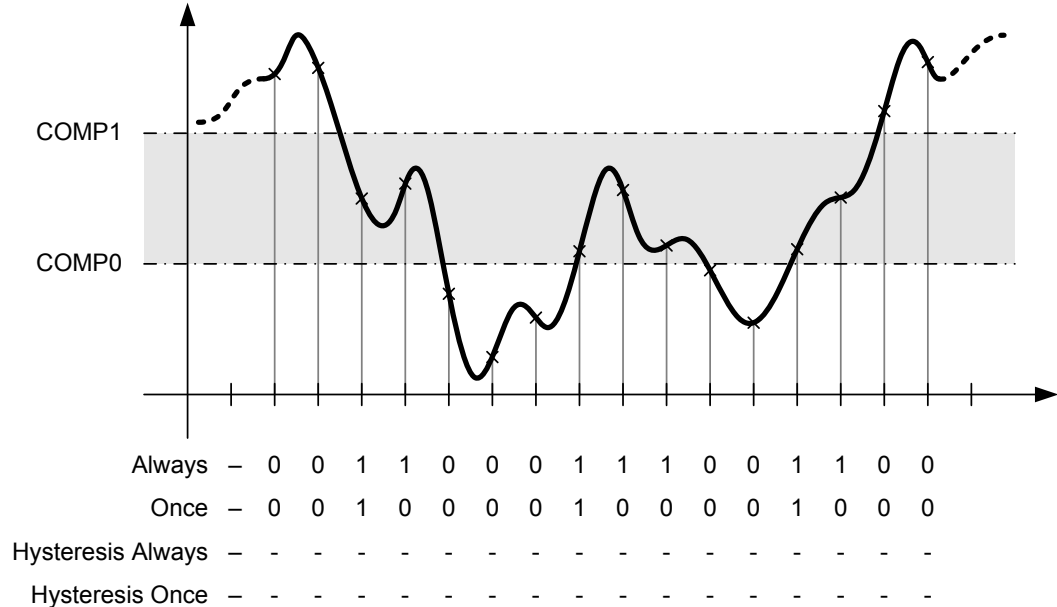
Figure 14-9. Low-Band Operation (CIC=0x0 and/or CTC=0x0)



Mid-Band Operation

To operate in the mid-band region, either the `CIC` field or the `CTC` field in the `ADCDCCTLn` register must be programmed to `0x1`. This setting causes interrupts or triggers to be generated in the mid-band region according the operation mode. Only the Always and Once operational modes are available in the mid-band region. An example of the state of the interrupt/trigger signal in the mid-band region for each of the allowed operational modes is shown in Figure 14-10 on page 515. Note that a "0" in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

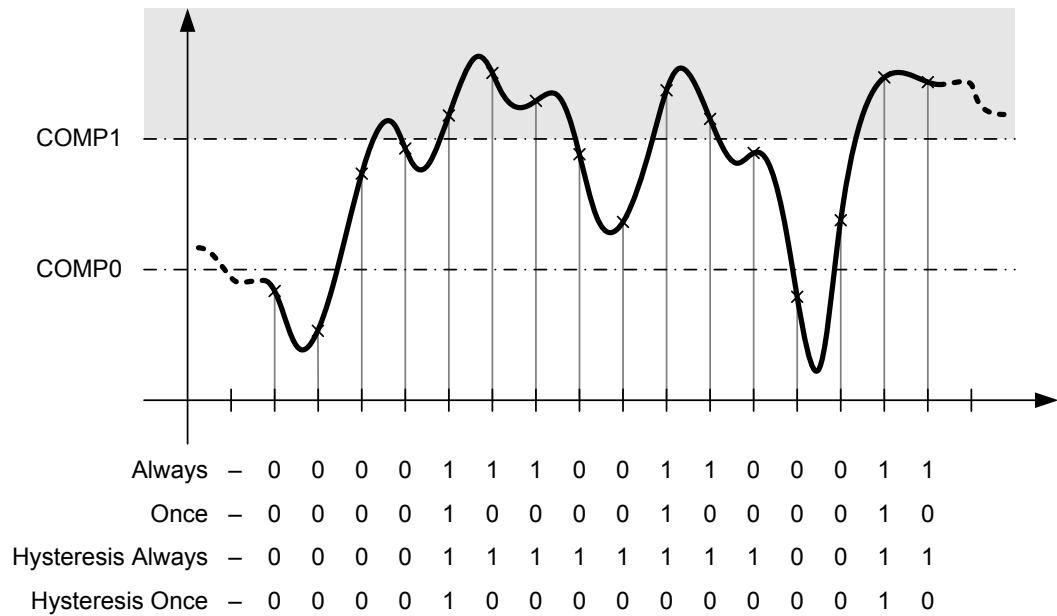
Figure 14-10. Mid-Band Operation (CIC=0x1 and/or CTC=0x1)



High-Band Operation

To operate in the high-band region, either the `CIC` field or the `CTC` field in the `ADCDCCTLn` register must be programmed to 0x3. This setting causes interrupts or triggers to be generated in the high-band region according to the operation mode. An example of the state of the interrupt/trigger signal in the high-band region for each of the allowed operational modes is shown in Figure 14-11 on page 516. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

Figure 14-11. High-Band Operation (CIC=0x3 and/or CTC=0x3)



14.4 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and programmed to a supported crystal frequency in the **RCC** register (see page 121). Using unsupported frequencies can cause faulty operation in the ADC module.

14.4.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps: enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC0** register (see page 166).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 184). To find out which GPIO port to enable, refer to Table 24-5 on page 1035.
3. Set the GPIO **AFSEL** bits for the ADC input pins (see page 340). To determine which GPIOs to configure, see Table 24-4 on page 1028.
4. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the **AINx** and **VREFA** signals to the appropriate pins (see page 358 and Table 24-5 on page 1035).
5. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** register (see page 356) in the associated GPIO block.

- 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.

14.4.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization because each sample sequencer is completely programmable.

The configuration for each sample sequencer should be as follows:

- Ensure that the sample sequencer is disabled by clearing the corresponding **ASEN_n** 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.
- Configure the trigger event for the sample sequencer in the **ADCEMUX** register.
- For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUX_n** register.
- For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTL_n** register. When programming the last nibble, ensure that the **END** bit is set. Failure to set the **END** bit causes unpredictable behavior.
- If interrupts are to be used, set the corresponding **MASK** bit in the **ADCIM** register.
- Enable the sample sequencer logic by setting the corresponding **ASEN_n** bit in the **ADCACTSS** register.

14.5 Register Map

Table 14-4 on page 517 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to that ADC module's base address of:

- ADC0: 0x4003.8000
- ADC1: 0x4003.9000

Note that the ADC module clock must be enabled before the registers can be programmed (see page 166).

Table 14-4. ADC Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|----------|-------|-------------|--------------------------------|----------|
| 0x000 | ADCACTSS | R/W | 0x0000.0000 | ADC Active Sample Sequencer | 520 |
| 0x004 | ADCRIS | RO | 0x0000.0000 | ADC Raw Interrupt Status | 521 |
| 0x008 | ADCIM | R/W | 0x0000.0000 | ADC Interrupt Mask | 523 |
| 0x00C | ADCISC | R/W1C | 0x0000.0000 | ADC Interrupt Status and Clear | 525 |
| 0x010 | ADCOSTAT | R/W1C | 0x0000.0000 | ADC Overflow Status | 528 |
| 0x014 | ADCEMUX | R/W | 0x0000.0000 | ADC Event Multiplexer Select | 530 |

Table 14-4. ADC Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-------------|-------|-------------|---|----------|
| 0x018 | ADCUSTAT | R/W1C | 0x0000.0000 | ADC Underflow Status | 532 |
| 0x020 | ADCSSPRI | R/W | 0x0000.3210 | ADC Sample Sequencer Priority | 533 |
| 0x028 | ADCPSSI | WO | - | ADC Processor Sample Sequence Initiate | 535 |
| 0x030 | ADCSAC | R/W | 0x0000.0000 | ADC Sample Averaging Control | 537 |
| 0x034 | ADCDCISC | R/W1C | 0x0000.0000 | ADC Digital Comparator Interrupt Status and Clear | 538 |
| 0x038 | ADCCTL | R/W | 0x0000.0000 | ADC Control | 540 |
| 0x040 | ADCSSMUX0 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 0 | 541 |
| 0x044 | ADCSSCTL0 | R/W | 0x0000.0000 | ADC Sample Sequence Control 0 | 543 |
| 0x048 | ADCSSFIFO0 | RO | 0x0000.0000 | ADC Sample Sequence Result FIFO 0 | 546 |
| 0x04C | ADCSSFSTAT0 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 0 Status | 547 |
| 0x050 | ADCSSOP0 | R/W | 0x0000.0000 | ADC Sample Sequence 0 Operation | 549 |
| 0x054 | ADCSSDC0 | R/W | 0x0000.0000 | ADC Sample Sequence 0 Digital Comparator Select | 551 |
| 0x060 | ADCSSMUX1 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 1 | 553 |
| 0x064 | ADCSSCTL1 | R/W | 0x0000.0000 | ADC Sample Sequence Control 1 | 554 |
| 0x068 | ADCSSFIFO1 | RO | 0x0000.0000 | ADC Sample Sequence Result FIFO 1 | 546 |
| 0x06C | ADCSSFSTAT1 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 1 Status | 547 |
| 0x070 | ADCSSOP1 | R/W | 0x0000.0000 | ADC Sample Sequence 1 Operation | 556 |
| 0x074 | ADCSSDC1 | R/W | 0x0000.0000 | ADC Sample Sequence 1 Digital Comparator Select | 557 |
| 0x080 | ADCSSMUX2 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 2 | 553 |
| 0x084 | ADCSSCTL2 | R/W | 0x0000.0000 | ADC Sample Sequence Control 2 | 554 |
| 0x088 | ADCSSFIFO2 | RO | 0x0000.0000 | ADC Sample Sequence Result FIFO 2 | 546 |
| 0x08C | ADCSSFSTAT2 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 2 Status | 547 |
| 0x090 | ADCSSOP2 | R/W | 0x0000.0000 | ADC Sample Sequence 2 Operation | 556 |
| 0x094 | ADCSSDC2 | R/W | 0x0000.0000 | ADC Sample Sequence 2 Digital Comparator Select | 557 |
| 0x0A0 | ADCSSMUX3 | R/W | 0x0000.0000 | ADC Sample Sequence Input Multiplexer Select 3 | 559 |
| 0x0A4 | ADCSSCTL3 | R/W | 0x0000.0002 | ADC Sample Sequence Control 3 | 560 |
| 0x0A8 | ADCSSFIFO3 | RO | 0x0000.0000 | ADC Sample Sequence Result FIFO 3 | 546 |
| 0x0AC | ADCSSFSTAT3 | RO | 0x0000.0100 | ADC Sample Sequence FIFO 3 Status | 547 |
| 0x0B0 | ADCSSOP3 | R/W | 0x0000.0000 | ADC Sample Sequence 3 Operation | 561 |
| 0x0B4 | ADCSSDC3 | R/W | 0x0000.0000 | ADC Sample Sequence 3 Digital Comparator Select | 562 |
| 0xD00 | ADCDCRIC | R/W | 0x0000.0000 | ADC Digital Comparator Reset Initial Conditions | 563 |
| 0xE00 | ADCDCCTL0 | R/W | 0x0000.0000 | ADC Digital Comparator Control 0 | 568 |

Table 14-4. ADC Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------|------|-------------|----------------------------------|----------|
| 0xE04 | ADCDCCTL1 | R/W | 0x0000.0000 | ADC Digital Comparator Control 1 | 568 |
| 0xE08 | ADCDCCTL2 | R/W | 0x0000.0000 | ADC Digital Comparator Control 2 | 568 |
| 0xE0C | ADCDCCTL3 | R/W | 0x0000.0000 | ADC Digital Comparator Control 3 | 568 |
| 0xE10 | ADCDCCTL4 | R/W | 0x0000.0000 | ADC Digital Comparator Control 4 | 568 |
| 0xE14 | ADCDCCTL5 | R/W | 0x0000.0000 | ADC Digital Comparator Control 5 | 568 |
| 0xE18 | ADCDCCTL6 | R/W | 0x0000.0000 | ADC Digital Comparator Control 6 | 568 |
| 0xE1C | ADCDCCTL7 | R/W | 0x0000.0000 | ADC Digital Comparator Control 7 | 568 |
| 0xE40 | ADCDCCMP0 | R/W | 0x0000.0000 | ADC Digital Comparator Range 0 | 572 |
| 0xE44 | ADCDCCMP1 | R/W | 0x0000.0000 | ADC Digital Comparator Range 1 | 572 |
| 0xE48 | ADCDCCMP2 | R/W | 0x0000.0000 | ADC Digital Comparator Range 2 | 572 |
| 0xE4C | ADCDCCMP3 | R/W | 0x0000.0000 | ADC Digital Comparator Range 3 | 572 |
| 0xE50 | ADCDCCMP4 | R/W | 0x0000.0000 | ADC Digital Comparator Range 4 | 572 |
| 0xE54 | ADCDCCMP5 | R/W | 0x0000.0000 | ADC Digital Comparator Range 5 | 572 |
| 0xE58 | ADCDCCMP6 | R/W | 0x0000.0000 | ADC Digital Comparator Range 6 | 572 |
| 0xE5C | ADCDCCMP7 | R/W | 0x0000.0000 | ADC Digital Comparator Range 7 | 572 |

14.6 Register Descriptions

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

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x000
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-------|-------|-------|-------|-----|
| | 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 | | | | | | | | | | | | ASEN3 | ASEN2 | ASEN1 | ASEN0 | |
| Type | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | ASEN3 | R/W | 0 | ADC SS3 Enable Value Description 1 Sample Sequencer 3 is enabled. 0 Sample Sequencer 3 is disabled. |
| 2 | ASEN2 | R/W | 0 | ADC SS2 Enable Value Description 1 Sample Sequencer 2 is enabled. 0 Sample Sequencer 2 is disabled. |
| 1 | ASEN1 | R/W | 0 | ADC SS1 Enable Value Description 1 Sample Sequencer 1 is enabled. 0 Sample Sequencer 1 is disabled. |
| 0 | ASEN0 | R/W | 0 | ADC SS0 Enable Value Description 1 Sample Sequencer 0 is enabled. 0 Sample Sequencer 0 is disabled. |

Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without sending the interrupts to the interrupt controller.

ADC Raw Interrupt Status (ADCRIS)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x004
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|------|------|------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | INRDC |
| 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 | | | | | | | | | | | | INR3 | INR2 | INR1 | INR0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:17 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 16 | INRDC | RO | 0 | Digital Comparator Raw Interrupt Status Value Description 1 At least one bit in the ADCDCISC register is set, meaning that a digital comparator interrupt has occurred. 0 All bits in the ADCDCISC register are clear. |
| 15:4 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | INR3 | RO | 0 | SS3 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective ADCSSCTL3 IEn bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN3 bit in the ADCISC register. |
| 2 | INR2 | RO | 0 | SS2 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective ADCSSCTL2 IEn bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN2 bit in the ADCISC register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 1 | INR1 | RO | 0 | SS1 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective ADCSSCTL1 I_{En} bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the $IN1$ bit in the ADCISC register. |
| 0 | INR0 | RO | 0 | SS0 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective ADCSSCTL0 I_{En} bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the $IN0$ bit in the ADCISC register. |

Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently. Only a single `DCONSSn` bit should be set at any given time. Setting more than one of these bits results in the `INRDC` bit from the `ADCRIS` register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines.

ADC Interrupt Mask (ADCIM)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x008
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|---------|---------|---------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | DCONSS3 | DCONSS2 | DCONSS1 | DCONSS0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | MASK3 | MASK2 | MASK1 | MASK0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:20 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 19 | DCONSS3 | R/W | 0 | Digital Comparator Interrupt on SS3 Value Description 1 The raw interrupt signal from the digital comparators (<code>INRDC</code> bit in the <code>ADCRIS</code> register) is sent to the interrupt controller on the SS3 interrupt line. 0 The status of the digital comparators does not affect the SS3 interrupt status. |
| 18 | DCONSS2 | R/W | 0 | Digital Comparator Interrupt on SS2 Value Description 1 The raw interrupt signal from the digital comparators (<code>INRDC</code> bit in the <code>ADCRIS</code> register) is sent to the interrupt controller on the SS2 interrupt line. 0 The status of the digital comparators does not affect the SS2 interrupt status. |
| 17 | DCONSS1 | R/W | 0 | Digital Comparator Interrupt on SS1 Value Description 1 The raw interrupt signal from the digital comparators (<code>INRDC</code> bit in the <code>ADCRIS</code> register) is sent to the interrupt controller on the SS1 interrupt line. 0 The status of the digital comparators does not affect the SS1 interrupt status. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 16 | DCONSS0 | R/W | 0 | Digital Comparator Interrupt on SS0 Value Description 1 The raw interrupt signal from the digital comparators (<i>INRDC</i> bit in the ADCRIS register) is sent to the interrupt controller on the SS0 interrupt line. 0 The status of the digital comparators does not affect the SS0 interrupt status. |
| 15:4 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | MASK3 | R/W | 0 | SS3 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 3 (ADCRIS register <i>INR3</i> bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 3 does not affect the SS3 interrupt status. |
| 2 | MASK2 | R/W | 0 | SS2 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 2 (ADCRIS register <i>INR2</i> bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 2 does not affect the SS2 interrupt status. |
| 1 | MASK1 | R/W | 0 | SS1 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 1 (ADCRIS register <i>INR1</i> bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 1 does not affect the SS1 interrupt status. |
| 0 | MASK0 | R/W | 0 | SS0 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 0 (ADCRIS register <i>INR0</i> bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 0 does not affect the SS0 interrupt status. |

Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective **INR** and **MASK** bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the **ADCDCISC** register. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence **INR_n** bits are still cleared via the **ADCISC** register, even if the **IN_n** bit is not set.

ADC Interrupt Status and Clear (ADCISC)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x00C
 Type R/W1C, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|---------|---------|---------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | DCINSS3 | DCINSS2 | DCINSS1 | DCINSS0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | IN3 | IN2 | IN1 | IN0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:20 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 19 | DCINSS3 | RO | 0 | Digital Comparator Interrupt Status on SS3 Value Description 1 Both the INRDC bit in the ADCRIS register and the DCONSS3 bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register. |
| 18 | DCINSS2 | RO | 0 | Digital Comparator Interrupt Status on SS2 Value Description 1 Both the INRDC bit in the ADCRIS register and the DCONSS2 bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|-------|--|
| 17 | DCINSS1 | RO | 0 | <p>Digital Comparator Interrupt Status on SS1</p> <p>Value Description</p> <p>1 Both the <code>INRDC</code> bit in the ADCRIS register and the <code>DCONSS1</code> bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1 to it. Clearing this bit also clears the <code>INRDC</code> bit in the ADCRIS register.</p> |
| 16 | DCINSS0 | RO | 0 | <p>Digital Comparator Interrupt Status on SS0</p> <p>Value Description</p> <p>1 Both the <code>INRDC</code> bit in the ADCRIS register and the <code>DCONSS0</code> bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1 to it. Clearing this bit also clears the <code>INRDC</code> bit in the ADCRIS register.</p> |
| 15:4 | reserved | RO | 0x000 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 3 | IN3 | R/W1C | 0 | <p>SS3 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Both the <code>INR3</code> bit in the ADCRIS register and the <code>MASK3</code> bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR3</code> bit in the ADCRIS register.</p> |
| 2 | IN2 | R/W1C | 0 | <p>SS2 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Both the <code>INR2</code> bit in the ADCRIS register and the <code>MASK2</code> bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR2</code> bit in the ADCRIS register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|-------|-------|---|
| 1 | IN1 | R/W1C | 0 | <p>SS1 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Both the <code>INR1</code> bit in the ADCRIS register and the <code>MASK1</code> bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR1</code> bit in the ADCRIS register.</p> |
| 0 | IN0 | R/W1C | 0 | <p>SS0 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Both the <code>INR0</code> bit in the ADCRIS register and the <code>MASK0</code> bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR0</code> bit in the ADCRIS register.</p> |

Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADC Overflow Status (ADCOSTAT)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x010
 Type R/W1C, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-----|-------|-------|-------|-------|
| | 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 | | | | | | | | | | | | OV3 | OV2 | OV1 | OV0 | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|--|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | OV3 | R/W1C | 0 | SS3 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1. |
| 2 | OV2 | R/W1C | 0 | SS2 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1. |
| 1 | OV1 | R/W1C | 0 | SS1 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1. |

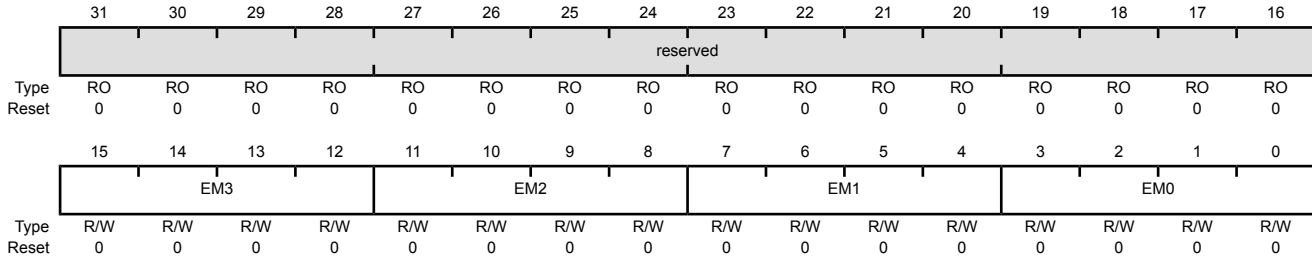
| Bit/Field | Name | Type | Reset | Description |
|-----------|------|-------|-------|---|
| 0 | OV0 | R/W1C | 0 | SS0 FIFO Overflow |
| | | | | Value Description |
| | | | | 1 The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. |
| | | | | 0 The FIFO has not overflowed. |
| | | | | This bit is cleared by writing a 1. |

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x014
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|------------------------------|------|--------|---|-------|-------|-----|---------------------|-----|---------------------|-----|---------------------|-----|---------------------|-----|---------------------|-----|-------|-----|----------|-----|----------|-----|----------|-----|----------|---------|----------|-----|------------------------------|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15:12 | EM3 | R/W | 0x0 | SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Event</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>Processor (default)</td></tr> <tr><td>0x1</td><td>Analog Comparator 0</td></tr> <tr><td>0x2</td><td>Analog Comparator 1</td></tr> <tr><td>0x3</td><td>Analog Comparator 2</td></tr> <tr><td>0x4</td><td>External (GPIO PB4)</td></tr> <tr><td>0x5</td><td>Timer</td></tr> <tr><td>0x6</td><td>reserved</td></tr> <tr><td>0x7</td><td>reserved</td></tr> <tr><td>0x8</td><td>reserved</td></tr> <tr><td>0x9</td><td>reserved</td></tr> <tr><td>0xA-0xE</td><td>reserved</td></tr> <tr><td>0xF</td><td>Always (continuously sample)</td></tr> </tbody> </table> <p>Note: PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.</p> In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register (see page 453). | Value | Event | 0x0 | Processor (default) | 0x1 | Analog Comparator 0 | 0x2 | Analog Comparator 1 | 0x3 | Analog Comparator 2 | 0x4 | External (GPIO PB4) | 0x5 | Timer | 0x6 | reserved | 0x7 | reserved | 0x8 | reserved | 0x9 | reserved | 0xA-0xE | reserved | 0xF | Always (continuously sample) |
| Value | Event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | Processor (default) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | Analog Comparator 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | Analog Comparator 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | Analog Comparator 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | External (GPIO PB4) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | Timer | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA-0xE | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF | Always (continuously sample) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11:8 | EM2 | R/W | 0x0 | SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations are the same as those for the EM3 field. | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 7:4 | EM1 | R/W | 0x0 | SS1 Trigger Select This field selects the trigger source for Sample Sequencer 1. The valid configurations are the same as those for the EM3 field. |
| 3:0 | EM0 | R/W | 0x0 | SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0. The valid configurations are the same as those for the EM3 field. |

Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

ADC Underflow Status (ADCUSTAT)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x018
 Type R/W1C, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-----|-------|-------|-------|-------|
| | 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 | | | | | | | | | | | | UV3 | UV2 | UV1 | UV0 | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|---|-------|------------|--|-------|-------------|---|---|---|-------------------------------|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | |
| 3 | UV3 | R/W1C | 0 | <p>SS3 FIFO Underflow</p> <p>The valid configurations for this field are shown below. This bit is cleared by writing a 1.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned.</td> </tr> <tr> <td>0</td> <td>The FIFO has not underflowed.</td> </tr> </tbody> </table> | Value | Description | 1 | The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. | 0 | The FIFO has not underflowed. |
| Value | Description | | | | | | | | | |
| 1 | The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. | | | | | | | | | |
| 0 | The FIFO has not underflowed. | | | | | | | | | |
| 2 | UV2 | R/W1C | 0 | <p>SS2 FIFO Underflow</p> <p>The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.</p> | | | | | | |
| 1 | UV1 | R/W1C | 0 | <p>SS1 FIFO Underflow</p> <p>The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.</p> | | | | | | |
| 0 | UV0 | R/W1C | 0 | <p>SS0 FIFO Underflow</p> <p>The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.</p> | | | | | | |

Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADC Sample Sequencer Priority (ADCSSPRI)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x020
 Type R/W, reset 0x0000.3210

| | | | | | | | | | | | | | | | | |
|-------|----------|----|-----|-----|----------|----|-----|-----|----------|----|-----|-----|----------|----|-----|-----|
| | 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 | | reserved | | SS2 | | reserved | | SS1 | | reserved | | SS0 | |
| Type | RO | RO | R/W | R/W | RO | RO | R/W | R/W | RO | RO | R/W | R/W | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| 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 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |
| 11:10 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9:8 | SS2 | R/W | 0x2 | SS2 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |
| 7:6 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5:4 | SS1 | R/W | 0x1 | SS1 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |
| 3:2 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 1:0 | SS0 | R/W | 0x0 | SS0 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal. |

Register 9: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

This register also provides a means to configure and then initiate concurrent sampling on all ADC modules. To do this, the first ADC module should be configured. The **ADCPSSI** register for that module should then be written. The appropriate **SS** bits should be set along with the **SYNCWAIT** bit. Additional ADC modules should then be configured following the same procedure. Once the final ADC module is configured, its **ADCPSSI** register should be written with the appropriate **SS** bits set along with the **GSYNC** bit. All of the ADC modules then begin concurrent sampling according to their configuration.

ADC Processor Sample Sequence Initiate (ADCPSSI)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x028

Type WO, reset -

| | | | | | | | | | | | | | | | | |
|-------|----------|----------|----|----|----------|----------|----|----|----|----|----|----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | GSYNC | reserved | | | SYNCWAIT | reserved | | | | | | | | | | |
| Type | R/W | WO | WO | WO | R/W | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | - | - | - | 0 | - | - | - | - | - | - | - | - | - | - | - |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | SS3 | SS2 | SS1 | SS0 |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31 | GSYNC | R/W | 0 | Global Synchronize |
| | | | | Value Description |
| | | | | 1 This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SS_n bit and the SYNCWAIT bit starts sampling once this bit is written. |
| | | | | 0 This bit is cleared once sampling has been initiated. |
| 30:28 | reserved | WO | - | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 27 | SYNCWAIT | R/W | 0 | Synchronize Wait |
| | | | | Value Description |
| | | | | 1 This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set. |
| | | | | 0 Sampling begins when a sample sequence has been initiated. |
| 26:4 | reserved | WO | - | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 3 | SS3 | WO | - | <p>SS3 Initiate</p> <p>Value Description</p> <p>1 Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.</p> <p>0 No effect.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p> |
| 2 | SS2 | WO | - | <p>SS2 Initiate</p> <p>Value Description</p> <p>1 Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.</p> <p>0 No effect.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p> |
| 1 | SS1 | WO | - | <p>SS1 Initiate</p> <p>Value Description</p> <p>1 Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.</p> <p>0 No effect.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p> |
| 0 | SS0 | WO | - | <p>SS0 Initiate</p> <p>Value Description</p> <p>1 Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.</p> <p>0 No effect.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p> |

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)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x030
 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 | | | | | | | | | | | | | AVG | | | |
| Type | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 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 Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034

This register provides status and acknowledgement of digital comparator interrupts. One bit is provided for each comparator.

ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x034
 Type R/W1C, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|--------|--------|--------|--------|--------|--------|--------|--------|
| | 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 | | | | | | | | DCINT7 | DCINT6 | DCINT5 | DCINT4 | DCINT3 | DCINT2 | DCINT1 | DCINT0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7 | DCINT7 | R/W1C | 0 | Digital Comparator 7 Interrupt Status and Clear Value Description 1 Digital Comparator 7 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1. |
| 6 | DCINT6 | R/W1C | 0 | Digital Comparator 6 Interrupt Status and Clear Value Description 1 Digital Comparator 6 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1. |
| 5 | DCINT5 | R/W1C | 0 | Digital Comparator 5 Interrupt Status and Clear Value Description 1 Digital Comparator 5 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|-------|-------|---|
| 4 | DCINT4 | R/W1C | 0 | <p>Digital Comparator 4 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Digital Comparator 4 has generated an interrupt.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1.</p> |
| 3 | DCINT3 | R/W1C | 0 | <p>Digital Comparator 3 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Digital Comparator 3 has generated an interrupt.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1.</p> |
| 2 | DCINT2 | R/W1C | 0 | <p>Digital Comparator 2 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Digital Comparator 2 has generated an interrupt.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1.</p> |
| 1 | DCINT1 | R/W1C | 0 | <p>Digital Comparator 1 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Digital Comparator 1 has generated an interrupt.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1.</p> |
| 0 | DCINT0 | R/W1C | 0 | <p>Digital Comparator 0 Interrupt Status and Clear</p> <p>Value Description</p> <p>1 Digital Comparator 0 has generated an interrupt.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1.</p> |

Register 12: ADC Control (ADCCTL), offset 0x038

This register selects the voltage reference.

ADC Control (ADCCTL)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

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 | | | | | | | | | | | | | | | VREF |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

Value Description

- 1 The external VREFA input is the voltage reference.
- 0 The internal reference as the voltage reference.

Register 13: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x040
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|------|-----|-----|-----|------|-----|-----|-----|------|-----|-----|-----|------|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | MUX7 | | | | MUX6 | | | | MUX5 | | | | MUX4 | | | |
| 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 |
| | MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| 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 |
|-----------|------|------|-------|--|
| 31:28 | 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 0x1 indicates the input is AIN1. |
| 27:24 | 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. |
| 23:20 | 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. |
| 19:16 | 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. |
| 15:12 | 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. |
| 11:8 | MUX2 | R/W | 0x0 | 3rd Sample Input Select The MUX2 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 7:4 | MUX1 | R/W | 0x0 | 2nd Sample Input Select The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. |
| 3:0 | MUX0 | R/W | 0x0 | 1st Sample Input Select 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 14: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the `END` bit must be set for the final sample, whether it be after the first sample, eighth sample, or any sample in between. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Control 0 (ADCSSCTL0)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x044
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|-----|-----|------|-----|-----|-----|------|-----|-----|-----|------|-----|-----|-----|------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | TS7 | IE7 | END7 | D7 | TS6 | IE6 | END6 | D6 | TS5 | IE5 | END5 | D5 | TS4 | IE4 | END4 | D4 |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TS3 | IE3 | END3 | D3 | TS2 | IE2 | END2 | D2 | TS1 | IE1 | END1 | D1 | TS0 | IE0 | END0 | D0 |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 31 | TS7 | R/W | 0 | 8th Sample Temp Sensor Select |
| | | | | Value Description |
| | | | | 1 The temperature sensor is read during the eighth sample of the sample sequence. |
| | | | | 0 The input pin specified by the ADCSSMUXn register is read during the eighth sample of the sample sequence. |
| 30 | IE7 | R/W | 0 | 8th Sample Interrupt Enable |
| | | | | Value Description |
| | | | | 1 The raw interrupt signal (<code>INR0</code> bit) is asserted at the end of the eighth sample's conversion. If the <code>MASK0</code> bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller. |
| | | | | 0 The raw interrupt is not asserted to the interrupt controller. |
| | | | | It is legal to have multiple samples within a sequence generate interrupts. |
| 29 | END7 | R/W | 0 | 8th Sample is End of Sequence |
| | | | | Value Description |
| | | | | 1 The eighth sample is the last sample of the sequence. |
| | | | | 0 Another sample in the sequence is the final sample. |
| | | | | It is possible to end the sequence on any sample position. Software must set an <code>ENDn</code> bit somewhere within the sequence. Samples defined after the sample containing a set <code>ENDn</code> bit are not requested for conversion even though the fields may be non-zero. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 28 | D7 | R/W | 0 | 8th Sample Diff Input Select Value Description 1 The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". 0 The analog inputs are not differentially sampled. Because the temperature sensor does not have a differential option, this bit must not be set when the TS7 bit is set. |
| 27 | TS6 | R/W | 0 | 7th Sample Temp Sensor Select Same definition as TS7 but used during the seventh sample. |
| 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 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. |
| 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 15: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

Register 16: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

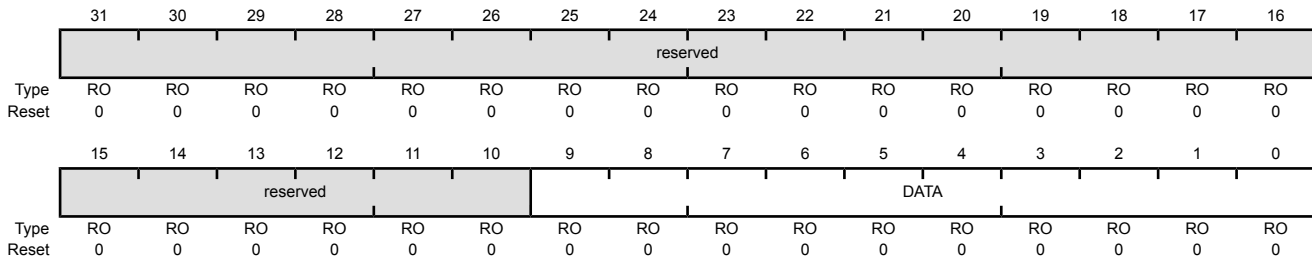
Register 17: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

Register 18: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

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)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x048
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:10 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9:0 | DATA | RO | 0x000 | Conversion Result Data |

Register 19: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C**Register 20: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C****Register 21: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C****Register 22: 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, which has 8 entries; **ADCSSFSTAT1** on FIFO1, which has 4 entries; **ADCSSFSTAT2** on FIFO2, which has 4 entries; and **ADCSSFSTAT3** on FIFO3 which has a single entry.

ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x04C
 Type RO, reset 0x0000.0100

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|------|----------|----|----|-------|------|----|----|----|------|----|----|----|
| | 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 | | | FULL | reserved | | | EMPTY | HPTR | | | | TPTR | | | |
| 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 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:13 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 12 | FULL | RO | 0 | FIFO Full Value Description 1 The FIFO is currently full. 0 The FIFO is not currently full. |
| 11:9 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | EMPTY | RO | 1 | FIFO Empty Value Description 1 The FIFO is currently empty. 0 The FIFO is not currently empty. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 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 23: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

ADC Sample Sequence 0 Operation (ADCSSOP0)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x050
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|--------|----------|----|----|--------|----------|----|----|--------|----------|----|----|--------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | S7DCOP | reserved | | | S6DCOP | reserved | | | S5DCOP | reserved | | | S4DCOP |
| Type | RO | RO | RO | R/W | RO | RO | RO | R/W | RO | RO | RO | R/W | RO | RO | RO | 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 | | | S3DCOP | reserved | | | S2DCOP | reserved | | | S1DCOP | reserved | | | S0DCOP |
| Type | RO | RO | RO | R/W | RO | RO | RO | R/W | RO | RO | RO | R/W | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:29 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 28 | S7DCOP | R/W | 0 | Sample 7 Digital Comparator Operation Value Description 1 The eighth sample is sent to the digital comparator unit specified by the <i>S7DCSEL</i> bit in the ADCSSDC0 register, and the value is not written to the FIFO. 0 The eighth sample is saved in Sample Sequence FIFO0. |
| 27:25 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 24 | S6DCOP | R/W | 0 | Sample 6 Digital Comparator Operation Same definition as <i>S7DCOP</i> but used during the seventh sample. |
| 23:21 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 20 | S5DCOP | R/W | 0 | Sample 5 Digital Comparator Operation Same definition as <i>S7DCOP</i> but used during the sixth sample. |
| 19:17 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 16 | S4DCOP | R/W | 0 | Sample 4 Digital Comparator Operation Same definition as <i>S7DCOP</i> but used during the fifth sample. |

| 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 | S3DCOP | R/W | 0 | Sample 3 Digital Comparator Operation Same definition as S7DCOP but used during the fourth sample. |
| 11:9 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | S2DCOP | R/W | 0 | Sample 2 Digital Comparator Operation Same definition as S7DCOP but used during the third sample. |
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | S1DCOP | R/W | 0 | Sample 1 Digital Comparator Operation Same definition as S7DCOP but used during the second sample. |
| 3:1 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | S0DCOP | R/W | 0 | Sample 0 Digital Comparator Operation Same definition as S7DCOP but used during the first sample. |

Register 24: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding S_nDCOP bit in the **ADCSSOP0** register is set.

ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x054
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|---------|-----|-----|-----|---------|-----|-----|-----|---------|-----|-----|-----|---------|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | S7DCSEL | | | | S6DCSEL | | | | S5DCSEL | | | | S4DCSEL | | | |
| 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 |
| | S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |
| 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 | | | | | | | | | | | | | | | | | | |
|-----------|--|------|-------|---|-------|-------------|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|
| 31:28 | S7DCSEL | R/W | 0x0 | <p>Sample 7 Digital Comparator Select</p> <p>When the $S7DCOP$ bit in the ADCSSOP0 register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer 0.</p> <p>Note: Values not listed are reserved.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0)</td> </tr> <tr> <td>0x1</td> <td>Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1)</td> </tr> <tr> <td>0x2</td> <td>Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2)</td> </tr> <tr> <td>0x3</td> <td>Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3)</td> </tr> <tr> <td>0x4</td> <td>Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4)</td> </tr> <tr> <td>0x5</td> <td>Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5)</td> </tr> <tr> <td>0x6</td> <td>Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6)</td> </tr> <tr> <td>0x7</td> <td>Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7)</td> </tr> </tbody> </table> | Value | Description | 0x0 | Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0) | 0x1 | Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1) | 0x2 | Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2) | 0x3 | Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3) | 0x4 | Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4) | 0x5 | Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5) | 0x6 | Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6) | 0x7 | Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7) |
| Value | Description | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0) | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1) | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2) | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3) | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4) | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5) | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6) | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7) | | | | | | | | | | | | | | | | | | | | | |
| 27:24 | S6DCSEL | R/W | 0x0 | <p>Sample 6 Digital Comparator Select</p> <p>This field has the same encodings as $S7DCSEL$ but is used during the seventh sample.</p> | | | | | | | | | | | | | | | | | | |
| 23:20 | S5DCSEL | R/W | 0x0 | <p>Sample 5 Digital Comparator Select</p> <p>This field has the same encodings as $S7DCSEL$ but is used during the sixth sample.</p> | | | | | | | | | | | | | | | | | | |
| 19:16 | S4DCSEL | R/W | 0x0 | <p>Sample 4 Digital Comparator Select</p> <p>This field has the same encodings as $S7DCSEL$ but is used during the fifth sample.</p> | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 15:12 | S3DCSEL | R/W | 0x0 | Sample 3 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the fourth sample. |
| 11:8 | S2DCSEL | R/W | 0x0 | Sample 2 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the third sample. |
| 7:4 | S1DCSEL | R/W | 0x0 | Sample 1 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the second sample. |
| 3:0 | S0DCSEL | R/W | 0x0 | Sample 0 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the first sample. |

Register 25: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 26: 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 541 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x060
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|------|-----|-----|-----|------|-----|-----|-----|------|-----|-----|-----|
| | 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 |
| | MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| 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. |
| 15:12 | MUX3 | R/W | 0x0 | 4th Sample Input Select |
| 11:8 | MUX2 | R/W | 0x0 | 3rd Sample Input Select |
| 7:4 | MUX1 | R/W | 0x0 | 2nd Sample Input Select |
| 3:0 | MUX0 | R/W | 0x0 | 1st Sample Input Select |

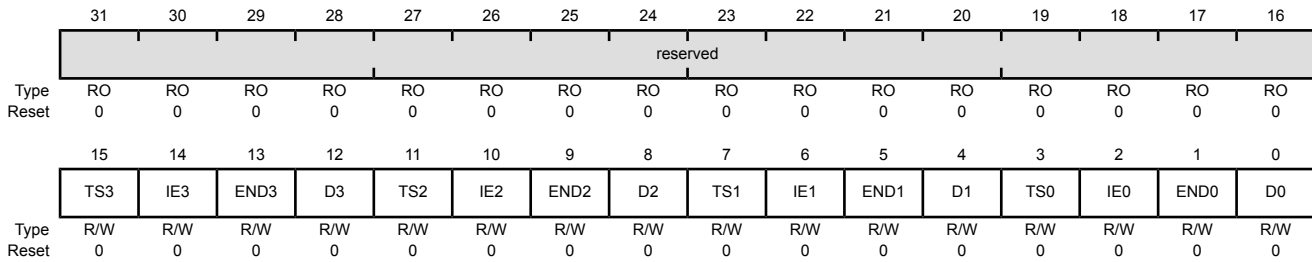
Register 27: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 28: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the **END** bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 543 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control 1 (ADCSSCTL1)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x064
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15 | TS3 | R/W | 0 | 4th Sample Temp Sensor Select 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 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 29: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070

Register 30: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090

This register determines whether the sample from the given conversion on Sample Sequence n is saved in the Sample Sequence n FIFO or sent to the digital comparator unit. The **ADCSSOP1** register controls Sample Sequencer 1 and the **ADCSSOP2** register controls Sample Sequencer 2.

ADC Sample Sequence 1 Operation (ADCSSOP1)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x070
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:13 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 12 | S3DCOP | R/W | 0 | Sample 3 Digital Comparator Operation Value Description 1 The fourth sample is sent to the digital comparator unit specified by the S3DCSEL bit in the ADCSSDC0n register, and the value is not written to the FIFO. 0 The fourth sample is saved in Sample Sequence FIFO. |
| 11:9 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | S2DCOP | R/W | 0 | Sample 2 Digital Comparator Operation Same definition as S3DCOP but used during the third sample. |
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | S1DCOP | R/W | 0 | Sample 1 Digital Comparator Operation Same definition as S3DCOP but used during the second sample. |
| 3:1 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | S0DCOP | R/W | 0 | Sample 0 Digital Comparator Operation Same definition as S3DCOP but used during the first sample. |

Register 31: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074**Register 32: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094**

These registers determine which digital comparator receives the sample from the given conversion on Sample Sequence n if the corresponding S_nDCOP bit in the **ADCSSOPn** register is set. The **ADCSSDC1** register controls the selection for Sample Sequencer 1 and the **ADCSSDC2** register controls the selection for Sample Sequencer 2.

ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x074
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|---------|-----|-----|-----|---------|-----|-----|-----|---------|-----|-----|-----|
| | 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 |
| | S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |
| 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. | | | | | | | | | | | | | | | | | | |
| 15:12 | S3DCSEL | R/W | 0x0 | <p>Sample 3 Digital Comparator Select</p> <p>When the $S3DCOP$ bit in the ADCSSOPn register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer n.</p> <p>Note: Values not listed are reserved.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0)</td></tr> <tr><td>0x1</td><td>Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1)</td></tr> <tr><td>0x2</td><td>Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2)</td></tr> <tr><td>0x3</td><td>Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3)</td></tr> <tr><td>0x4</td><td>Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4)</td></tr> <tr><td>0x5</td><td>Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5)</td></tr> <tr><td>0x6</td><td>Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6)</td></tr> <tr><td>0x7</td><td>Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7)</td></tr> </tbody> </table> | Value | Description | 0x0 | Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0) | 0x1 | Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1) | 0x2 | Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2) | 0x3 | Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3) | 0x4 | Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4) | 0x5 | Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5) | 0x6 | Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6) | 0x7 | Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7) |
| Value | Description | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0) | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1) | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2) | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3) | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4) | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5) | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6) | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7) | | | | | | | | | | | | | | | | | | | | | |
| 11:8 | S2DCSEL | R/W | 0x0 | <p>Sample 2 Digital Comparator Select</p> <p>This field has the same encodings as S3DCSEL but is used during the third sample.</p> | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 7:4 | S1DCSEL | R/W | 0x0 | Sample 1 Digital Comparator Select This field has the same encodings as S3DCSEL but is used during the second sample. |
| 3:0 | S0DCSEL | R/W | 0x0 | Sample 0 Digital Comparator Select This field has the same encodings as S3DCSEL but is used during the first sample. |

Register 33: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for the sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 541 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x0A0
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|------|-----|-----|-----|
| | 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 | | | | | | | | | | | | MUX0 | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3:0 | MUX0 | R/W | 0 | 1st Sample Input Select |

Register 34: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. The `END0` bit is always set as this sequencer can execute only one sample. This register is 4 bits wide and contains information for one possible sample. See the `ADCSSCTL0` register on page 543 for detailed bit descriptions.

ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x0A4
 Type R/W, reset 0x0000.0002

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-----|-----|------|-----|-----|
| | 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 | | | | | | | | | | | | TS0 | IE0 | END0 | D0 | |
| Type | RO | 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 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | TS0 | R/W | 0 | 1st Sample Temp Sensor Select Same definition as <code>TS7</code> but used during the first sample. |
| 2 | IE0 | R/W | 0 | 1st Sample Interrupt Enable Same definition as <code>IE7</code> but used during the first sample. |
| 1 | END0 | R/W | 1 | 1st Sample is End of Sequence Same definition as <code>END7</code> but used during the first sample. Because this sequencer has only one entry, this bit must be set. |
| 0 | D0 | R/W | 0 | 1st Sample Diff Input Select Same definition as <code>D7</code> but used during the first sample. |

Register 35: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0

This register determines whether the sample from the given conversion on Sample Sequence 3 is saved in the Sample Sequence 3 FIFO or sent to the digital comparator unit.

ADC Sample Sequence 3 Operation (ADCSSOP3)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x0B0

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|-----|
| | 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 | | | | | | | | | | | | | | | S0DCOP | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | S0DCOP | R/W | 0 | Sample 0 Digital Comparator Operation |
| | | | | Value Description |
| | | | | 1 The sample is sent to the digital comparator unit specified by the S0DCSEL bit in the ADCSSDC03 register, and the value is not written to the FIFO. |
| | | | | 0 The sample is saved in Sample Sequence FIFO3. |

Register 36: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 3 if the corresponding S_{nDCOP} bit in the **ADCSSOP3** register is set.

ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x0B4
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|---------|-----|-----|-----|
| | 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 | | | | | | | | | | | | S0DCSEL | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3:0 | S0DCSEL | R/W | 0x0 | Sample 0 Digital Comparator Select |

When the S_{0DCOP} bit in the **ADCSSOP3** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the sample from Sample Sequencer 3.

Note: Values not listed are reserved.

| Value | Description |
|-------|--|
| 0x0 | Digital Comparator Unit 0 (ADCDCOMP0 and ADCCCTL0) |
| 0x1 | Digital Comparator Unit 1 (ADCDCOMP1 and ADCCCTL1) |
| 0x2 | Digital Comparator Unit 2 (ADCDCOMP2 and ADCCCTL2) |
| 0x3 | Digital Comparator Unit 3 (ADCDCOMP3 and ADCCCTL3) |
| 0x4 | Digital Comparator Unit 4 (ADCDCOMP4 and ADCCCTL4) |
| 0x5 | Digital Comparator Unit 5 (ADCDCOMP5 and ADCCCTL5) |
| 0x6 | Digital Comparator Unit 6 (ADCDCOMP6 and ADCCCTL6) |
| 0x7 | Digital Comparator Unit 7 (ADCDCOMP7 and ADCCCTL7) |

Register 37: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00

This register provides the ability to reset any of the digital comparator interrupt or trigger functions back to their initial conditions. Resetting these functions ensures that the data that is being used by the interrupt and trigger functions in the digital comparator unit is not stale.

ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0xD00
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|---------|---------|---------|---------|---------|---------|---------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | DCTRIG7 | DCTRIG6 | DCTRIG5 | DCTRIG4 | DCTRIG3 | DCTRIG2 | DCTRIG1 | DCTRIG0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | DCINT7 | DCINT6 | DCINT5 | DCINT4 | DCINT3 | DCINT2 | DCINT1 | DCINT0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31:24 | reserved | RO | 0x00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 23 | DCTRIG7 | R/W | 0 | <p>Digital Comparator Trigger 7</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 7 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 22 | DCTRIG6 | R/W | 0 | <p>Digital Comparator Trigger 6</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 6 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 21 | DCTRIG5 | R/W | 0 | <p>Digital Comparator Trigger 5</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 5 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 20 | DCTRIG4 | R/W | 0 | <p>Digital Comparator Trigger 4</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 4 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 19 | DCTRIG3 | R/W | 0 | <p>Digital Comparator Trigger 3</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 3 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 18 | DCTRIG2 | R/W | 0 | <p>Digital Comparator Trigger 2</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 2 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 17 | DCTRIG1 | R/W | 0 | <p>Digital Comparator Trigger 1</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 1 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 16 | DCTRIG0 | R/W | 0 | <p>Digital Comparator Trigger 0</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 0 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 15:8 | reserved | RO | 0x00 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 7 | DCINT7 | R/W | 0 | <p>Digital Comparator Interrupt 7</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 7 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 6 | DCINT6 | R/W | 0 | <p>Digital Comparator Interrupt 6</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 6 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 5 | DCINT5 | R/W | 0 | <p>Digital Comparator Interrupt 5</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 5 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 4 | DCINT4 | R/W | 0 | <p>Digital Comparator Interrupt 4</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 4 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 3 | DCINT3 | R/W | 0 | <p>Digital Comparator Interrupt 3</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 3 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 2 | DCINT2 | R/W | 0 | <p>Digital Comparator Interrupt 2</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 2 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 1 | DCINT1 | R/W | 0 | <p>Digital Comparator Interrupt 1</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 1 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |
| 0 | DCINT0 | R/W | 0 | <p>Digital Comparator Interrupt 0</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 0 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p> |

Register 38: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00

Register 39: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04

Register 40: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08

Register 41: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C

Register 42: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10

Register 43: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14

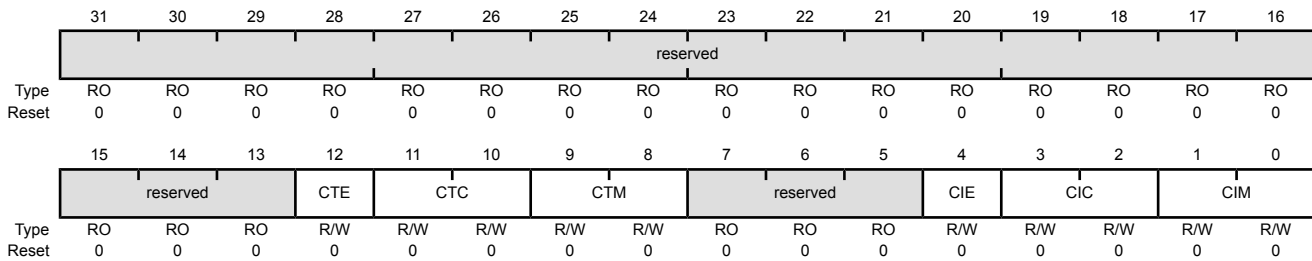
Register 44: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18

Register 45: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C

This register provides the comparison encodings that generate an interrupt or PWM trigger.

ADC Digital Comparator Control 0 (ADCDCCTL0)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0xE00
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description | |
|-------------------|----------|---|----------|---|--|
| 31:13 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | |
| 12 | CTE | R/W | 0 | Comparison Trigger Enable | |
| Value Description | | | | | |
| | 1 | Enables the trigger function state machine. The ADC conversion data is used to determine if a trigger should be generated according to the programming of the CTC and CTM fields. | | | |
| | 0 | Disables the trigger function state machine. ADC conversion data is ignored by the trigger function. | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 11:10 | CTC | R/W | 0x0 | <p>Comparison Trigger Condition</p> <p>This field specifies the operational region in which a trigger is generated when the ADC conversion data is compared against the values of COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the ADCDCMPx registers.</p> <p>Value Description</p> <p>0x0 Low Band ADC Data < COMP0 and < COMP1</p> <p>0x1 Mid Band COMP0 ≤ ADC Data < COMP1</p> <p>0x2 reserved</p> <p>0x3 High Band COMP0 ≤ COMP1 ≤ ADC Data</p> |
| 9:8 | CTM | R/W | 0x0 | <p>Comparison Trigger Mode</p> <p>This field specifies the mode by which the trigger comparison is made.</p> <p>Value Description</p> <p>0x0 Always This mode generates a trigger every time the ADC conversion data falls within the selected operational region.</p> <p>0x1 Once This mode generates a trigger the first time that the ADC conversion data enters the selected operational region.</p> <p>0x2 Hysteresis Always This mode generates a trigger when the ADC conversion data falls within the selected operational region and continues to generate the trigger until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p> <p>0x3 Hysteresis Once This mode generates a trigger the first time that the ADC conversion data falls within the selected operational region. No additional triggers are generated until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p> |
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 4 | CIE | R/W | 0 | <p>Comparison Interrupt Enable</p> <p>Value Description</p> <p>1 Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the CIE and CIM fields.</p> <p>0 Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.</p> |
| 3:2 | CIC | R/W | 0x0 | <p>Comparison Interrupt Condition</p> <p>This field specifies the operational region in which an interrupt is generated when the ADC conversion data is compared against the values of COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the ADCDCCMPx registers.</p> <p>Value Description</p> <p>0x0 Low Band ADC Data < COMP0 and < COMP1</p> <p>0x1 Mid Band COMP0 ≤ ADC Data < COMP1</p> <p>0x2 reserved</p> <p>0x3 High Band COMP0 < COMP1 ≤ ADC Data</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 1:0 | CIM | R/W | 0x0 | <p>Comparison Interrupt Mode</p> <p>This field specifies the mode by which the interrupt comparison is made.</p> <p>Value Description</p> <p>0x0 Always</p> <p>This mode generates an interrupt every time the ADC conversion data falls within the selected operational region.</p> <p>0x1 Once</p> <p>This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region.</p> <p>0x2 Hysteresis Always</p> <p>This mode generates an interrupt when the ADC conversion data falls within the selected operational region and continues to generate the interrupt until the hysteresis condition is cleared by entering the opposite operational region.</p> <p>Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p> <p>0x3 Hysteresis Once</p> <p>This mode generates an interrupt the first time that the ADC conversion data falls within the selected operational region. No additional interrupts are generated until the hysteresis condition is cleared by entering the opposite operational region.</p> <p>Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p> |

Register 46: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40

Register 47: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44

Register 48: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48

Register 49: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C

Register 50: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50

Register 51: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54

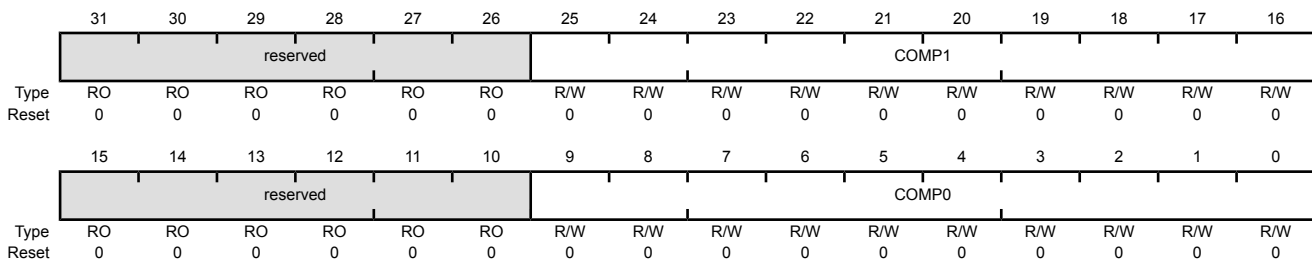
Register 52: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58

Register 53: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C

This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region. Note that the value in the COMP1 field must be greater than or equal to the value in the COMP0 field or unexpected results can occur.

ADC Digital Comparator Range 0 (ADCDCCMP0)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0xE40
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31:26 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 25:16 | COMP1 | R/W | 0x000 | Compare 1 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the high-band region. Note that the value of COMP1 must be greater than or equal to the value of COMP0. |
| 15:10 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9:0 | COMP0 | R/W | 0x000 | Compare 0 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region. |

15 Universal Asynchronous Receivers/Transmitters (UARTs)

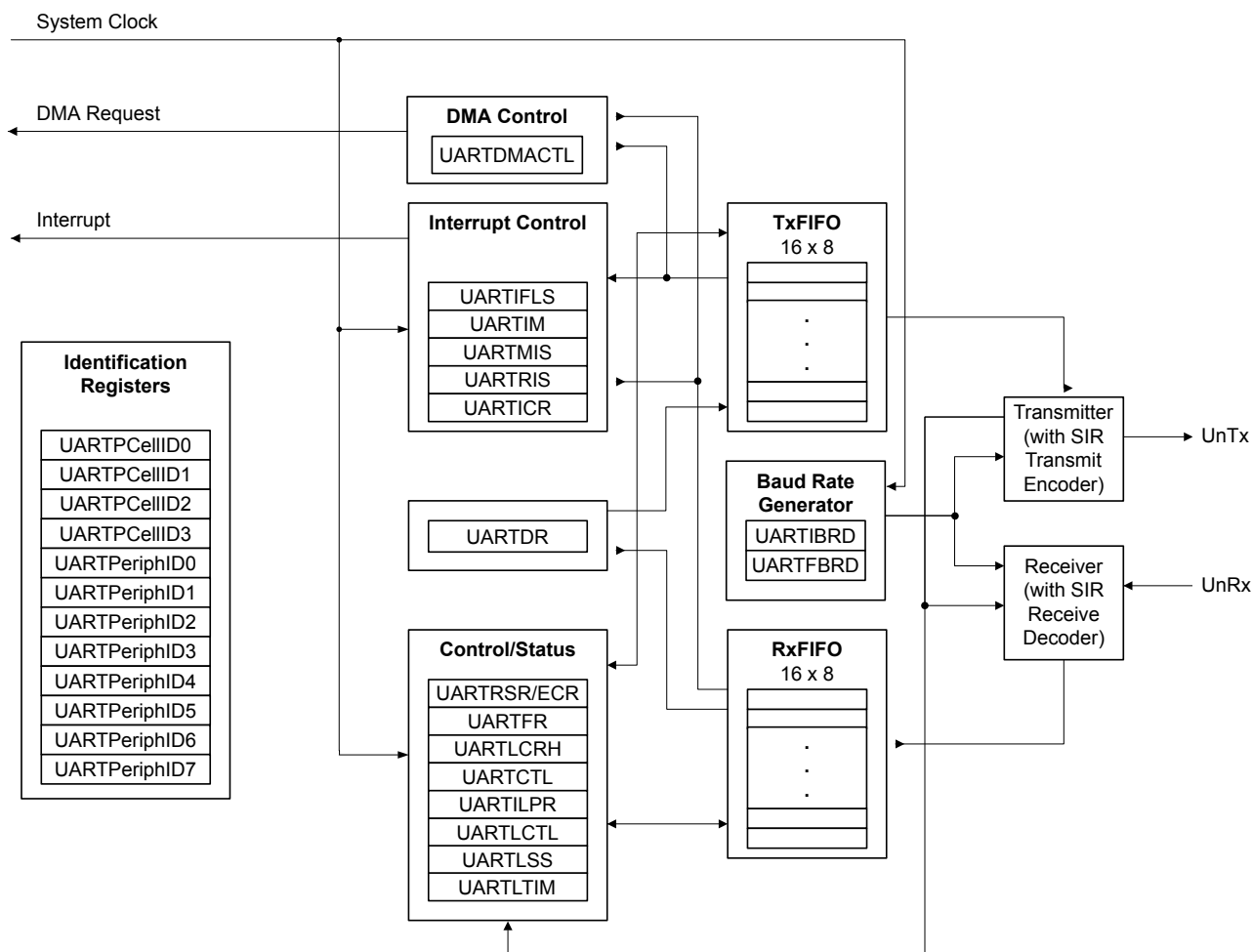
The Stellaris® LM3S9B90 controller includes three Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- False-start bit detection
- 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 μ s) bit durations
 - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- Full modem handshake support (on UART1)
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive

- Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
- Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

15.1 Block Diagram

Figure 15-1. UART Module Block Diagram



15.2 Signal Description

Table 15-1 on page 575 lists the external signals of the UART module and describes the function of each. The UART signals are alternate functions for some GPIO signals and default to be GPIO signals at reset, with the exception of the U0Rx and U0Tx pins which default to the UART function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these UART signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the UART function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOCTL)** register (page 358) to assign the UART signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318.

Table 15-1. Signals for UART

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|----------------------------------|--|----------|--------------------------|---|
| U0Rx | 26 | PA0 (1) | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| U0Tx | 27 | PA1 (1) | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U1CTS | 2 10 34 | PE6 (9) PD0 (9) PA6 (9) | I | TTL | UART module 1 Clear To Send modem status input signal. |
| U1DCD | 1 11 35 | PE7 (9) PD1 (9) PA7 (9) | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| U1DSR | 47 | PF0 (9) | I | TTL | UART module 1 Data Set Ready modem output control line. |
| U1DTR | 100 | PD7 (9) | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| U1RI | 97 | PD4 (9) | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| U1RTS | 61 | PF1 (9) | O | TTL | UART module 1 Request to Send modem output control line. |
| U1Rx | 10 12 23 26 66 92 | PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7) | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| U1Tx | 11 13 22 27 67 91 | PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7) | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U2Rx | 10 19 92 98 | PD0 (4) PG0 (1) PB4 (4) PD5 (9) | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| U2Tx | 6 11 18 99 | PE4 (5) PD1 (4) PG1 (1) PD6 (9) | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |

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

15.3 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 599). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the **UARTEN** bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

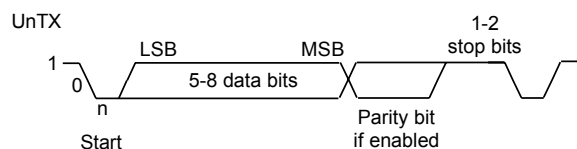
The UART module also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

15.3.1 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 15-2 on page 576 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 15-2. UART Character Frame



15.3.2 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 595) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 596). 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.)

$$BRD = BRDI + BRDF = \text{UARTSysClk} / (\text{ClkDiv} * \text{Baud Rate})$$

where *UARTSysClk* is the system clock connected to the UART, and *ClkDiv* is either 16 (if *HSE* in **UARTCTL** is clear) or 8 (if *HSE* is set).

The 6-bit fractional number (that is to be loaded into the *DIVFRAC* bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(\text{BRDF} * 64 + 0.5)$$

The UART generates an internal baud-rate reference clock at 8x or 16x the baud-rate (referred to as *Baud8* and *Baud16*, depending on the setting of the *HSE* bit (bit 5) in **UARTCTL**). This reference clock is divided by 8 or 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 597), 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

15.3.3 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 591) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The **BUSY** bit is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the $UnRx$ signal is continuously 1), and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of **Baud16** or fourth cycle of **Baud8** depending on the setting of the **HSE** bit (bit 5) in **UARTCTL** (described in “Transmit/Receive Logic” on page 576).

The start bit is valid if the $UnRx$ signal is still low on the eighth cycle of **Baud16** (**HSE** clear) or the fourth cycle of **Baud 8** (**HSE** set), otherwise a false start bit is detected and is ignored. Start bit errors can be viewed in the **UART Receive Status (UARTSR)** register (see page 588). If the start bit was valid, successive data bits are sampled on every 16th cycle of **Baud16** or 8th cycle of **Baud8** (that is, one bit period later) according to the programmed length of the data characters and value of the **HSE** bit in **UARTCTL**. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if the $UnRx$ signal is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

15.3.4 Serial IR (SIR)

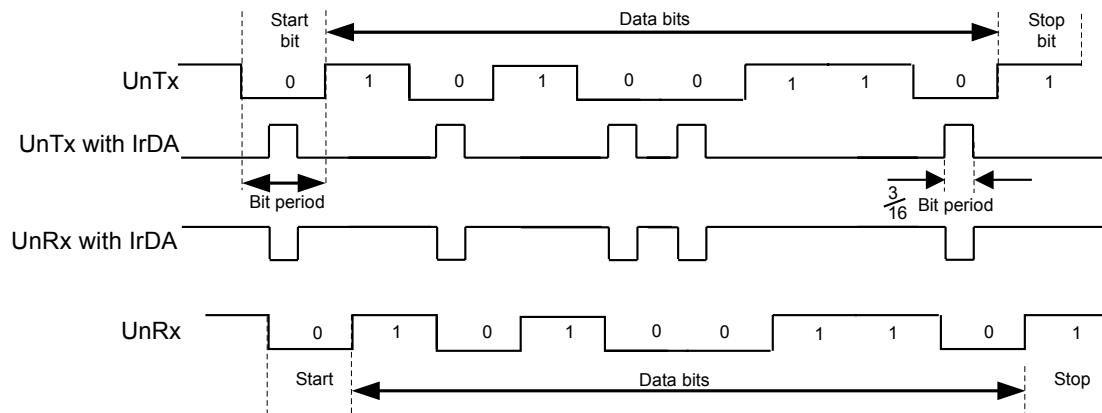
The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream and a half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output and decoded input to the UART. When enabled, the SIR block uses the $UnTx$ and $UnRx$ pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW and driving the UART input pin LOW.

- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 μ s, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register. See page 594 for more information on IrDA low-power pulse-duration configuration.

Figure 15-3 on page 578 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 15-3. 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.

15.3.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (**SMART**) of the **UARTCTL** register is set, the **UnTx** signal is used as a bit clock, and the **UnRx** signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design.

When using ISO 7816 mode, the **UARTLCRH** register must be set to transmit 8-bit words (**WLEN** bits 6:5 configured to 0x3) with EVEN parity (**PEN** set and **EPS** set). In this mode, the UART automatically uses 2 stop bits, and the **STP2** bit of the **UARTLCRH** register is ignored.

If a parity error is detected during transmission, **UnRx** is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

15.3.6 Modem Handshake Support

This section describes how to configure and use the modem status signals for UART1 when connected as a DTE (data terminal equipment) or as a DCE (data communications equipment). In general, a modem is a DCE and a computing device that connects to a modem is the DTE.

15.3.6.1 Signaling

The status signals provided by UART1 differ based on whether the UART is used as a DTE or DCE. When used as a DTE, the modem status signals are defined as:

- $\overline{\text{UICTS}}$ is Clear To Send
- $\overline{\text{UIDSR}}$ is Data Set Ready
- $\overline{\text{UIDCD}}$ is Data Carrier Detect
- $\overline{\text{UIRI}}$ is Ring Indicator
- $\overline{\text{UIRTS}}$ is Request To Send
- $\overline{\text{UIDTR}}$ is Data Terminal Ready

When used as a DCE, the the modem status signals are defined as:

- $\overline{\text{UICTS}}$ is Request To Send
- $\overline{\text{UIDSR}}$ is Data Terminal Ready
- $\overline{\text{UIRTS}}$ is Clear To Send
- $\overline{\text{UIDTR}}$ is Data Set Ready

Note that the support for DCE functions Data Carrier Detect and Ring Indicator are not provided. If these signals are required, their function can be emulated by using a general-purpose I/O signal and providing software support.

15.3.6.2 Flow Control Methods

Flow control can be accomplished by either hardware or software. The following sections describe the different methods.

Hardware Flow Control (RTS/CTS)

Hardware flow control between two devices is accomplished by connecting the $\overline{\text{UIRTS}}$ output to the Clear-To-Send input on the receiving device, and connecting the Request-To-Send output on the receiving device to the $\overline{\text{UICTS}}$ input.

The $\overline{\text{UICTS}}$ input controls the transmitter. The transmitter may only transmit data when the $\overline{\text{UICTS}}$ input is asserted. The $\overline{\text{UIRTS}}$ output signal indicates the state of the receive FIFO. $\overline{\text{UICTS}}$ remains asserted until the preprogrammed watermark level is reached, indicating that the Receive FIFO has no space to store additional characters.

The **UARTCTL** register bits 15 (CTSEN) and 14 (RTSEN) specify the flow control mode as shown in Table 15-2 on page 580.

Table 15-2. Flow Control Mode

| CTSEN | RTSEN | Description |
|-------|-------|--|
| 1 | 1 | RTS and CTS flow control enabled |
| 1 | 0 | Only CTS flow control enabled |
| 0 | 1 | Only RTS flow control enabled |
| 0 | 0 | Both RTS and CTS flow control disabled |

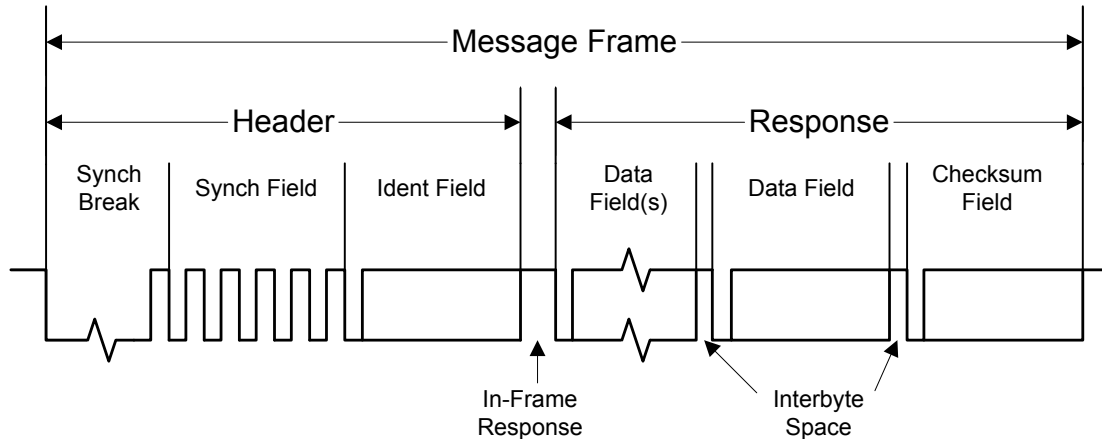
Note that when `RTSEN` is 1, software cannot modify the $\overline{\text{U}1\text{RTS}}$ output value through the `UARTCTL` register Request to Send (RTS) bit, and the status of the RTS bit should be ignored.

Software Flow Control (Modem Status Interrupts)

Software flow control between two devices is accomplished by using interrupts to indicate the status of the UART. Interrupts may be generated for $\overline{\text{U}1\text{DSR}}$, $\overline{\text{U}1\text{DCD}}$, $\overline{\text{U}1\text{CTS}}$, and $\overline{\text{U}1\text{RI}}$ using the `UARTIM` bits 3 through 0 respectively. The raw and masked interrupt status may be checked using the `UARTRIS` and `UARTMIS` register. These interrupts may be cleared using the `UARTICR` register.

15.3.7 LIN Support

The UART module offers hardware support for the LIN protocol as either a master or a slave. The LIN mode is enabled by setting the `LIN` bit in the `UARTCTL` register. A LIN message is identified by the use of a Sync Break at the beginning of the message. The Sync Break is a transmission of a series of 0s. The Sync Break is followed by the Sync data field (0x55). Figure 15-4 on page 580 illustrates the structure of a LIN message.

Figure 15-4. LIN Message


The UART should be configured as followed to operate in LIN mode:

1. Configure the UART for 1 start bit, 8 data bits, no parity, and 1 stop bit. Enable the Transmit FIFO.
2. Set the `LIN` bit in the `UARTCTL` register.

When preparing to send a LIN message, the TXFIFO should contain the Sync data (0x55) at FIFO location 0 and the Identifier data at location 1, followed by the data to be transmitted, and with the checksum in the final FIFO entry.

15.3.7.1 LIN Master

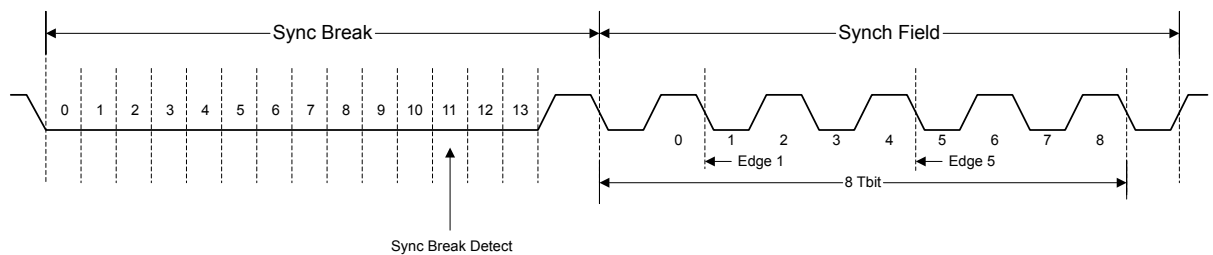
The UART is enabled to be the LIN master by setting the `MASTER` bit in the `UARTLCTL` register. The length of the Sync Break is programmable using the `BLEN` field in the `UARTLCTL` register and can be 13-16 bits (baud clock cycles).

15.3.7.2 LIN Slave

The LIN UART slave is required to adjust its baud rate to that of the LIN master. In slave mode, the LIN UART recognizes the Sync Break, which must be at least 13 bits in duration. A timer is provided to capture timing data on the 1st and 5th falling edges of the Sync field so that the baud rate can be adjusted to match the master.

After detecting a Sync Break, the UART waits for the synchronization field. The first falling edge generates an interrupt using the `LME1RIS` bit in the `UARTRIS` register, and the timer value is captured and stored in the `UARTLSS` register (`T1`). On the fifth falling edge, a second interrupt is generated using the `LME5RIS` bit in the `UARTRIS` register, and the timer value is captured again (`T2`). The actual baud rate can be calculated using $(T2-T1)/8$, and the local baud rate should be adjusted as needed. Figure 15-5 on page 581 illustrates the synchronization field.

Figure 15-5. LIN Synchronization Field



15.3.8 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 586). 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 597).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 591) 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 603). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, and $\frac{7}{8}$. For example, if the $\frac{1}{4}$ 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 $\frac{1}{2}$ mark.

15.3.9 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error
- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the `TXIFLSEL` bit in the **UARTIFLS** register is met, or if the `EOT` bit in **UARTCTRL** is set, when the last bit of all transmitted data leaves the serializer)
- Receive (when condition defined in the `RXIFLSEL` bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 613).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 605) by setting the corresponding `IM` bits. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 609).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by writing a 1 to the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 616).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

15.3.10 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the `LBE` bit in the **UARTCTL** register (see page 599). In loopback mode, data transmitted on the `UnTx` output is received on the `UnRx` input.

15.3.11 DMA Operation

The UART provides an interface to the μ DMA controller with separate channels for transmit and receive. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART asserts a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level configured in the **UARTIFLS** register. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the μ DMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, set the `RXDMAE` bit of the **DMA Control (UARTDMACTL)** register. To enable DMA operation for the transmit channel, set the `TXDMAE` bit of the **UARTDMACTL** register. The UART can also be configured to stop using DMA for the receive

channel if a receive error occurs. If the `DMAERR` bit of the `UARTDMACR` register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

If DMA is enabled, then the μ DMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the UART interrupt vector. Therefore, if interrupts are used for UART operation and DMA is enabled, the UART interrupt handler must be designed to handle the μ DMA completion interrupt.

See “Micro Direct Memory Access (μ DMA)” on page 258 for more details about programming the μ DMA controller.

15.4 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

1. The peripheral clock must be enabled by setting the `UART0`, `UART1`, or `UART2` bits in the `RCGC1` register (see page 175).
2. The clock to the appropriate GPIO module must be enabled via the `RCGC2` register in the System Control module (see page 184).
3. Set the GPIO `AFSEL` bits for the appropriate pins (see page 340). To determine which GPIOs to configure, see Table 24-4 on page 1028.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected (see page 342 and page 350).
5. Configure the `PMCn` fields in the `GPIOPCTL` register to assign the UART signals to the appropriate pins (see page 358 and Table 24-5 on page 1035).

To use the UARTs, the peripheral clock must be enabled by setting the `UART0`, `UART1`, or `UART2` bits in the `RCGC1` register (see page 175). In addition, the clock to the appropriate GPIO module must be enabled via the `RCGC2` register in the System Control module (see page 184). To find out which GPIO port to enable, refer to Table 24-5 on page 1035.

This section discusses the steps that are required to use a UART module. For this example, the UART clock is assumed to be 20 MHz, and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), because the `UARTIBRD` and `UARTFBRD` registers must be written before the `UARTLCRH` register. Using the equation described in “Baud-Rate Generation” on page 576, the BRD can be calculated:

$$\text{BRD} = 20,000,000 / (16 * 115,200) = 10.8507$$

which means that the `DIVINT` field of the **UARTIBRD** register (see page 595) should be set to 10 decimal or `0xA`. The value to be loaded into the **UARTFBRD** register (see page 596) is calculated by the equation:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(0.8507 * 64 + 0.5) = 54$$

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the `UARTEN` bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.
4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of `0x0000.0060`).
5. Optionally, configure the μ DMA channel (see “Micro Direct Memory Access (μ DMA)” on page 258) and enable the DMA option(s) in the **UARTDMACTL** register.
6. Enable the UART by setting the `UARTEN` bit in the **UARTCTL** register.

15.5 Register Map

Table 15-3 on page 584 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 175).

Note: The UART must be disabled (see the `UARTEN` bit in the **UARTCTL** register on page 599) 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 15-3. UART Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|-------------|-----------------------------------|----------|
| 0x000 | UARTDR | R/W | 0x0000.0000 | UART Data | 586 |
| 0x004 | UARTRSR/UARTECR | R/W | 0x0000.0000 | UART Receive Status/Error Clear | 588 |
| 0x018 | UARTFR | RO | 0x0000.0090 | UART Flag | 591 |
| 0x020 | UARTILPR | R/W | 0x0000.0000 | UART IrDA Low-Power Register | 594 |
| 0x024 | UARTIBRD | R/W | 0x0000.0000 | UART Integer Baud-Rate Divisor | 595 |
| 0x028 | UARTFBRD | R/W | 0x0000.0000 | UART Fractional Baud-Rate Divisor | 596 |
| 0x02C | UARTLCRH | R/W | 0x0000.0000 | UART Line Control | 597 |
| 0x030 | UARTCTL | R/W | 0x0000.0300 | UART Control | 599 |
| 0x034 | UARTIFLS | R/W | 0x0000.0012 | UART Interrupt FIFO Level Select | 603 |

Table 15-3. UART Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|---------------|------|-------------|----------------------------------|----------|
| 0x038 | UARTIM | R/W | 0x0000.0000 | UART Interrupt Mask | 605 |
| 0x03C | UARTRIS | RO | 0x0000.000F | UART Raw Interrupt Status | 609 |
| 0x040 | UARTMIS | RO | 0x0000.0000 | UART Masked Interrupt Status | 613 |
| 0x044 | UARTICR | W1C | 0x0000.0000 | UART Interrupt Clear | 616 |
| 0x048 | UARTDMACTL | R/W | 0x0000.0000 | UART DMA Control | 618 |
| 0x090 | UARTLCTL | R/W | 0x0000.0000 | UART LIN Control | 619 |
| 0x094 | UARTLSS | RO | 0x0000.0000 | UART LIN Snap Shot | 620 |
| 0x098 | UARTLTIM | RO | 0x0000.0000 | UART LIN Timer | 621 |
| 0xFD0 | UARTPeriphID4 | RO | 0x0000.0000 | UART Peripheral Identification 4 | 622 |
| 0xFD4 | UARTPeriphID5 | RO | 0x0000.0000 | UART Peripheral Identification 5 | 623 |
| 0xFD8 | UARTPeriphID6 | RO | 0x0000.0000 | UART Peripheral Identification 6 | 624 |
| 0xFDC | UARTPeriphID7 | RO | 0x0000.0000 | UART Peripheral Identification 7 | 625 |
| 0xFE0 | UARTPeriphID0 | RO | 0x0000.0060 | UART Peripheral Identification 0 | 626 |
| 0xFE4 | UARTPeriphID1 | RO | 0x0000.0000 | UART Peripheral Identification 1 | 627 |
| 0xFE8 | UARTPeriphID2 | RO | 0x0000.0018 | UART Peripheral Identification 2 | 628 |
| 0xFEC | UARTPeriphID3 | RO | 0x0000.0001 | UART Peripheral Identification 3 | 629 |
| 0xFF0 | UARTPCellID0 | RO | 0x0000.000D | UART PrimeCell Identification 0 | 630 |
| 0xFF4 | UARTPCellID1 | RO | 0x0000.00F0 | UART PrimeCell Identification 1 | 631 |
| 0xFF8 | UARTPCellID2 | RO | 0x0000.0005 | UART PrimeCell Identification 2 | 632 |
| 0xFFC | UARTPCellID3 | RO | 0x0000.00B1 | UART PrimeCell Identification 3 | 633 |

15.6 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

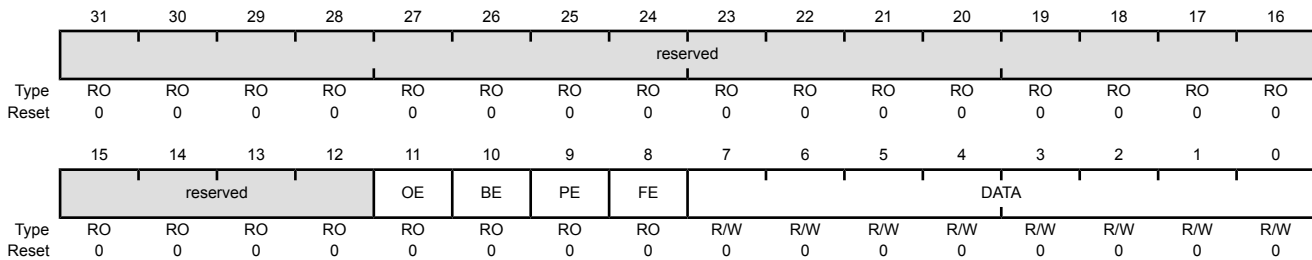
This register is the data register (the interface to the FIFOs).

For transmitted data, if the FIFO is enabled, data written to this location is pushed onto the transmit FIFO. If the FIFO is disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If the FIFO is disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x000
 Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:12 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 11 | OE | RO | 0 | UART Overrun Error |
| | | | | Value Description |
| | | | | 1 New data was received when the FIFO was full, resulting in data loss. |
| | | | | 0 No data has been lost due to a FIFO overrun. |
| 10 | BE | RO | 0 | UART Break Error |
| | | | | Value Description |
| | | | | 1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). |
| | | | | 0 No break condition has occurred |

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 | <p>UART Parity Error</p> <p>Value Description</p> <p>1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH register.</p> <p>0 No parity error has occurred</p> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p> |
| 8 | FE | RO | 0 | <p>UART Framing Error</p> <p>Value Description</p> <p>1 The received character does not have a valid stop bit (a valid stop bit is 1).</p> <p>0 No framing error has occurred</p> |
| 7:0 | DATA | R/W | 0x00 | <p>Data Transmitted or Received</p> <p>Data that is to be transmitted via the UART is written to this field.</p> <p>When read, this field contains the data that was received by the UART.</p> |

Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared on reset.

Read-Only Status Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

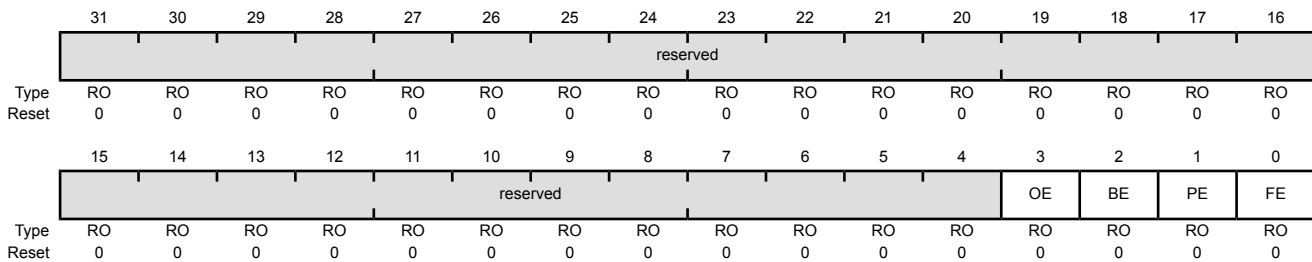
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x004

Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | OE | RO | 0 | UART Overrun Error |
| | | | | Value Description |
| | | | | 1 New data was received when the FIFO was full, resulting in data loss. |
| | | | | 0 No data has been lost due to a FIFO overrun. |
| | | | | This bit is cleared by a write to UARTECR . |
| | | | | The FIFO contents remain valid because no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must read the data in order to empty the FIFO. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 2 | BE | RO | 0 | UART Break Error Value Description 1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). 0 No break condition has occurred 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. |
| 1 | PE | RO | 0 | UART Parity Error Value Description 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH register. 0 No parity error has occurred This bit is cleared to 0 by a write to UARTECR . |
| 0 | FE | RO | 0 | UART Framing Error Value Description 1 The received character does not have a valid stop bit (a valid stop bit is 1). 0 No framing error has occurred 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. |

Write-Only Error Clear Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x004
 Type WO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | DATA | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | WO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | WO | 0x00 | Error Clear A write to this register of any data clears the framing, parity, break, and overrun flags. |

Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the **TXFF**, **RXFF**, and **BUSY** bits are 0, and **TXFE** and **RXFE** bits are 1. The **RI**, **DCD**, **DSR** and **CTS** bits indicate the modem status.

Note that bits [8,2:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Flag (UARTFR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x018
 Type RO, reset 0x0000.0090

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|------|------|------|------|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | RI | TXFE | RXFF | TXFF | RXFE | BUSY | DCD | DSR | CTS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|--|
| 31:9 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 8 | RI | RO | 0 | Ring Indicator Value Description 1 The URI signal is asserted. 0 The URI signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 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. Value Description 1 If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty. 0 The transmitter has data to transmit. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 6 | RXFF | RO | 0 | <p>UART Receive FIFO Full</p> <p>The meaning of this bit depends on the state of the <code>FEN</code> bit in the UARTLCRH register.</p> <p>Value Description</p> <p>1 If the FIFO is disabled (<code>FEN</code> is 0), the receive holding register is full.</p> <p>If the FIFO is enabled (<code>FEN</code> is 1), the receive FIFO is full.</p> <p>0 The receiver can receive data.</p> |
| 5 | TXFF | RO | 0 | <p>UART Transmit FIFO Full</p> <p>The meaning of this bit depends on the state of the <code>FEN</code> bit in the UARTLCRH register.</p> <p>Value Description</p> <p>1 If the FIFO is disabled (<code>FEN</code> is 0), the transmit holding register is full.</p> <p>If the FIFO is enabled (<code>FEN</code> is 1), the transmit FIFO is full.</p> <p>0 The transmitter is not full.</p> |
| 4 | RXFE | RO | 1 | <p>UART Receive FIFO Empty</p> <p>The meaning of this bit depends on the state of the <code>FEN</code> bit in the UARTLCRH register.</p> <p>Value Description</p> <p>1 If the FIFO is disabled (<code>FEN</code> is 0), the receive holding register is empty.</p> <p>If the FIFO is enabled (<code>FEN</code> is 1), the receive FIFO is empty.</p> <p>0 The receiver is not empty.</p> |
| 3 | BUSY | RO | 0 | <p>UART Busy</p> <p>Value Description</p> <p>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.</p> <p>0 The UART is not busy.</p> <p>This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).</p> |
| 2 | DCD | RO | 0 | <p>Data Carrier Detect</p> <p>Value Description</p> <p>1 The <code>U1DCD</code> signal is asserted.</p> <p>0 The <code>U1DCD</code> signal is not asserted.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 1 | DSR | RO | 0 | Data Set Ready Value Description 1 The U1DSR signal is asserted. 0 The U1DSR signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 0 | CTS | RO | 0 | Clear To Send Value Description 1 The U1CTS signal is asserted. 0 The U1CTS signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register stores the 8-bit low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared when reset.

The internal $F_{IrLPBaud16}$ 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 $F_{IrLPBaud16}$ clock. The low-power divisor value is calculated as follows:

$$ILPDVSR = SysClk / F_{IrLPBaud16}$$

where $F_{IrLPBaud16}$ is nominally 1.8432 MHz.

The divisor must be programmed such that $1.42 \text{ MHz} < F_{IrLPBaud16} < 2.12 \text{ MHz}$, resulting in a low-power pulse duration of 1.41–2.11 μs (three times the period of $F_{IrLPBaud16}$). The minimum frequency of $F_{IrLPBaud16}$ ensures that pulses less than one period of $F_{IrLPBaud16}$ are rejected, but pulses greater than 1.4 μs are accepted as valid pulses.

Note: Zero is an illegal value. Programming a zero value results in no $F_{IrLPBaud16}$ 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|---------|-----|-----|-----|-----|-----|-----|-----|
| | 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 | | | | | | | | ILPDVSR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | ILPDVSR | R/W | 0x00 | IrDA Low-Power Divisor This field contains the 8-bit low-power divisor value. |

Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 576 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

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |
| | DIVINT | | | | | | | | | | | | | | | |
| 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. |
| 15:0 | 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 576 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|-----|---------|-----|-----|-----|-----|
| | 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 | | | | | | | | | | | DIVFRAC | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 |
|-----------|----------|------|------------|---|
| 31:6 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5:0 | DIVFRAC | R/W | 0x0 | Fractional Baud-Rate Divisor |

Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x02C
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-----|------|-----|-----|------|-----|-----|-----|
| | 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 | | | | | | | | SPS | WLEN | | FEN | STP2 | EPS | PEN | BRK |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|--|------|-----------|--|-------|-------------|-----|--|-----|--|-----|--------|-----|--------|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 7 | SPS | R/W | 0 | <p>UART Stick Parity Select</p> <p>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.</p> <p>When this bit is cleared, stick parity is disabled.</p> | | | | | | | | | | |
| 6:5 | WLEN | R/W | 0x0 | <p>UART Word Length</p> <p>The bits indicate the number of data bits transmitted or received in a frame as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>5 bits (default)</td> </tr> <tr> <td>0x1</td> <td>6 bits</td> </tr> <tr> <td>0x2</td> <td>7 bits</td> </tr> <tr> <td>0x3</td> <td>8 bits</td> </tr> </tbody> </table> | Value | Description | 0x0 | 5 bits (default) | 0x1 | 6 bits | 0x2 | 7 bits | 0x3 | 8 bits |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | 5 bits (default) | | | | | | | | | | | | | |
| 0x1 | 6 bits | | | | | | | | | | | | | |
| 0x2 | 7 bits | | | | | | | | | | | | | |
| 0x3 | 8 bits | | | | | | | | | | | | | |
| 4 | FEN | R/W | 0 | <p>UART Enable FIFOs</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The transmit and receive FIFO buffers are enabled (FIFO mode).</td> </tr> <tr> <td>0</td> <td>The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.</td> </tr> </tbody> </table> | Value | Description | 1 | The transmit and receive FIFO buffers are enabled (FIFO mode). | 0 | The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 1 | The transmit and receive FIFO buffers are enabled (FIFO mode). | | | | | | | | | | | | | |
| 0 | The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers. | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 3 | STP2 | R/W | 0 | <p>UART Two Stop Bits Select</p> <p>Value Description</p> <p>1 Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.</p> <p>When in 7816 smartcard mode (the <code>SMART</code> bit is set in the <code>UARTCTL</code> register), the number of stop bits is forced to 2.</p> <p>0 One stop bit is transmitted at the end of a frame.</p> |
| 2 | EPS | R/W | 0 | <p>UART Even Parity Select</p> <p>Value Description</p> <p>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.</p> <p>0 Odd parity is performed, which checks for an odd number of 1s.</p> <p>This bit has no effect when parity is disabled by the <code>PEN</code> bit.</p> |
| 1 | PEN | R/W | 0 | <p>UART Parity Enable</p> <p>Value Description</p> <p>1 Parity checking and generation is enabled.</p> <p>0 Parity is disabled and no parity bit is added to the data frame.</p> |
| 0 | BRK | R/W | 0 | <p>UART Send Break</p> <p>Value Description</p> <p>1 A Low level is continually output on the <code>UnTx</code> signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods).</p> <p>0 Normal use.</p> |

Register 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (**TXE**) and Receive Enable (**RXE**) bits, which are set.

To enable the UART module, the **UARTEN** bit must be set. If software requires a configuration change in the module, the **UARTEN** bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

Note that bits [15:14,11:10] are only implemented on UART1. These bits are reserved on UART0 and UART2.

Note: The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by clearing bit 4 (**FEN**) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x030
 Type R/W, reset 0x0000.0300

| | | | | | | | | | | | | | | | | |
|-------|----------|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-------|-------|-------|--------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | CTSEN | RTSEN | reserved | RTS | DTR | RXE | TXE | LBE | LIN | HSE | EOT | SMART | SIRLP | SIREN | UARTEN | |
| Type | R/W | R/W | RO | RO | 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 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 15 | CTSEN | R/W | 0 | <p>Enable Clear To Send</p> <p>Value Description</p> <p>1 CTS hardware flow control is enabled. Data is only transmitted when the <code>U1CTS</code> signal is asserted.</p> <p>0 CTS hardware flow control is disabled.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 14 | RTSEN | R/W | 0 | <p>Enable Request to Send</p> <p>Value Description</p> <p>1 RTS hardware flow control is enabled. Data is only requested (by asserting <code>U1RTS</code>) when the receive FIFO has available entries.</p> <p>0 RTS hardware flow control is disabled.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 13:12 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 11 | RTS | R/W | 0 | <p>Request to Send</p> <p>When <code>RTSEN</code> is clear, the status of this bit is reflected on the <code>U1RTS</code> signal. If <code>RTSEN</code> is set, this bit is ignored on a write and should be ignored on read.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 10 | DTR | R/W | 0 | <p>Data Terminal Ready</p> <p>This bit sets the state of the <code>U1DTR</code> output.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 9 | RXE | R/W | 1 | <p>UART Receive Enable</p> <p>Value Description</p> <p>1 The receive section of the UART is enabled.</p> <p>0 The receive section of the UART is disabled.</p> <p>If the UART is disabled in the middle of a receive, it completes the current character before stopping.</p> <p>Note: To enable reception, the <code>UARTEN</code> bit must also be set.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 8 | TXE | R/W | 1 | <p>UART Transmit Enable</p> <p>Value Description</p> <p>1 The transmit section of the UART is enabled.</p> <p>0 The transmit section of the UART is disabled.</p> <p>If the UART is disabled in the middle of a transmission, it completes the current character before stopping.</p> <p>Note: To enable transmission, the <code>UARTEN</code> bit must also be set.</p> |
| 7 | LBE | R/W | 0 | <p>UART Loop Back Enable</p> <p>Value Description</p> <p>1 The <code>UnTx</code> path is fed through the <code>UnRx</code> path.</p> <p>0 Normal operation.</p> |
| 6 | LIN | R/W | 0 | <p>LIN Mode Enable</p> <p>Value Description</p> <p>1 The UART operates in LIN mode.</p> <p>0 Normal operation.</p> |
| 5 | HSE | R/W | 0 | <p>High-Speed Enable</p> <p>Value Description</p> <p>1 The UART is clocked using the system clock divided by 16.</p> <p>0 The UART is clocked using the system clock divided by 8.</p> <p>Note: System clock used is also dependent on the baud-rate divisor configuration (see page 595) and page 596).</p> |
| 4 | EOT | R/W | 0 | <p>End of Transmission</p> <p>This bit determines the behavior of the <code>TXRIS</code> bit in the <code>UARTRIS</code> register.</p> <p>Value Description</p> <p>1 The <code>TXRIS</code> bit is set only once all transmitted data, including stop bits, have cleared the serializer.</p> <p>0 The <code>TXRIS</code> bit is set when the transmit FIFO condition specified in <code>UARTIFLS</code> is met.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 3 | SMART | R/W | 0 | <p>ISO 7816 Smart Card Support</p> <p>Value Description</p> <p>1 The UART operates in Smart Card mode.</p> <p>0 Normal operation.</p> <p>The application must ensure that it sets 8-bit word length (<i>WLEN</i> set to 0x3) and even parity (<i>PEN</i> set to 1, <i>EPS</i> set to 1, <i>SPS</i> set to 0) in UARTLCRH when using ISO 7816 mode.</p> <p>In this mode, the value of the <i>STP2</i> bit in UARTLCRH is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message.</p> |
| 2 | SIRLP | R/W | 0 | <p>UART SIR Low-Power Mode</p> <p>This bit selects the IrDA encoding mode.</p> <p>Value Description</p> <p>1 The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the <i>IrLPPBaud16</i> input signal, regardless of the selected bit rate.</p> <p>0 Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.</p> <p>Setting this bit uses less power, but might reduce transmission distances. See page 594 for more information.</p> |
| 1 | SIREN | R/W | 0 | <p>UART SIR Enable</p> <p>Value Description</p> <p>1 The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.</p> <p>0 Normal operation.</p> |
| 0 | UARTEN | R/W | 0 | <p>UART Enable</p> <p>Value Description</p> <p>1 The UART is enabled.</p> <p>0 The UART is disabled.</p> <p>If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.</p> |

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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----------|-----|-----|----------|-----|-----|
| | 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 | | | | | | | | | | RXIFLSEL | | | TXIFLSEL | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 | 1 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:6 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5:3 | RXIFLSEL | R/W | 0x2 | UART Receive Interrupt FIFO Level Select |

The trigger points for the receive interrupt are as follows:

| Value | Description |
|---------|---|
| 0x0 | RX FIFO $\geq \frac{1}{8}$ full |
| 0x1 | RX FIFO $\geq \frac{1}{4}$ full |
| 0x2 | RX FIFO $\geq \frac{1}{2}$ full (default) |
| 0x3 | RX FIFO $\geq \frac{3}{4}$ full |
| 0x4 | RX FIFO $\geq \frac{7}{8}$ 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 \leq $\frac{1}{8}$ full 0x1 TX FIFO \leq $\frac{1}{4}$ full 0x2 TX FIFO \leq $\frac{1}{2}$ full (default) 0x3 TX FIFO \leq $\frac{3}{4}$ full 0x4 TX FIFO \leq full 0x5-0x7 Reserved Note: If the EOT bit in UARTCTL is set (see page 599), the transmit interrupt is generated once the FIFO is completely empty and all data including stop bits have left the transmit serializer. In this case, the setting of TXIFLSEL is ignored. |

Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x038
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|--------|--------|----------|----------|------|------|------|------|------|------|------|-------|-------|-------|------|
| | 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 |
| | LME5IM | LME1IM | LMSBIM | reserved | reserved | OEIM | BEIM | PEIM | FEIM | RTIM | TXIM | RXIM | DSRIM | DCDIM | CTSIM | RIIM |
| Type | R/W | R/W | R/W | RO | RO | 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 | LME5IM | R/W | 0 | LIN Mode Edge 5 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the LME5RIS bit in the UARTRIS register is set. 0 The LME5RIS interrupt is suppressed and not sent to the interrupt controller. |
| 14 | LME1IM | R/W | 0 | LIN Mode Edge 1 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the LME1RIS bit in the UARTRIS register is set. 0 The LME1RIS interrupt is suppressed and not sent to the interrupt controller. |
| 13 | LMSBIM | R/W | 0 | LIN Mode Sync Break Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the LMSBRIS bit in the UARTRIS register is set. 0 The LMSBRIS interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 12:11 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10 | OEIM | R/W | 0 | UART Overrun Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the <code>OERIS</code> bit in the UARTRIS register is set. 0 The <code>OERIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 9 | BEIM | R/W | 0 | UART Break Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the <code>BERIS</code> bit in the UARTRIS register is set. 0 The <code>BERIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 8 | PEIM | R/W | 0 | UART Parity Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the <code>PERIS</code> bit in the UARTRIS register is set. 0 The <code>PERIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 7 | FEIM | R/W | 0 | UART Framing Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the <code>FERIS</code> bit in the UARTRIS register is set. 0 The <code>FERIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 6 | RTIM | R/W | 0 | UART Receive Time-Out Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the <code>RTRIS</code> bit in the UARTRIS register is set. 0 The <code>RTRIS</code> interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|--|
| 5 | TXIM | R/W | 0 | <p>UART Transmit Interrupt Mask</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the TXRIS bit in the UARTRIS register is set.</p> <p>0 The TXRIS interrupt is suppressed and not sent to the interrupt controller.</p> |
| 4 | RXIM | R/W | 0 | <p>UART Receive Interrupt Mask</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the RXRIS bit in the UARTRIS register is set.</p> <p>0 The RXRIS interrupt is suppressed and not sent to the interrupt controller.</p> |
| 3 | DSRIM | R/W | 0 | <p>UART Data Set Ready Modem Interrupt Mask</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the DSRRIS bit in the UARTRIS register is set.</p> <p>0 The DSRRIS interrupt is suppressed and not sent to the interrupt controller.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 2 | DCDIM | R/W | 0 | <p>UART Data Carrier Detect Modem Interrupt Mask</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the DCDRIS bit in the UARTRIS register is set.</p> <p>0 The DCDRIS interrupt is suppressed and not sent to the interrupt controller.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 1 | CTSIM | R/W | 0 | <p>UART Clear to Send Modem Interrupt Mask</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the CTSRIS bit in the UARTRIS register is set.</p> <p>0 The CTSRIS interrupt is suppressed and not sent to the interrupt controller.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 0 | RIM | R/W | 0 | UART Ring Indicator Modem Interrupt Mask |
| | | | | Value Description |
| | | | 1 | An interrupt is sent to the interrupt controller when the <code>RIRIS</code> bit in the UARTRIS register is set. |
| | | | 0 | The <code>RIRIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| | | | | This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x03C
 Type RO, reset 0x0000.000F

| | | | | | | | | | | | | | | | | |
|-------|----------|---------|---------|----------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|-------|----|
| | 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 |
| | LME5RIS | LME1RIS | LMSBRIS | reserved | OERIS | BERIS | PERIS | FERIS | RTRIS | TXRIS | RXRIS | DSRRIS | DCDRIS | CTSRIS | RIRIS | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | 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 | LME5RIS | RO | 0 | <p>LIN Mode Edge 5 Raw Interrupt Status</p> <p>Value Description</p> <p>1 The timer value at the 5th falling edge of the LIN Sync Field has been captured.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the <code>LME5IC</code> bit in the UARTICR register.</p> |
| 14 | LME1RIS | RO | 0 | <p>LIN Mode Edge 1 Raw Interrupt Status</p> <p>Value Description</p> <p>1 The timer value at the 1st falling edge of the LIN Sync Field has been captured.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the <code>LME1IC</code> bit in the UARTICR register.</p> |
| 13 | LMSBRIS | RO | 0 | <p>LIN Mode Sync Break Raw Interrupt Status</p> <p>Value Description</p> <p>1 A LIN Sync Break has been detected.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the <code>LMSBIC</code> bit in the UARTICR register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 12:11 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10 | OERIS | RO | 0 | <p>UART Overrun Error Raw Interrupt Status</p> <p>Value Description</p> <p>1 An overrun error has occurred.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register.</p> |
| 9 | BERIS | RO | 0 | <p>UART Break Error Raw Interrupt Status</p> <p>Value Description</p> <p>1 A break error has occurred.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register.</p> |
| 8 | PERIS | RO | 0 | <p>UART Parity Error Raw Interrupt Status</p> <p>Value Description</p> <p>1 A parity error has occurred.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register.</p> |
| 7 | FERIS | RO | 0 | <p>UART Framing Error Raw Interrupt Status</p> <p>Value Description</p> <p>1 A framing error has occurred.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register.</p> |
| 6 | RTRIS | RO | 0 | <p>UART Receive Time-Out Raw Interrupt Status</p> <p>Value Description</p> <p>1 A receive time out has occurred.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 5 | TXRIS | RO | 0 | <p>UART Transmit Raw Interrupt Status</p> <p>Value Description</p> <p>1 If the EOT bit in the UARTCTRL register is clear, the transmit FIFO level has passed through the condition defined in the UARTIFLS register.</p> <p>If the EOT bit is clear, the last bit of all transmitted data and flags has left the serializer.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register.</p> |
| 4 | RXRIS | RO | 0 | <p>UART Receive Raw Interrupt Status</p> <p>Value Description</p> <p>1 The receive FIFO level has passed through the condition defined in the UARTIFLS register.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the RXIC bit in the UARTICR register.</p> |
| 3 | DSRRIS | RO | 0 | <p>UART Data Set Ready Modem Raw Interrupt Status</p> <p>Value Description</p> <p>1 Data Set Ready used for software flow control.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the DSRIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 2 | DCDRIS | RO | 0 | <p>UART Data Carrier Detect Modem Raw Interrupt Status</p> <p>Value Description</p> <p>1 Data Carrier Detect used for software flow control.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the DCDIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 1 | CTSRIS | RO | 0 | <p>UART Clear to Send Modem Raw Interrupt Status</p> <p>Value Description</p> <p>1 Clear to Send used for software flow control.</p> <p>0 No interrupt</p> <p>This bit is cleared by writing a 1 to the CTSIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|--|
| 0 | RIRIS | RO | 0 | UART Ring Indicator Modem Raw Interrupt Status Value Description 1 Ring Indicator used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the RIIIC bit in the UARTICR register. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x040
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|---------|---------|----------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|-------|----|
| | 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 |
| | LME5MIS | LME1MIS | LMSBMIS | reserved | OEMIS | BEMIS | PEMIS | FEMIS | RTMIS | TXMIS | RXMIS | DSRMIS | DCDMIS | CTSMIS | RIMIS | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|--|
| 31:16 | 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 | LME5MIS | RO | 0 | <p>LIN Mode Edge 5 Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to the 5th falling edge of the LIN Sync Field.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the LME5IC bit in the UARTICR register.</p> |
| 14 | LME1MIS | RO | 0 | <p>LIN Mode Edge 1 Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to the 1st falling edge of the LIN Sync Field.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the LME1IC bit in the UARTICR register.</p> |
| 13 | LMSBMIS | RO | 0 | <p>LIN Mode Sync Break Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due the receipt of a LIN Sync Break.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the LMSBIC bit in the UARTICR register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 12:11 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10 | OEMIS | RO | 0 | <p>UART Overrun Error Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to an overrun error.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register.</p> |
| 9 | BEMIS | RO | 0 | <p>UART Break Error Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to a break error.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register.</p> |
| 8 | PEMIS | RO | 0 | <p>UART Parity Error Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to a parity error.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register.</p> |
| 7 | FEMIS | RO | 0 | <p>UART Framing Error Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to a framing error.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register.</p> |
| 6 | RTMIS | RO | 0 | <p>UART Receive Time-Out Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to a receive time out.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register.</p> |
| 5 | TXMIS | RO | 0 | <p>UART Transmit Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set).</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 4 | RXMIS | RO | 0 | <p>UART Receive Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to passing through the specified receive FIFO level.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the <code>RXIC</code> bit in the UARTICR register.</p> |
| 3 | DSRMIS | RO | 0 | <p>UART Data Set Ready Modem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Data Set Ready.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the <code>DSRIC</code> bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 2 | DCDMIS | RO | 0 | <p>UART Data Carrier Detect odem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Data Carrier Detect.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the <code>DCDIC</code> bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 1 | CTSMIS | RO | 0 | <p>UART Clear to Send Modem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Clear to Send.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the <code>CTSIC</code> bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |
| 0 | RIMIS | RO | 0 | <p>UART Ring Indicator Modem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Ring Indicator.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the <code>RIIC</code> bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p> |

Register 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x044
 Type W1C, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|---------|---------|----------|------|------|------|------|------|------|------|--------|--------|--------|-------|-----|
| | 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 |
| | LME5MIC | LME1MIC | LMSBMIC | reserved | OEIC | BEIC | PEIC | FEIC | RTIC | TXIC | RXIC | DSRMIC | DCDMIC | CTSMIC | RIMIC | |
| Type | W1C | W1C | W1C | RO | RO | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 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 | LME5MIC | W1C | 0 | LIN Mode Edge 5 Interrupt Clear Writing a 1 to this bit clears the LME5RIS bit in the UARTRIS register and the LME5MIS bit in the UARTMIS register. |
| 14 | LME1MIC | W1C | 0 | LIN Mode Edge 1 Interrupt Clear Writing a 1 to this bit clears the LME1RIS bit in the UARTRIS register and the LME1MIS bit in the UARTMIS register. |
| 13 | LMSBMIC | W1C | 0 | LIN Mode Sync Break Interrupt Clear Writing a 1 to this bit clears the LMSBRIS bit in the UARTRIS register and the LMSBMIS bit in the UARTMIS register. |
| 12:11 | reserved | RO | 0x0 | Software should not rely on the value of 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 Writing a 1 to this bit clears the OERIS bit in the UARTRIS register and the OEMIS bit in the UARTMIS register. |
| 9 | BEIC | W1C | 0 | Break Error Interrupt Clear Writing a 1 to this bit clears the BERIS bit in the UARTRIS register and the BEMIS bit in the UARTMIS register. |
| 8 | PEIC | W1C | 0 | Parity Error Interrupt Clear Writing a 1 to this bit clears the PERIS bit in the UARTRIS register and the PEMIS bit in the UARTMIS register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 7 | FEIC | W1C | 0 | Framing Error Interrupt Clear Writing a 1 to this bit clears the <code>FERIS</code> bit in the UARTRIS register and the <code>FEMIS</code> bit in the UARTMIS register. |
| 6 | RTIC | W1C | 0 | Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the <code>RTRIS</code> bit in the UARTRIS register and the <code>RTMIS</code> bit in the UARTMIS register. |
| 5 | TXIC | W1C | 0 | Transmit Interrupt Clear Writing a 1 to this bit clears the <code>TXRIS</code> bit in the UARTRIS register and the <code>TXMIS</code> bit in the UARTMIS register. |
| 4 | RXIC | W1C | 0 | Receive Interrupt Clear Writing a 1 to this bit clears the <code>RXRIS</code> bit in the UARTRIS register and the <code>RXMIS</code> bit in the UARTMIS register. |
| 3 | DSRMIC | W1C | 0 | UART Data Set Ready Modem Interrupt Clear Writing a 1 to this bit clears the <code>DSRRIS</code> bit in the UARTRIS register and the <code>DSRMIS</code> bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 2 | DCDMIC | W1C | 0 | UART Data Carrier Detect odem Interrupt Clear Writing a 1 to this bit clears the <code>DCDRIS</code> bit in the UARTRIS register and the <code>DCDMIS</code> bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 1 | CTSMIC | W1C | 0 | UART Clear to Send Modem Interrupt Clear Writing a 1 to this bit clears the <code>CTSRIS</code> bit in the UARTRIS register and the <code>CTSMIS</code> bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |
| 0 | RIMIC | W1C | 0 | UART Ring Indicator Modem Interrupt Clear Writing a 1 to this bit clears the <code>RIRIS</code> bit in the UARTRIS register and the <code>RIMIS</code> bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2. |

Register 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

UART DMA Control (UARTDMACTL)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x048
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|--------|--------|--------|
| | 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 | | | | | | | | | | | | | DMAERR | TXDMAE | RXDMAE |
| 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 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------------|---|
| 31:3 | reserved | RO | 0x00000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | DMAERR | R/W | 0 | DMA on Error Value Description 1 μDMA receive requests are automatically disabled when a receive error occurs. 0 μDMA receive requests are unaffected when a receive error occurs. |
| 1 | TXDMAE | R/W | 0 | Transmit DMA Enable Value Description 1 μDMA for the transmit FIFO is enabled. 0 μDMA for the transmit FIFO is disabled. |
| 0 | RXDMAE | R/W | 0 | Receive DMA Enable Value Description 1 μDMA for the receive FIFO is enabled. 0 μDMA for the receive FIFO is disabled. |

Register 15: UART LIN Control (UARTLCTL), offset 0x090

The **UARTLCTL** register is the configures the operation of the UART when in LIN mode.

UART LIN Control (UARTLCTL)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x090
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|------|-----|----------|----|----|--------|
| | 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 | | | | | | | | | | BLEN | | reserved | | | MASTER |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

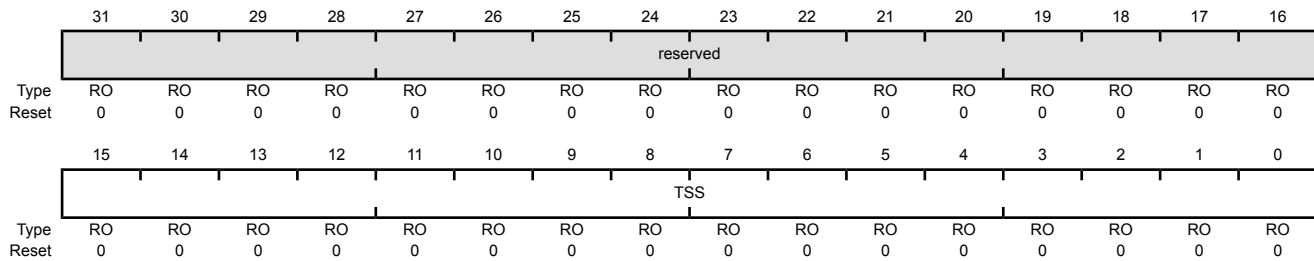
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|--|
| 31:6 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5:4 | BLEN | R/W | 0x0 | Sync Break Length Value Description 0x3 Sync break length is 16T bits 0x2 Sync break length is 15T bits 0x1 Sync break length is 14T bits 0x0 Sync break length is 13T bits (default) |
| 3:1 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | MASTER | R/W | 0 | LIN Master Enable Value Description 1 The UART operates as a LIN master. 0 The UART operates as a LIN slave. |

Register 16: UART LIN Snap Shot (UARTLSS), offset 0x094

The **UARTLSS** register captures the free-running timer value when either the Sync Edge 1 or the Sync Edge 5 is detected in LIN mode.

UART LIN Snap Shot (UARTLSS)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x094
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TSS | RO | 0x0000 | Timer Snap Shot This field contains the value of the free-running timer when either the Sync Edge 5 or the Sync Edge 1 was detected. |

Register 17: UART LIN Timer (UARTLTIM), offset 0x098

The **UARTLTIM** register contains the current timer value for the free-running timer that is used to calculate the baud rate when in LIN slave mode. The value in this register is used along with the value in the **UART LIN Snap Shot (UARTLSS)** register to adjust the baud rate to match that of the master.

UART LIN Timer (UARTLTIM)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x098

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 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 |
| | TIMER | | | | | | | | | | | | | | | |
| 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 |

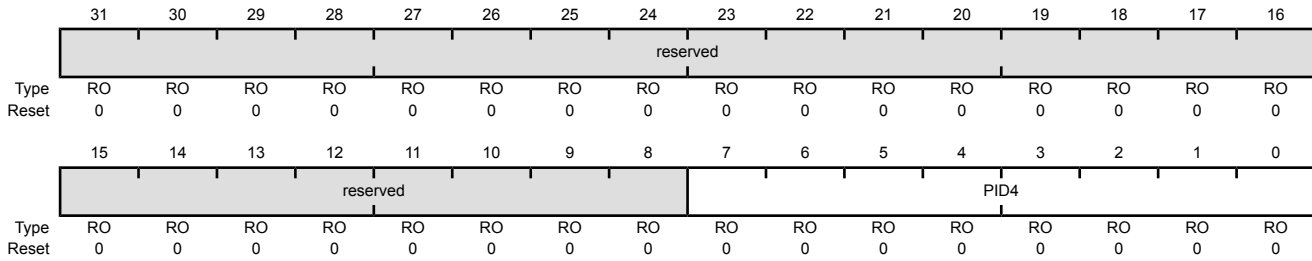
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TIMER | RO | 0x0000 | Timer Value This field contains the value of the free-running timer. |

Register 18: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFD0
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. |

Register 19: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFD4

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID5 | | | | | | | |
| 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 |

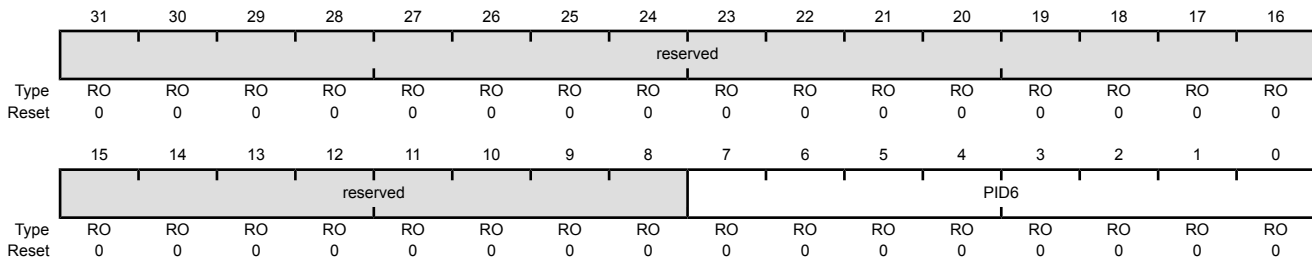
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. |

Register 20: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFD8
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. |

Register 21: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFDC

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID7 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. |

Register 22: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFE0
 Type RO, reset 0x0000.0060

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID0 | | | | | | | |
| 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 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x60 | UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. |

Register 23: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFE4

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID1 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x00 | UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. |

Register 24: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFE8
 Type RO, reset 0x0000.0018

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID2 | | | | | | | |
| 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 | 1 | 1 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. |

Register 25: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFEC

Type RO, reset 0x0000.0001

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID3 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

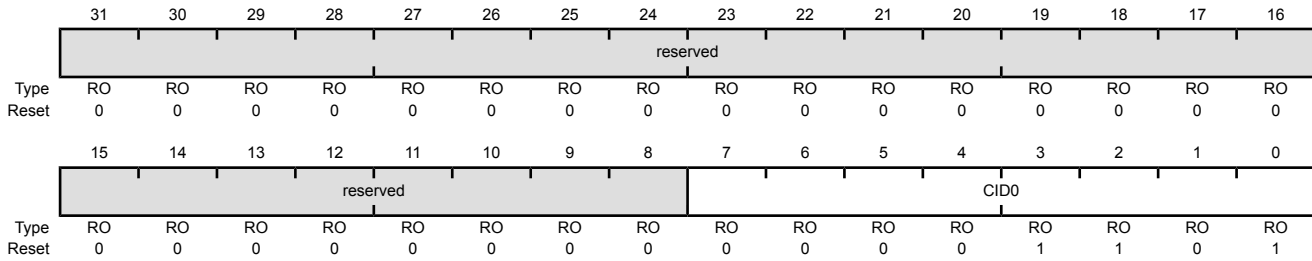
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. |

Register 26: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFF0
 Type RO, reset 0x0000.000D



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID0 | RO | 0x0D | UART PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system. |

Register 27: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFF4

Type RO, reset 0x0000.00F0

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | CID1 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

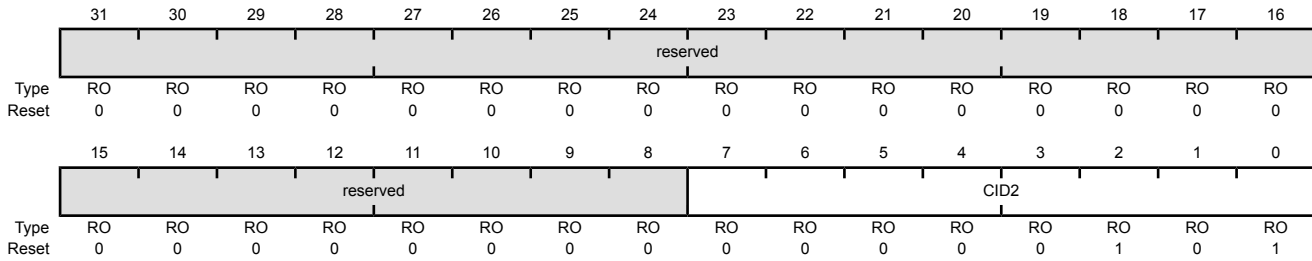
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | UART PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system. |

Register 28: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFF8
 Type RO, reset 0x0000.0005



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | UART PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system. |

Register 29: UART PrimeCell Identification 3 (UARTPCelIID3), offset 0xFFC

The **UARTPCelIIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 3 (UARTPCelIID3)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFFC

Type RO, reset 0x0000.00B1

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID3 | | | | | | | |
| 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 | 1 | 1 | 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. |
| 7:0 | CID3 | RO | 0xB1 | UART PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system. |

16 Synchronous Serial Interface (SSI)

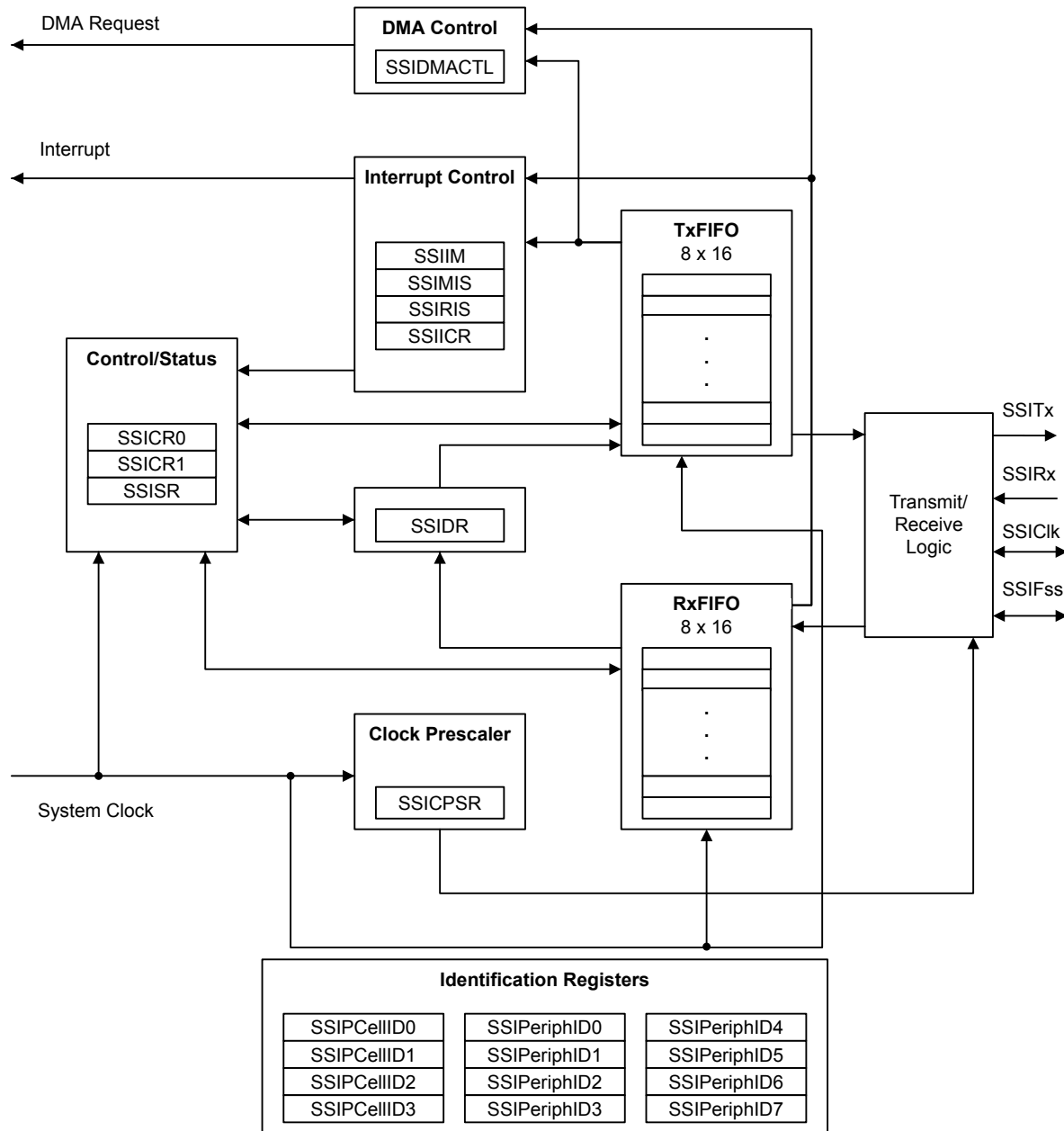
The Stellaris[®] microcontroller includes two Synchronous Serial Interface (SSI) modules. Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris[®] LM3S9B90 controller includes two SSI modules with the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, each 16 bits wide and 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive single request asserted when data is in the FIFO; burst request asserted when FIFO contains 4 entries
 - Transmit single request asserted when there is space in the FIFO; burst request asserted when FIFO contains 4 entries

16.1 Block Diagram

Figure 16-1. SSI Module Block Diagram



16.2 Signal Description

Table 16-1 on page 636 lists the external signals of the SSI module and describes the function of each. The SSI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the `SSI0Clk`, `SSI0Fss`, `SSI0Rx`, and `SSI0Tx` pins which default to the SSI function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the SSI signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the SSI function. The number in

parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 358) to assign the SSI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 318.

Table 16-1. Signals for SSI

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|----------------|--------------------------------|----------|--------------------------|------------------------|
| SSI0Clk | 28 | PA2 (1) | I/O | TTL | SSI module 0 clock. |
| SSI0Fss | 29 | PA3 (1) | I/O | TTL | SSI module 0 frame. |
| SSI0Rx | 30 | PA4 (1) | I | TTL | SSI module 0 receive. |
| SSI0Tx | 31 | PA5 (1) | O | TTL | SSI module 0 transmit. |
| SSI1Clk | 60 74 76 | PF2 (9) PE0 (2) PH4 (11) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | 59 63 75 | PF3 (9) PH5 (11) PE1 (2) | I/O | TTL | SSI module 1 frame. |
| SSI1Rx | 42 62 95 | PF4 (9) PH6 (11) PE2 (2) | I | TTL | SSI module 1 receive. |
| SSI1Tx | 15 41 96 | PH7 (11) PF5 (9) PE3 (2) | O | TTL | SSI module 1 transmit. |

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

16.3 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The SSI also supports the μ DMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the μ DMA module. μ DMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 663).

16.3.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (SysClk). The clock is first divided by an even prescale value `CPSDVSR` from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 656). The clock is further divided by a value from 1 to 256, which is $1 + SCR$, where `SCR` is the value programmed in the **SSI Control 0 (SSICR0)** register (see page 649).

The frequency of the output clock `SSIClk` is defined by:

$$SSIClk = SysClk / (CPSDVSR * (1 + SCR))$$

Note: For master mode, the system clock must be at least two times faster than the `SSIClk`. For slave mode, the system clock must be at least 12 times faster than the `SSIClk`.

See “Synchronous Serial Interface (SSI)” on page 1054 to view SSI timing parameters.

16.3.2 FIFO Operation

16.3.2.1 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 653), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the **SSITx** pin.

In slave mode, the SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the 8th most recent value in the transmit FIFO. If less than 8 values have been written to the transmit FIFO since the SSI module clock was enabled using the **SSI** bit in the **RGCG1** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The SSI can be configured to generate an interrupt or a μ DMA request when the FIFO is empty.

16.3.2.2 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.

16.3.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service (when the transmit FIFO is half full or less)
- Receive FIFO service (when the receive FIFO is half full or more)
- Receive FIFO time-out
- Receive FIFO overrun
- End of transmission

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI generates a single interrupt request to the controller regardless of the number of active interrupts. Each of the four individual maskable interrupts can be masked by clearing the appropriate bit in the **SSI Interrupt Mask (SSIIM)** register (see page 657). Setting the appropriate mask bit enables the interrupt.

The individual outputs, along with a combined interrupt output, allow use of either a global interrupt service routine or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the **SSI Raw Interrupt Status (SSIRIS)** and **SSI Masked Interrupt Status (SSIMIS)** registers (see page 658 and page 660, respectively).

The receive FIFO has a time-out period that is 32 **SSIClk** periods (whether or not **SSIClk** is currently active) and is started when the RX FIFO goes from EMPTY to not-EMPTY. If the RX FIFO is emptied before 32 clocks have passed, the time-out period is reset. As a result, the ISR should

clear the Receive FIFO Time-out Interrupt just after reading out the RX FIFO by writing a 1 to the `RTIC` bit in the **SSI Interrupt Clear (SSIICR)** register. The interrupt should not be cleared so late that the ISR returns before the interrupt is actually cleared, or the ISR may be re-activated unnecessarily.

The End-of-Transmission interrupt is most useful in determining the last value read (such as a response from a command), because there is no $32 \cdot \text{SSIClk}$ delay.

16.3.4 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.

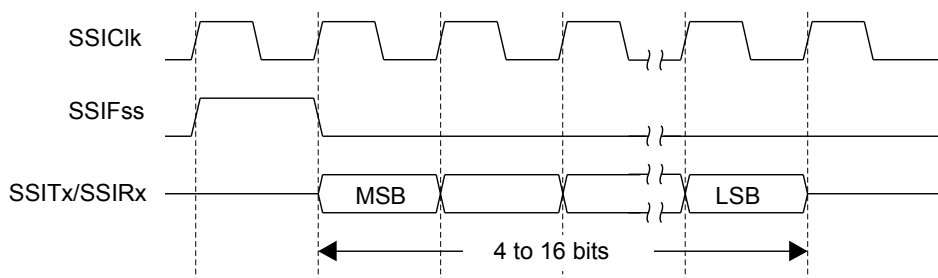
For Texas Instruments synchronous serial frame format, the `SSIFSS` pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of `SSIClk` 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.

16.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 16-2 on page 638 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 16-2. TI Synchronous Serial Frame Format (Single Transfer)

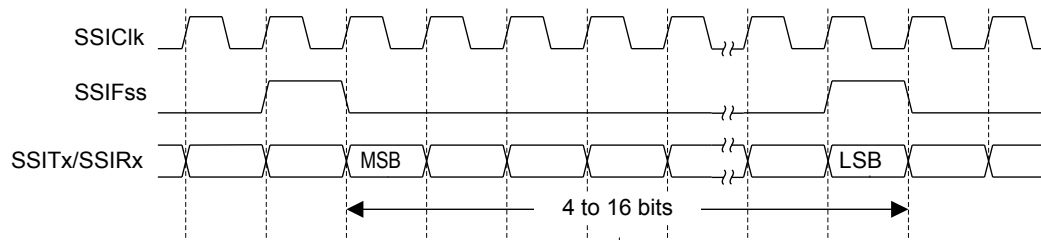


In this mode, $SSIClk$ and $SSIFss$ are forced Low, and the transmit data line $SSITx$ is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, $SSIFss$ is pulsed High for one $SSIClk$ period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of $SSIClk$, the MSB of the 4 to 16-bit data frame is shifted out on the $SSITx$ pin. Likewise, the MSB of the received data is shifted onto the $SSIRx$ pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of $SSIClk$. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of $SSIClk$ after the LSB has been latched.

Figure 16-3 on page 639 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 16-3. TI Synchronous Serial Frame Format (Continuous Transfer)



16.3.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the $SSIFss$ signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the $SSIClk$ signal are programmable through the SPO and SPH bits in the **SSISCR0** control register.

SPO Clock Polarity Bit

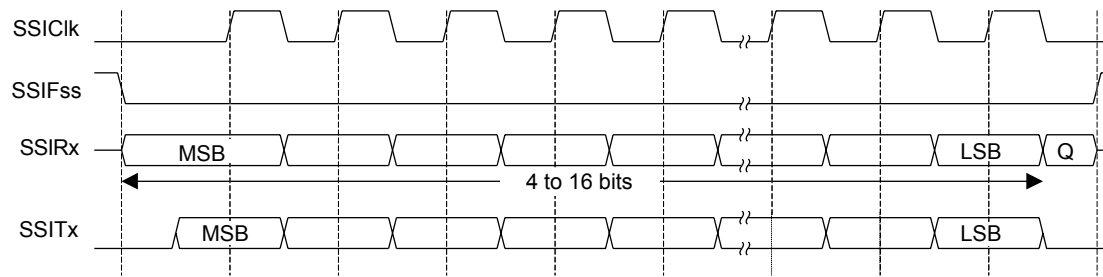
When the SPO clock polarity control bit is clear, it produces a steady state Low value on the $SSIClk$ pin. If the SPO bit is set, a steady state High value is placed on the $SSIClk$ pin when data is not being transferred.

SPH Phase Control Bit

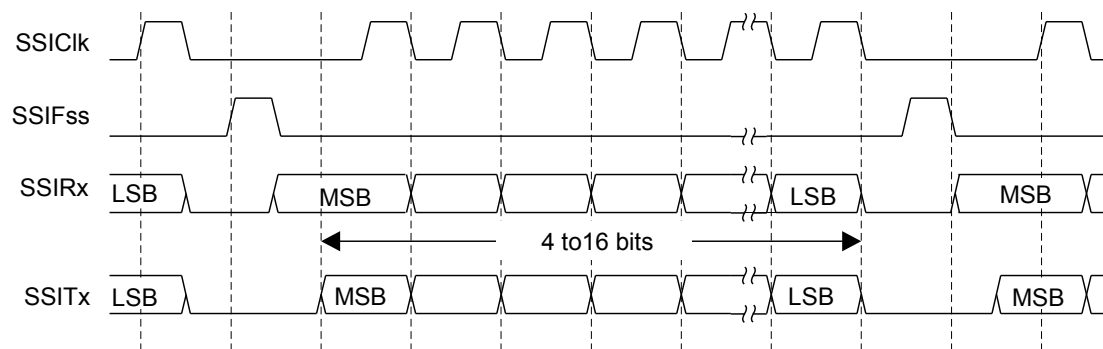
The SPH phase control bit selects the clock edge that captures data and allows it to change state. The state of this bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is clear, data is captured on the first clock edge transition. If the SPH bit is set, data is captured on the second clock edge transition.

16.3.4.3 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 16-4 on page 640 and Figure 16-5 on page 640.

Figure 16-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0

Note: Q is undefined.

Figure 16-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, causing slave data to be enabled onto the SSIRx input line of the master. The master SSITx output pad is enabled.

One half SSIClk period later, valid master data is transferred to the SSITx pin. Once both the master and slave data have been set, the SSIClk master clock pin goes High after one additional half SSIClk period.

The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

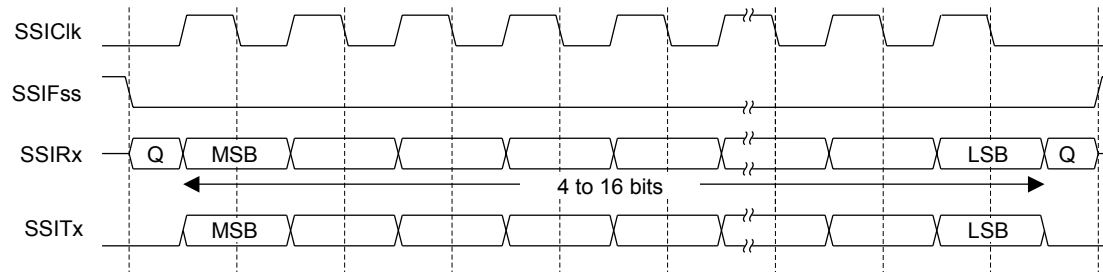
However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the

serial peripheral data write. On completion of the continuous transfer, the $SSIF_{SS}$ pin is returned to its idle state one $SSIClk$ period after the last bit has been captured.

16.3.4.4 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 16-6 on page 641, which covers both single and continuous transfers.

Figure 16-6. Freescale SPI Frame Format with $SPO=0$ and $SPH=1$



Note: Q is undefined.

In this configuration, during idle periods:

- $SSIClk$ is forced Low
- $SSIF_{SS}$ is forced High
- The transmit data line $SSITx$ is arbitrarily forced Low
- When the SSI is configured as a master, it enables the $SSIClk$ pad
- When the SSI is configured as a slave, it disables the $SSIClk$ pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the $SSIF_{SS}$ master signal being driven Low. The master $SSITx$ output is enabled. After an additional one-half $SSIClk$ period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the $SSIClk$ is enabled with a rising edge transition.

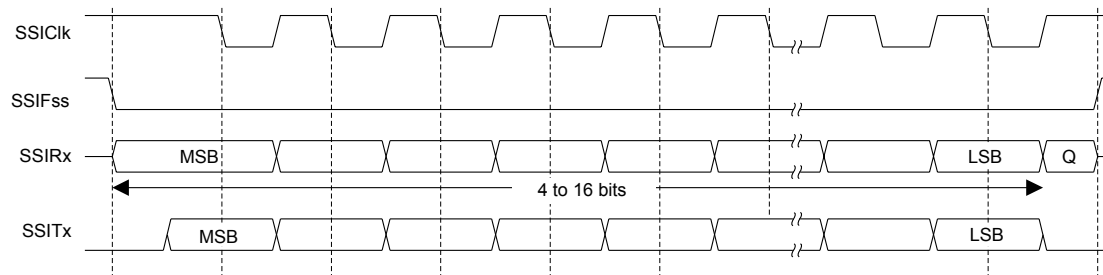
Data is then captured on the falling edges and propagated on the rising edges of the $SSIClk$ signal.

In the case of a single word transfer, after all bits have been transferred, the $SSIF_{SS}$ line is returned to its idle High state one $SSIClk$ period after the last bit has been captured.

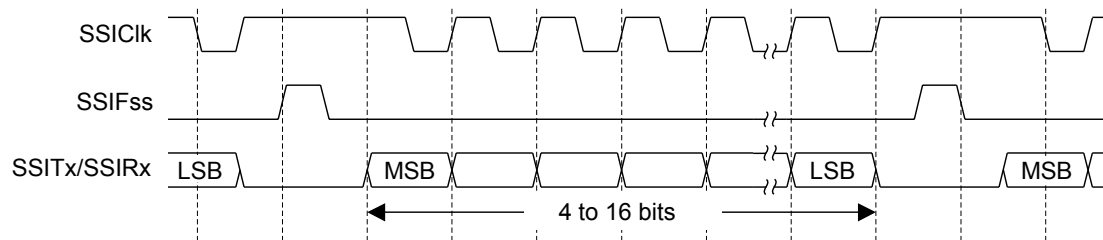
For continuous back-to-back transfers, the $SSIF_{SS}$ pin is held Low between successive data words, and termination is the same as that of the single word transfer.

16.3.4.5 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 16-7 on page 642 and Figure 16-8 on page 642.

Figure 16-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0

Note: Q is undefined.

Figure 16-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, causing slave data to be immediately transferred onto the SSIRx line of the master. The master SSITx output pad is enabled.

One-half period later, valid master data is transferred to the SSITx line. Once both the master and slave data have been set, the SSIClk master clock pin becomes Low after one additional half SSIClk period, meaning that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

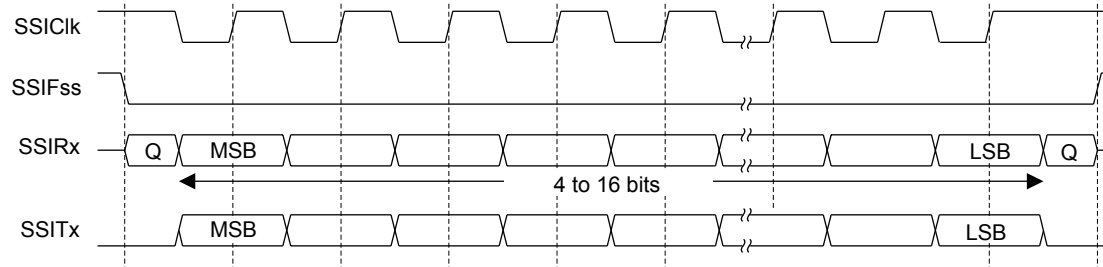
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is clear. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

16.3.4.6 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 16-9 on page 643, which covers both single and continuous transfers.

Figure 16-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output pad is enabled. After an additional one-half SSIClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSIClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSIClk signal.

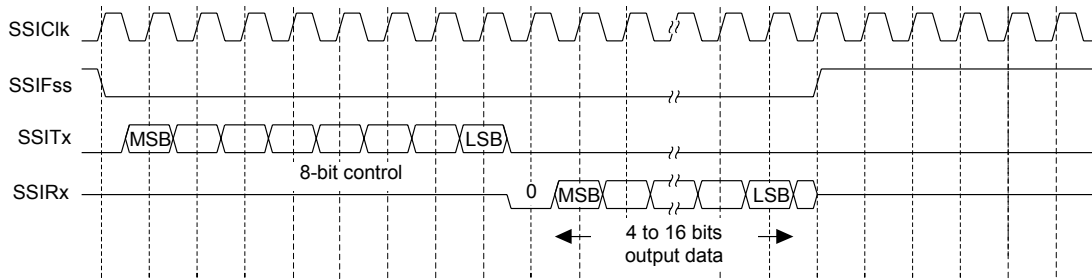
After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

16.3.4.7 MICROWIRE Frame Format

Figure 16-10 on page 644 shows the MICROWIRE frame format for a single frame. Figure 16-11 on page 645 shows the same format when back-to-back frames are transmitted.

Figure 16-10. MICROWIRE Frame Format (Single Frame)

MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex and uses a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

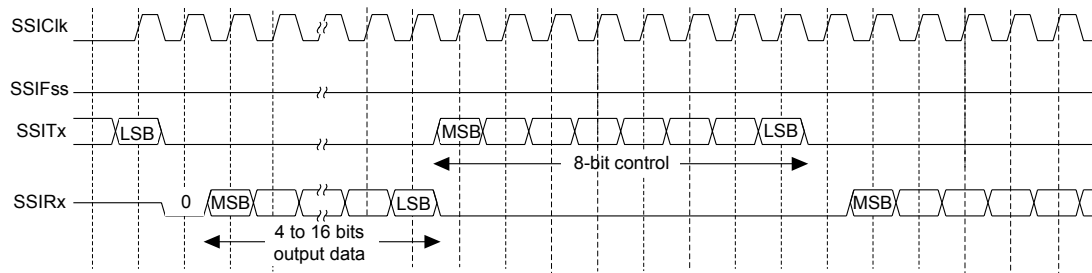
- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic and the MSB of the 8-bit control frame to be shifted out onto the SSITx pin. SSIFss remains Low for the duration of the frame transmission. The SSIRx pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on each rising edge of SSIClk. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the SSIRx line on the falling edge of SSIClk. The SSI in turn latches each bit on the rising edge of SSIClk. At the end of the frame, for single transfers, the SSIFss signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, causing the data to be transferred to the receive FIFO.

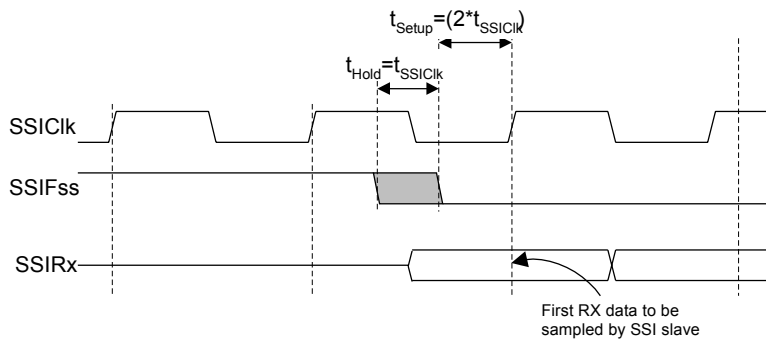
Note: The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter or when the SSIFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.

Figure 16-11. 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 16-12 on page 645 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 16-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements

16.3.5 DMA Operation

The SSI peripheral provides an interface to the μ DMA controller with separate channels for transmit and receive. The μ DMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When μ DMA operation is enabled, the SSI asserts a μ DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever at least one empty location is in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst μ DMA transfer requests are handled automatically by the μ DMA controller depending how the μ DMA channel is configured. To enable μ DMA operation for the receive channel, the `RXDMAE` bit of the **DMA Control (SSIDMACTL)** register should be set. To enable μ DMA operation for the transmit channel, the `TXDMAE` bit of **SSIDMACTL** should be set. If μ DMA is enabled, then the μ DMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and μ DMA is enabled, the SSI interrupt handler must be designed to handle the μ DMA completion interrupt.

See “Micro Direct Memory Access (μ DMA)” on page 258 for more details about programming the μ DMA controller.

16.4 Initialization and Configuration

To enable and initialize the SSI, the following steps are necessary:

1. Enable the SSI module by setting the `SSI` bit in the **RCGC1** register (see page 175).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 184). To find out which GPIO port to enable, refer to Table 24-5 on page 1035.
3. Set the GPIO `AFSEL` bits for the appropriate pins (see page 340). To determine which GPIOs to configure, see Table 24-4 on page 1028.
4. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the SSI signals to the appropriate pins. See page 358 and Table 24-5 on page 1035.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the `SSE` bit in the **SSICR1** register is clear before making any configuration changes.
2. Select whether the SSI is a master or slave:
 - a. For master operations, set the **SSICR1** register to 0x0000.0000.
 - b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
 - c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.
4. Write the **SSICR0** register with the following configuration:
 - Serial clock rate (`SCR`)
 - Desired clock phase/polarity, if using Freescale SPI mode (`SPH` and `SPO`)
 - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (`FRF`)
 - The data size (`DSS`)
5. Optionally, configure the μ DMA channel (see “Micro Direct Memory Access (μ DMA)” on page 258) and enable the DMA option(s) in the **SSIDMACTL** register.
6. Enable the SSI by setting the `SSE` bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (`SPO=1`, `SPH=1`)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

$$\text{SSIClk} = \text{SysClk} / (\text{CPSDVSR} * (1 + \text{SCR})) \times 10^6 = 20 \times 10^6 / (\text{CPSDVSR} * (1 + \text{SCR}))$$

In this case, if CPSDVSR=0x2, SCR must be 0x9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the **SSICR1** register is clear.
2. Write the **SSICR1** register with a value of 0x0000.0000.
3. Write the **SSICPSR** register with a value of 0x0000.0002.
4. Write the **SSICR0** register with a value of 0x0000.09C7.
5. The SSI is then enabled by setting the SSE bit in the **SSICR1** register.

16.5 Register Map

Table 16-2 on page 647 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000
- SSI1: 0x4000.9000

Note that the SSI module clock must be enabled before the registers can be programmed (see page 175).

Note: The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 16-2. SSI Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|--------------|------|-------------|---------------------------------|----------|
| 0x000 | SSICR0 | R/W | 0x0000.0000 | SSI Control 0 | 649 |
| 0x004 | SSICR1 | R/W | 0x0000.0000 | SSI Control 1 | 651 |
| 0x008 | SSIDR | R/W | 0x0000.0000 | SSI Data | 653 |
| 0x00C | SSISR | RO | 0x0000.0003 | SSI Status | 654 |
| 0x010 | SSICPSR | R/W | 0x0000.0000 | SSI Clock Prescale | 656 |
| 0x014 | SSIIM | R/W | 0x0000.0000 | SSI Interrupt Mask | 657 |
| 0x018 | SSIRIS | RO | 0x0000.0008 | SSI Raw Interrupt Status | 658 |
| 0x01C | SSIMIS | RO | 0x0000.0000 | SSI Masked Interrupt Status | 660 |
| 0x020 | SSIICR | W1C | 0x0000.0000 | SSI Interrupt Clear | 662 |
| 0x024 | SSIDMACTL | R/W | 0x0000.0000 | SSI DMA Control | 663 |
| 0xFD0 | SSIPeriphID4 | RO | 0x0000.0000 | SSI Peripheral Identification 4 | 664 |
| 0xFD4 | SSIPeriphID5 | RO | 0x0000.0000 | SSI Peripheral Identification 5 | 665 |

Table 16-2. SSI Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|--------------|------|-------------|---------------------------------|----------|
| 0xFD8 | SSIPeriphID6 | RO | 0x0000.0000 | SSI Peripheral Identification 6 | 666 |
| 0xFDC | SSIPeriphID7 | RO | 0x0000.0000 | SSI Peripheral Identification 7 | 667 |
| 0xFE0 | SSIPeriphID0 | RO | 0x0000.0022 | SSI Peripheral Identification 0 | 668 |
| 0xFE4 | SSIPeriphID1 | RO | 0x0000.0000 | SSI Peripheral Identification 1 | 669 |
| 0xFE8 | SSIPeriphID2 | RO | 0x0000.0018 | SSI Peripheral Identification 2 | 670 |
| 0xFEC | SSIPeriphID3 | RO | 0x0000.0001 | SSI Peripheral Identification 3 | 671 |
| 0xFF0 | SSIPCellID0 | RO | 0x0000.000D | SSI PrimeCell Identification 0 | 672 |
| 0xFF4 | SSIPCellID1 | RO | 0x0000.00F0 | SSI PrimeCell Identification 1 | 673 |
| 0xFF8 | SSIPCellID2 | RO | 0x0000.0005 | SSI PrimeCell Identification 2 | 674 |
| 0xFFC | SSIPCellID3 | RO | 0x0000.00B1 | SSI PrimeCell Identification 3 | 675 |

16.6 Register Descriptions

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

Register 1: SSI Control 0 (SSICR0), offset 0x000

The **SSICR0** register contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x000
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |
| | SCR | | | | | | | | SPH | SPO | FRF | | DSS | | | |
| 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. |
| 15:8 | SCR | R/W | 0x00 | SSI Serial Clock Rate This bit field is used to generate the transmit and receive bit rate of the SSI. The bit rate is: $BR = SSI\text{Clk} / (\text{CPSDVSR} * (1 + \text{SCR}))$ 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. This bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. Value Description 0 Data is captured on the first clock edge transition. 1 Data is captured on the second clock edge transition. |
| 6 | SPO | R/W | 0 | SSI Serial Clock Polarity Value Description 0 A steady state Low value is placed on the SSIClk pin. 1 A steady state High value is placed on the SSIClk pin when data is not being transferred. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 5:4 | FRF | R/W | 0x0 | SSI Frame Format Select Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Texas Instruments Synchronous Serial Frame Format 0x2 MICROWIRE Frame Format 0x3 Reserved |
| 3:0 | DSS | R/W | 0x0 | SSI Data Size Select Value Data Size 0x0-0x2 Reserved 0x3 4-bit data 0x4 5-bit data 0x5 6-bit data 0x6 7-bit data 0x7 8-bit data 0x8 9-bit data 0x9 10-bit data 0xA 11-bit data 0xB 12-bit data 0xC 13-bit data 0xD 14-bit data 0xE 15-bit data 0xF 16-bit data |

Register 2: SSI Control 1 (SSICR1), offset 0x004

The **SSICR1** register contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x004
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|
| | 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 | | | | | | | | | | | | EOT | 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 | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:5 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | EOT | R/W | 0 | End of Transmission Value Description 0 The TXRIS interrupt indicates that the transmit FIFO is half full or less. 1 The End of Transmit interrupt mode for the TXRIS interrupt is enabled. |
| 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. Value Description 0 SSI can drive the SSITx output in Slave 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 the SSI is disabled ($SSE=0$). Value Description 0 The SSI is configured as a master. 1 The SSI is configured as a slave. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 1 | SSE | R/W | 0 | SSI Synchronous Serial Port Enable Value Description 0 SSI operation is disabled. 1 SSI operation is enabled. Note: This bit must be cleared before any control registers are reprogrammed. |
| 0 | LBM | R/W | 0 | SSI Loopback Mode 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

The **SSIDR** register is 16-bits wide. When the **SSIDR** register is read, the entry in the receive FIFO that is pointed to by the current FIFO read pointer is accessed. When a data value is removed by the SSI receive logic from the incoming data frame, it is placed into the entry in the receive FIFO pointed to by the current FIFO write pointer.

When the **SSIDR** register is written to, the entry in the transmit FIFO that is pointed to by the write pointer is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. Each data value is loaded into the transmit serial shifter, then serially shifted out onto the **SSITx** pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the **SSE** bit in the **SSICR1** register is cleared, allowing the software to fill the transmit FIFO before enabling the SSI.

SSI Data (SSIDR)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x008
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |
| | DATA | | | | | | | | | | | | | | | |
| 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. |
| 15:0 | 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

The **SSISR** register contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x00C
 Type RO, reset 0x0000.0003

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | | | BSY | RFF | RNE | TNF | TFE |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:5 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | BSY | RO | 0 | SSI Busy Bit Value Description 0 The SSI is idle. 1 The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty. |
| 3 | RFF | RO | 0 | SSI Receive FIFO Full Value Description 0 The receive FIFO is not full. 1 The receive FIFO is full. |
| 2 | RNE | RO | 0 | SSI Receive FIFO Not Empty Value Description 0 The receive FIFO is empty. 1 The receive FIFO is not empty. |
| 1 | TNF | RO | 1 | SSI Transmit FIFO Not Full Value Description 0 The transmit FIFO is full. 1 The transmit FIFO is not full. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-----------------------------------|
| 0 | TFE | R0 | 1 | SSI Transmit FIFO Empty |
| | | | | Value Description |
| | | | | 0 The transmit FIFO is not empty. |
| | | | | 1 The transmit FIFO is empty. |

Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

The **SSICPSR** register specifies the division factor which is used to derive the **SSIClk** from the system clock. The clock is further divided by a value from 1 to 256, which is $1 + SCR$. **SCR** is programmed in the **SSICR0** register. The frequency of the **SSIClk** is defined by:

$$SSIClk = SysClk / (CPSDVSR * (1 + SCR))$$

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x010
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|---------|-----|-----|-----|-----|-----|-----|-----|
| | 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 | | | | | | | | CPSDVSR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:8 | reserved | RO | 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 on reset.

On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit sets the mask, preventing the interrupt from being signaled to the interrupt controller. Clearing a bit clears the corresponding mask, enabling the interrupt to be sent to the interrupt controller.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x014
 Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|------|------|------|-------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | | | | TXIM | RXIM | RTIM | RORIM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | TXIM | R/W | 0 | SSI Transmit FIFO Interrupt Mask Value Description 0 The transmit FIFO interrupt is masked. 1 The transmit FIFO interrupt is not masked. |
| 2 | RXIM | R/W | 0 | SSI Receive FIFO Interrupt Mask Value Description 0 The receive FIFO interrupt is masked. 1 The receive FIFO interrupt is not masked. |
| 1 | RTIM | R/W | 0 | SSI Receive Time-Out Interrupt Mask Value Description 0 The receive FIFO time-out interrupt is masked. 1 The receive FIFO time-out interrupt is not masked. |
| 0 | RORIM | R/W | 0 | SSI Receive Overrun Interrupt Mask Value Description 0 The receive FIFO overrun interrupt is masked. 1 The receive FIFO overrun interrupt is not masked. |

Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x018
 Type RO, reset 0x0000.0008

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|-------|--------|
| | 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 | 0 | 1 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | TXRIS | RO | 1 | SSI Transmit FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 If the EOT bit in the SSICR1 register is clear, the transmit FIFO is half full or less. If the EOT bit is set, the transmit FIFO is empty, and the last bit has been transmitted out of the serializer. This bit is cleared when the transmit FIFO is more than half full (if the EOT bit is clear) or when it has any data in it (if the EOT bit is set). |
| 2 | RXRIS | RO | 0 | SSI Receive FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 The receive FIFO is half full or more. This bit is cleared when the receive FIFO is less than half full. |
| 1 | RTRIS | RO | 0 | SSI Receive Time-Out Raw Interrupt Status Value Description 0 No interrupt. 1 The receive time-out has occurred. This bit is cleared when a 1 is written to the RTIC bit in the SSI Interrupt Clear (SSIICR) register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 0 | RORRIS | RO | 0 | SSI Receive Overrun Raw Interrupt Status Value Description 0 No interrupt. 1 The receive FIFO has overflowed This bit is cleared when a 1 is written to the RORIC bit in the SSI Interrupt Clear (SSIICR) register. |

Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C

The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x01C
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-------|-------|-------|--------|
| | 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 | | | | | | | | | | | | TXMIS | RXMIS | RTMIS | RORMIS |
| 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 |
|-----------|----------|------|------------|---|
| 31:4 | 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 | TXMIS | RO | 0 | SSI Transmit FIFO Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the transmit FIFO being half full or less (if the <code>EOT</code> bit is clear) or due to the transmission of the last data bit (if the <code>EOT</code> bit is set). This bit is cleared when the transmit FIFO is more than half full (if the <code>EOT</code> bit is clear) or when it has any data in it (if the <code>EOT</code> bit is set). |
| 2 | RXMIS | RO | 0 | SSI Receive FIFO Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive FIFO being half full or less. This bit is cleared when the receive FIFO is less than half full. |
| 1 | RTMIS | RO | 0 | SSI Receive Time-Out Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive time out. This bit is cleared when a 1 is written to the <code>RTIC</code> bit in the SSI Interrupt Clear (SSIICR) register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 0 | RORMIS | RO | 0 | SSI Receive Overrun Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive FIFO overflowing. This bit is cleared when a 1 is written to the RORIC bit in the SSI Interrupt Clear (SSIICR) register. |

Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0x020
 Type W1C, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|------|-------|-----|
| | 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 | | | | | | | | | | | | | | RTIC | RORIC | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------------|---|
| 31:2 | reserved | RO | 0x0000.0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | RTIC | W1C | 0 | SSI Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the RTRIS bit in the SSIRIS register and the RTMIS bit in the SSIMIS register. |
| 0 | RORIC | W1C | 0 | SSI Receive Overrun Interrupt Clear Writing a 1 to this bit clears the RORRIS bit in the SSIRIS register and the RORMIS bit in the SSIMIS register. |

Register 10: SSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the μ DMA control register.

SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0x024

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 | | | | | | | | | | | | | | TXDMAE | RXDMAE | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | TXDMAE | R/W | 0 | Transmit DMA Enable Value Description 0 μ DMA for the transmit FIFO is disabled. 1 μ DMA for the transmit FIFO is enabled. |
| 0 | RXDMAE | R/W | 0 | Receive DMA Enable Value Description 0 μ DMA for the receive FIFO is disabled. 1 μ DMA for the receive FIFO is enabled. |

Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFD0
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID4 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID4 | RO | 0x00 | SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. |

Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFD4
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID5 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID5 | RO | 0x00 | SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. |

Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFD8
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID6 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID6 | RO | 0x00 | SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. |

Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFDC
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID7 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID7 | RO | 0x00 | SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. |

Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFE0
 Type RO, reset 0x0000.0022

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID0 | | | | | | | |
| 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 | 1 | 0 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID0 | RO | 0x22 | SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral. |

Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFE4

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID1 | | | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID1 | RO | 0x00 | SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral. |

Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFE8
 Type RO, reset 0x0000.0018

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID2 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID2 | RO | 0x18 | SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral. |

Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFEC

Type RO, reset 0x0000.0001

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | PID3 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | PID3 | RO | 0x01 | SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral. |

Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFF0
 Type RO, reset 0x0000.000D

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID0 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 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. |
| 7:0 | CID0 | RO | 0x0D | SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system. |

Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFF4

Type RO, reset 0x0000.00F0

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | CID1 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID1 | RO | 0xF0 | SSI PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system. |

Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFF8
 Type RO, reset 0x0000.0005

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID2 | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | CID2 | RO | 0x05 | SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system. |

Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000
 SSI1 base: 0x4000.9000
 Offset 0xFFC
 Type RO, reset 0x0000.00B1

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|
| | 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 | | | | | | | | CID3 | | | | | | | |
| 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 | 1 | 1 | 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. |
| 7:0 | CID3 | RO | 0xB1 | SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system. |

17 Inter-Integrated Circuit (I²C) Interface

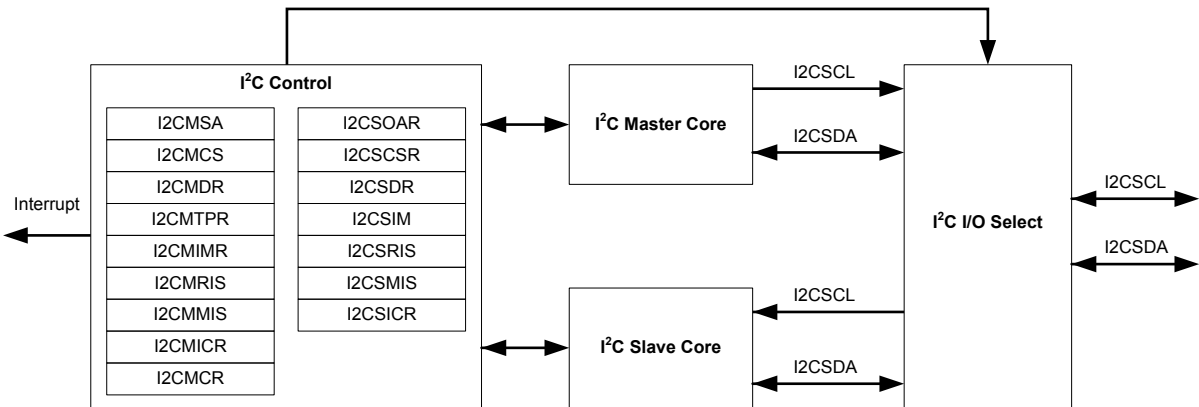
The Inter-Integrated Circuit (I²C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I²C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I²C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S9B90 microcontroller includes two I²C modules, providing the ability to interact (both transmit and receive) with other I²C devices on the bus.

The Stellaris[®] LM3S9B90 controller includes two I²C modules with the following features:

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

17.1 Block Diagram

Figure 17-1. I²C Block Diagram



17.2 Signal Description

Table 17-1 on page 677 lists the external signals of the I²C interface and describes the function of each. The I²C interface signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the I2C0SCL and I2CSDA pins which default to the I²C function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²C signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the I²C function. The number in parentheses is the encoding that must be programmed into the PMC_n field in the **GPIO Port Control (GPIOPTL)** register (page 358) to assign the I²C signal to the specified GPIO port pin. Note that the I²C pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318.

Table 17-1. Signals for I2C

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|----------------------------------|
| I2C0SCL | 72 | PB2 (1) | I/O | OD | I ² C module 0 clock. |
| I2C0SDA | 65 | PB3 (1) | I/O | OD | I ² C module 0 data. |
| I2C1SCL | 14 | PJ0 (11) | I/O | OD | I ² C module 1 clock. |
| | 19 | PG0 (3) | | | |
| | 26 | PA0 (8) | | | |
| | 34 | PA6 (1) | | | |
| I2C1SDA | 18 | PG1 (3) | I/O | OD | I ² C module 1 data. |
| | 27 | PA1 (8) | | | |
| | 35 | PA7 (1) | | | |
| | 87 | PJ1 (11) | | | |

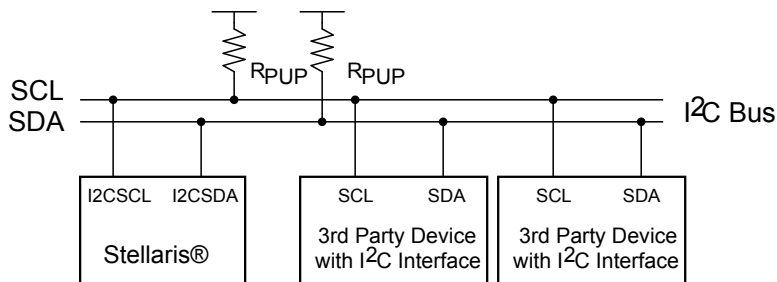
a. The TTL designation indicates the pin has TTL-compatible voltage levels.

17.3 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 17-2.

See “Inter-Integrated Circuit (I²C) Interface” on page 1055 for I²C timing diagrams.

Figure 17-2. I²C Bus Configuration



17.3.1 I²C Bus Functional Overview

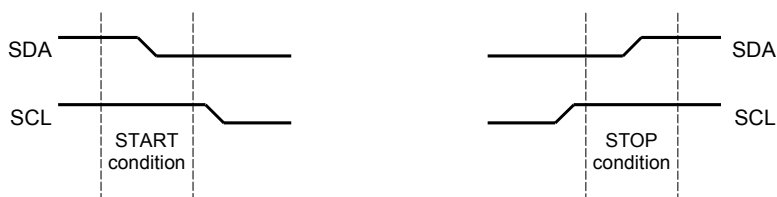
The I²C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on Stellaris[®] microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

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 678) 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.

17.3.1.1 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 17-3.

Figure 17-3. START and STOP Conditions



The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I²C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may

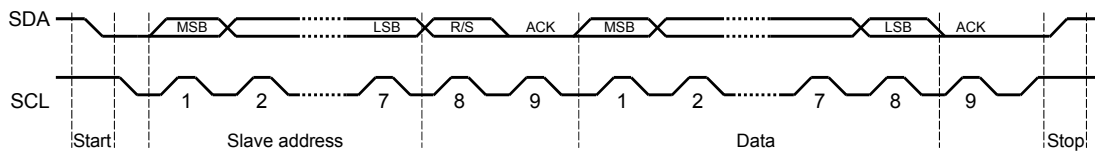
be read from the **I2CMDR** register. When the I²C module operates in Master receiver mode, the **ACK** bit is normally set causing the I²C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

When operating in slave mode, two bits in the **I2CSRIS** register indicate detection of start and stop conditions on the bus; while two bits in the **I2CSMIS** register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

17.3.1.2 Data Format with 7-Bit Address

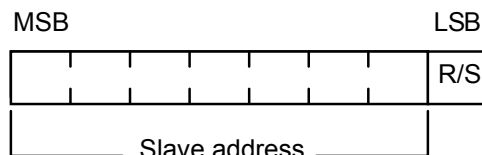
Data transfers follow the format shown in Figure 17-4. After the START condition, a slave address is transmitted. This address is 7-bits long followed by an eighth bit, which is a data direction bit (**R/S** bit in the **I2CMSA** register). If the **R/S** bit is clear, it indicates a transmit operation (send), and if it is set, it indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/transmit formats are then possible within a single transfer.

Figure 17-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 17-5). The eighth bit determines the direction of the message. A zero in the **R/S** position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave.

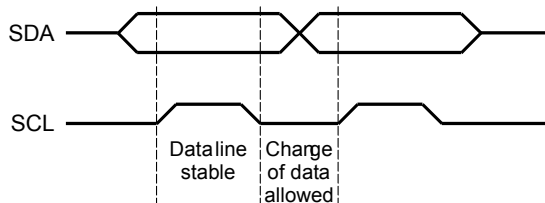
Figure 17-5. R/S Bit in First Byte



17.3.1.3 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 17-6).

Figure 17-6. Data Validity During Bit Transfer on the I²C Bus



17.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data transmitted out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 679.

When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

17.3.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a '1' (High) on SDA while another master transmits a '0' (Low) switches off its data output stage and retires until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

17.3.2 Available Speed Modes

The I²C bus can run in either Standard mode (100 kbps) or Fast mode (400 kbps). The selected mode should match the speed of the other I²C devices on the bus. The mode is selected by using a value in the **I²C Master Timer Period (I2CMTPR)** register that results in an SCL frequency of 100 kbps for Standard mode or 400 kbps for Fast mode.

The I²C clock rate is determined by the parameters *CLK_PRD*, *TIMER_PRD*, *SCL_LP*, and *SCL_HP* where:

CLK_PRD is the system clock period

SCL_LP is the low phase of SCL (fixed at 6)

SCL_HP is the high phase of SCL (fixed at 4)

TIMER_PRD is the programmed value in the **I2CMTPR** register (see page 699).

The I²C clock period is calculated as follows:

$$SCL_PERIOD = 2 \times (1 + TIMER_PRD) \times (SCL_LP + SCL_HP) \times CLK_PRD$$

For example:

$$CLK_PRD = 50 \text{ ns}$$

$$TIMER_PRD = 2$$

$$SCL_LP=6$$

$$SCL_HP=4$$

yields a SCL frequency of:

$$1/SCL_PERIOD = 333 \text{ Khz}$$

Table 17-2 gives examples of the timer periods that should be used to generate both Standard and Fast mode SCL frequencies based on various system clock frequencies.

Table 17-2. Examples of I²C Master Timer Period versus Speed Mode

| System Clock | Timer Period | Standard Mode | Timer Period | Fast Mode |
|--------------|--------------|---------------|--------------|-----------|
| 4 MHz | 0x01 | 100 Kbps | - | - |
| 6 MHz | 0x02 | 100 Kbps | - | - |
| 12.5 MHz | 0x06 | 89 Kbps | 0x01 | 312 Kbps |
| 16.7 MHz | 0x08 | 93 Kbps | 0x02 | 278 Kbps |
| 20 MHz | 0x09 | 100 Kbps | 0x02 | 333 Kbps |
| 25 MHz | 0x0C | 96.2 Kbps | 0x03 | 312 Kbps |
| 33 MHz | 0x10 | 97.1 Kbps | 0x04 | 330 Kbps |
| 40 MHz | 0x13 | 100 Kbps | 0x04 | 400 Kbps |
| 50 MHz | 0x18 | 100 Kbps | 0x06 | 357 Kbps |
| 80 MHz | 0x27 | 100 Kbps | 0x09 | 400 Kbps |

17.3.3 Interrupts

The I²C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I²C master and I²C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

17.3.3.1 I²C Master Interrupts

The I²C master module generates an interrupt when a transaction completes (either transmit or receive), 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` bit in the **I²C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction. An error condition is asserted if the last transaction wasn't acknowledged by the slave, or if the master was forced to give up ownership of the bus due to a lost arbitration round with another master. If an error is not detected, 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.

17.3.3.2 I²C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by setting the `DATAIM` bit in the **I²C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I²C Slave Data (I2CSDR)** register, by checking the `RREQ` and `TREQ` bits of the **I²C Slave Control/Status (I2CSCR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the `FBR` bit is set along with the `RREQ` bit. The interrupt is cleared by setting the `DATAIC` bit in the **I²C Slave Interrupt Clear (I2CSICR)** register.

In addition, the slave module can generate an interrupt when a start and stop condition is detected. These interrupts are enabled by setting the `STARTIM` and `STOPIM` bits of the **I²C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the `STOPIC` and `STARTIC` bits of the **I²C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Slave Raw Interrupt Status (I2CSRIS)** register.

17.3.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the `LPBK` bit in the **I²C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

17.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I²C transfer types in both master and slave mode.

17.3.5.1 I²C Master Command Sequences

The figures that follow show the command sequences available for the I²C master.

Figure 17-7. Master Single TRANSMIT

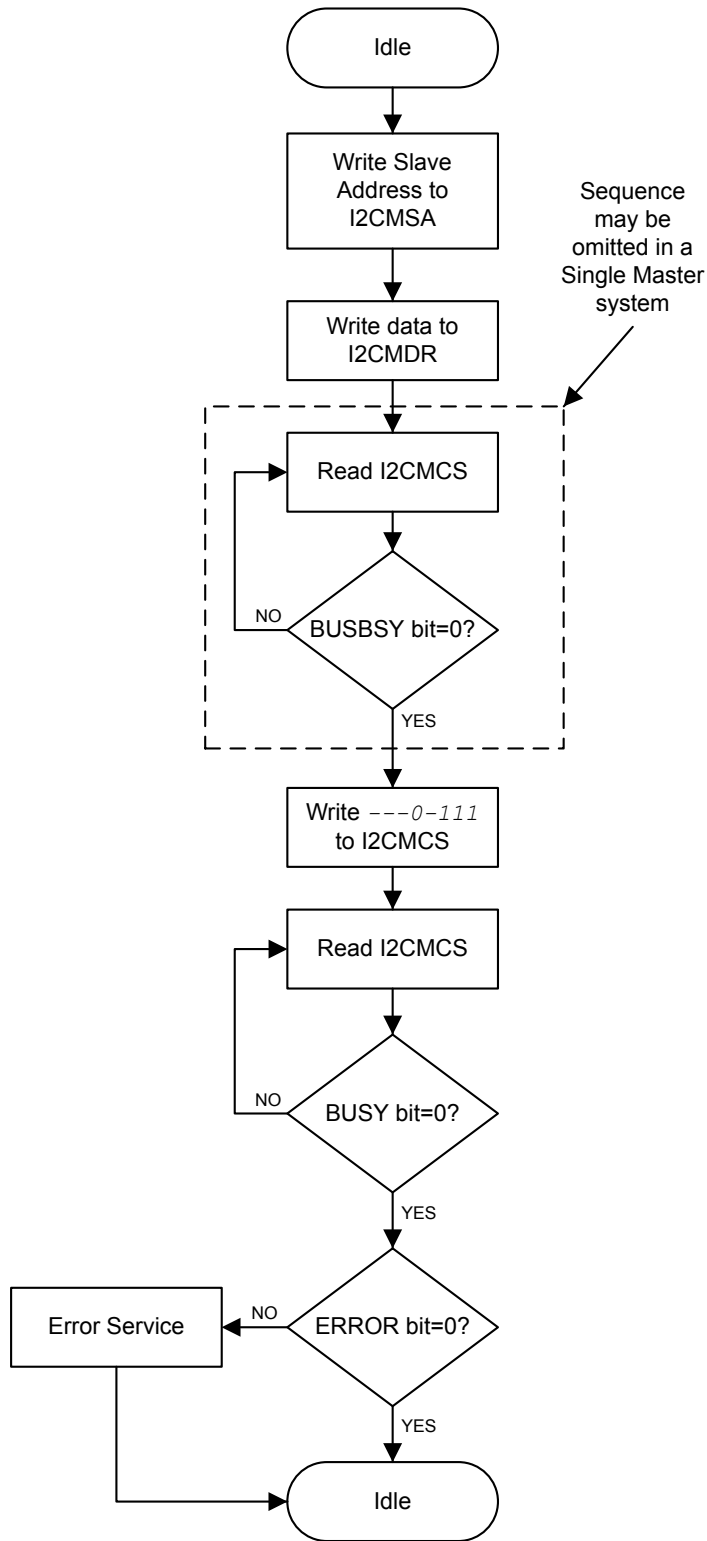


Figure 17-8. Master Single RECEIVE

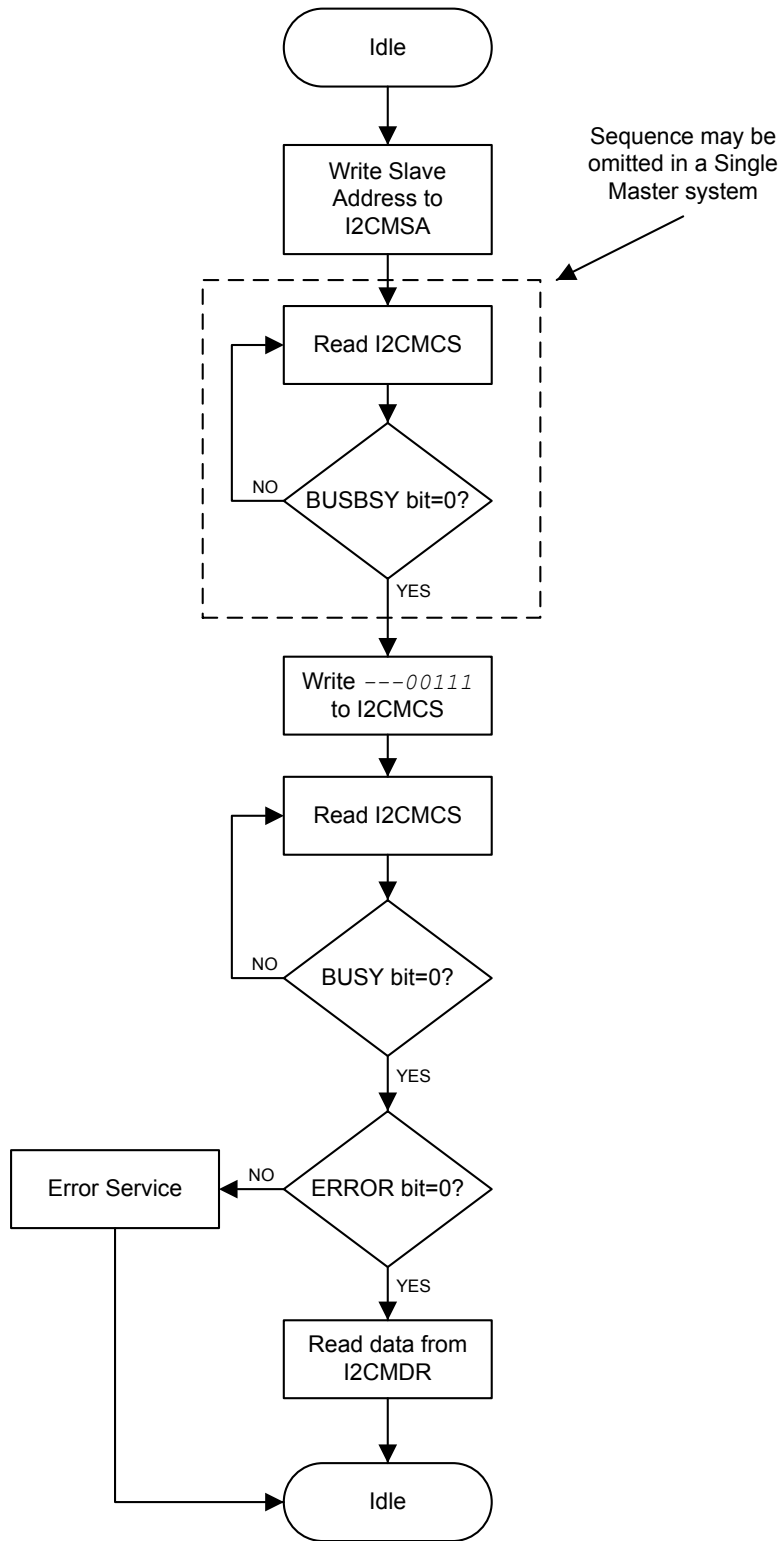


Figure 17-9. Master TRANSMIT with Repeated START

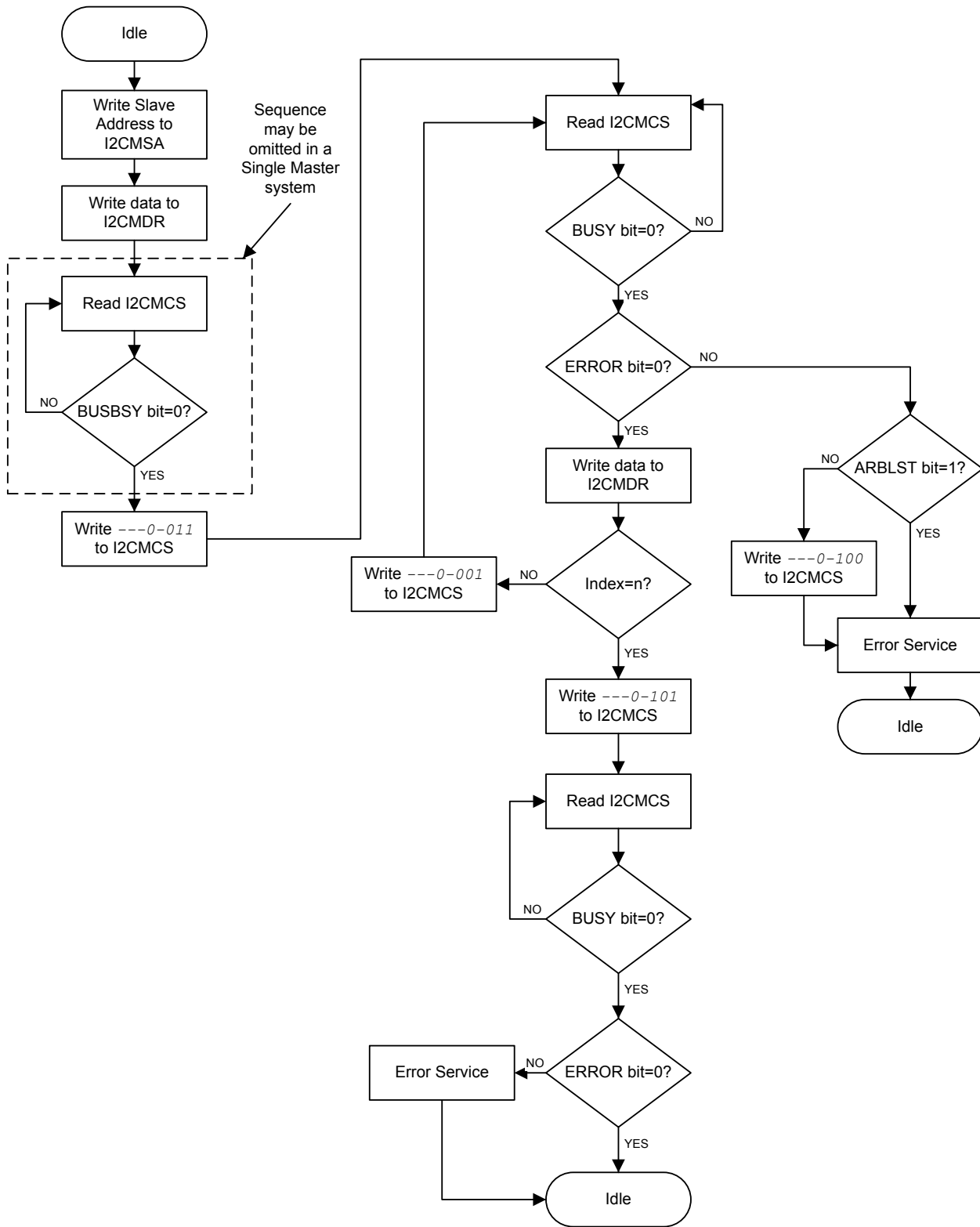


Figure 17-10. Master RECEIVE with Repeated START

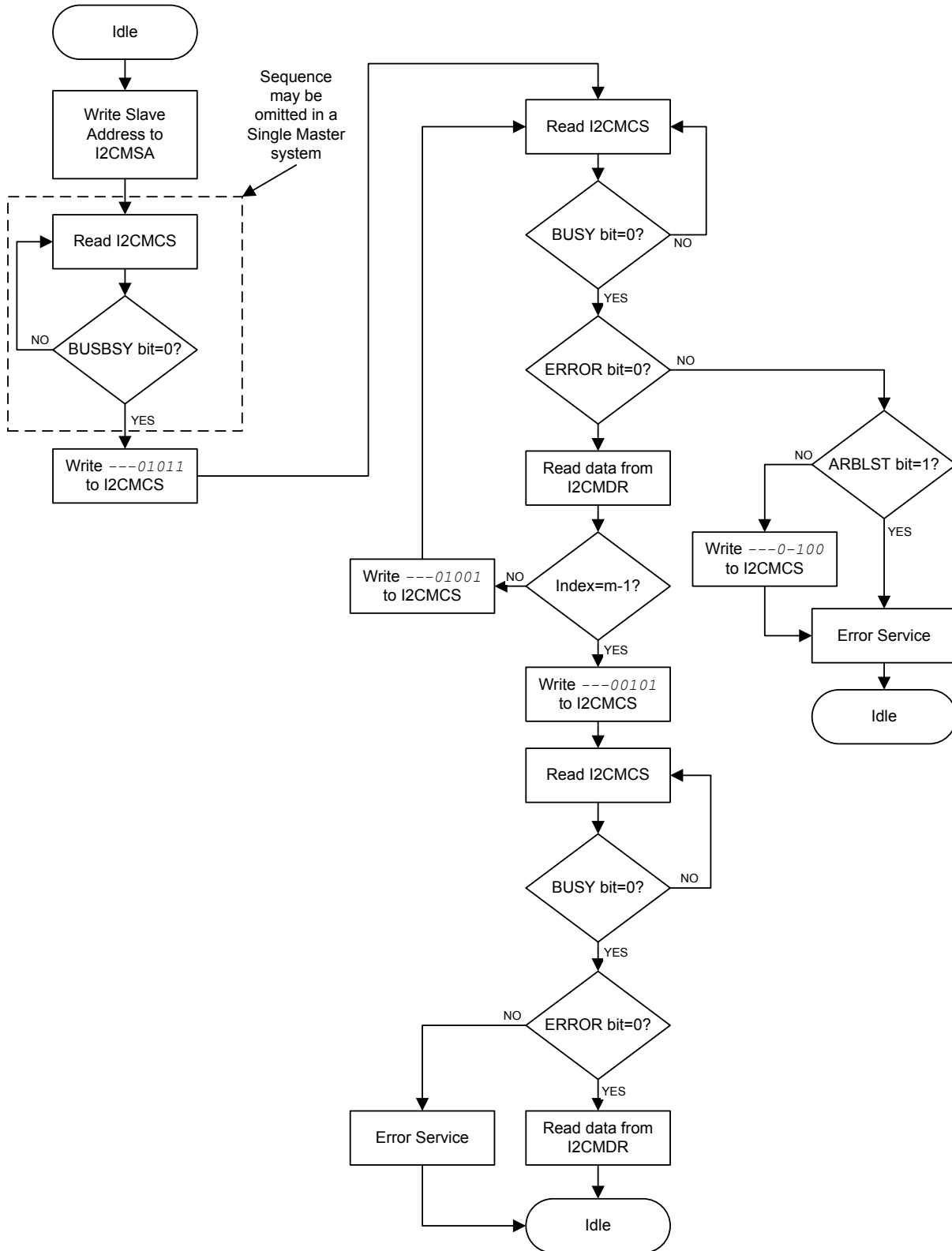
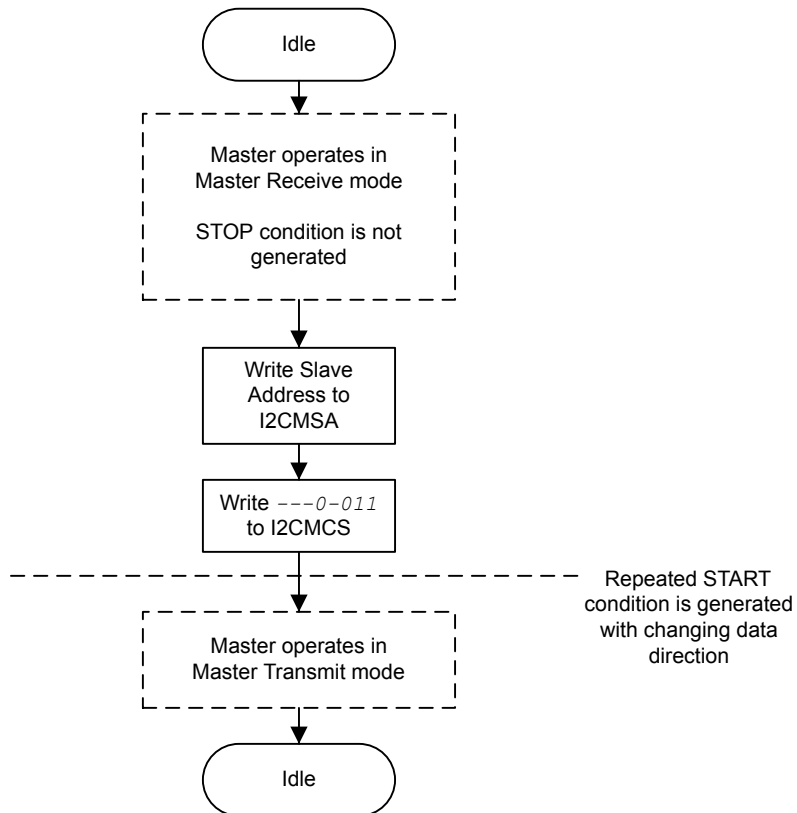


Figure 17-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START



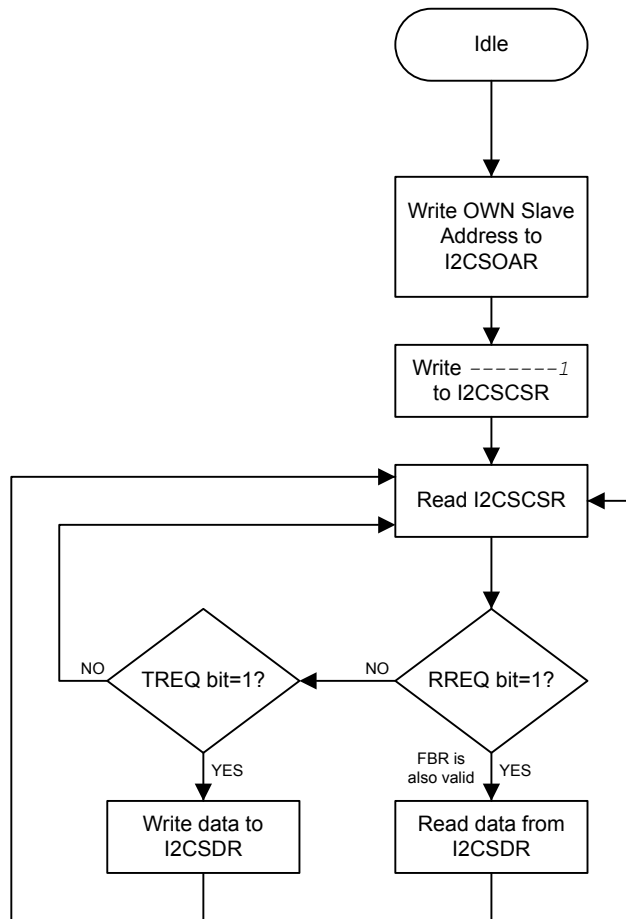
Figure 17-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START



17.3.5.2 I²C Slave Command Sequences

Figure 17-13 on page 689 presents the command sequence available for the I²C slave.

Figure 17-13. Slave Command Sequence



17.4 Initialization and Configuration

The following example shows how to configure the I²C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I²C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module (see page 175).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 184). To find out which GPIO port to enable, refer to Table 24-5 on page 1035.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 340). To determine which GPIOs to configure, see Table 24-4 on page 1028.
4. Enable the I²C pins for Open Drain operation. See page 345.
5. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the I²C signals to the appropriate pins. See page 358 and Table 24-5 on page 1035.
6. Initialize the I²C Master by writing the **I2CMCR** register with a value of 0x0000.0010.

7. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

$$\begin{aligned} \text{TPR} &= (\text{System Clock} / (2 * (\text{SCL_LP} + \text{SCL_HP}) * \text{SCL_CLK})) - 1; \\ \text{TPR} &= (20\text{MHz} / (2 * (6+4) * 100000)) - 1; \\ \text{TPR} &= 9 \end{aligned}$$

Write the **I2CMTPR** register with the value of 0x0000.0009.

8. Specify the slave address of the master and that the next operation is a Transmit by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
9. Place data (byte) to be transmitted in the data register by writing the **I2CMDR** register with the desired data.
10. Initiate a single byte transmit of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
11. Wait until the transmission completes by polling the **I2CMCS** register's **BUSBSY** bit until it has been cleared.

17.5 Register Map

Table 17-3 on page 690 lists the I²C registers. All addresses given are relative to the I²C base addresses for the master and slave:

- I²C Master 0: 0x4002.0000
- I²C Slave 0: 0x4002.0800
- I²C Master 1: 0x4002.1000
- I²C Slave 1: 0x4002.1800

Note that the I²C module clock must be enabled before the registers can be programmed (see page 175).

Table 17-3. Inter-Integrated Circuit (I²C) Interface Register Map

| Offset | Name | Type | Reset | Description | See page |
|------------------------------|---------|------|-------------|------------------------------------|----------|
| I²C Master | | | | | |
| 0x000 | I2CMSA | R/W | 0x0000.0000 | I2C Master Slave Address | 692 |
| 0x004 | I2CMCS | R/W | 0x0000.0000 | I2C Master Control/Status | 693 |
| 0x008 | I2CMDR | R/W | 0x0000.0000 | I2C Master Data | 698 |
| 0x00C | I2CMTPR | R/W | 0x0000.0001 | I2C Master Timer Period | 699 |
| 0x010 | I2CMIMR | R/W | 0x0000.0000 | I2C Master Interrupt Mask | 700 |
| 0x014 | I2CMRIS | RO | 0x0000.0000 | I2C Master Raw Interrupt Status | 701 |
| 0x018 | I2CMMIS | RO | 0x0000.0000 | I2C Master Masked Interrupt Status | 702 |
| 0x01C | I2CMICR | WO | 0x0000.0000 | I2C Master Interrupt Clear | 703 |
| 0x020 | I2CMCR | R/W | 0x0000.0000 | I2C Master Configuration | 704 |

Table 17-3. Inter-Integrated Circuit (I²C) Interface Register Map (*continued*)

| Offset | Name | Type | Reset | Description | See page |
|-----------------------------|---------|------|-------------|-----------------------------------|----------|
| I²C Slave | | | | | |
| 0x000 | I2CSOAR | R/W | 0x0000.0000 | I2C Slave Own Address | 705 |
| 0x004 | I2CSCSR | RO | 0x0000.0000 | I2C Slave Control/Status | 706 |
| 0x008 | I2CSDR | R/W | 0x0000.0000 | I2C Slave Data | 708 |
| 0x00C | I2CSIMR | R/W | 0x0000.0000 | I2C Slave Interrupt Mask | 709 |
| 0x010 | I2CSRIS | RO | 0x0000.0000 | I2C Slave Raw Interrupt Status | 710 |
| 0x014 | I2CSMIS | RO | 0x0000.0000 | I2C Slave Masked Interrupt Status | 711 |
| 0x018 | I2CSICR | WO | 0x0000.0000 | I2C Slave Interrupt Clear | 712 |

17.6 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 704.

Register 1: I²C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Transmit (Low).

I2C Master Slave Address (I2CMSA)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

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 | | | | | | | | | | | | | | | |
| 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 | | | | | | | | SA | | | | | | | R/S |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|-------------|------|-----------|--|-------|-------------|---|----------|---|---------|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | |
| 7:1 | SA | R/W | 0x00 | I ² C 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 Transmit (Low). <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Transmit</td> </tr> <tr> <td>1</td> <td>Receive</td> </tr> </tbody> </table> | Value | Description | 0 | Transmit | 1 | Receive |
| Value | Description | | | | | | | | | |
| 0 | Transmit | | | | | | | | | |
| 1 | Receive | | | | | | | | | |

Register 2: I²C Master Control/Status (I2CMCS), offset 0x004

This register accesses seven status bits when read and four control bits when written.

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 generates the START or REPEATED START condition.

The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I²C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and 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 is normally set causing the I²C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

Read-Only Status Register

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x004

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|--------|------|--------|--------|--------|-------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | BUSBSY | IDLE | ARBLST | DATACK | ADRACK | ERROR | BUSY |
| 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 |
|-----------|----------|------|-----------|---|
| 31:7 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6 | BUSBSY | RO | 0 | <p>Bus Busy</p> <p>Value Description</p> <p>0 The I²C bus is idle.</p> <p>1 The I²C bus is busy.</p> <p>The bit changes based on the START and STOP conditions.</p> |
| 5 | IDLE | RO | 0 | <p>I²C Idle</p> <p>Value Description</p> <p>0 The I²C controller is not idle.</p> <p>1 The I²C controller is idle.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 4 | ARBLST | RO | 0 | Arbitration Lost Value Description 0 The I ² C controller won arbitration. 1 The I ² C controller lost arbitration. |
| 3 | DATAACK | RO | 0 | Acknowledge Data Value Description 0 The transmitted data was acknowledged 1 The transmitted data was not acknowledged. |
| 2 | ADRACK | RO | 0 | Acknowledge Address Value Description 0 The transmitted address was acknowledged 1 The transmitted address was not acknowledged. |
| 1 | ERROR | RO | 0 | Error Value Description 0 No error was detected on the last operation. 1 An error occurred on the last operation. The error can be from the slave address not being acknowledged, the transmit data not being acknowledged, or because the controller lost arbitration. |
| 0 | BUSY | RO | 0 | I ² C Busy Value Description 0 The controller is idle. 1 The controller is busy. When the <code>BUSY</code> bit is set, the other status bits are not valid. |

Write-Only Control Register

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x004

Type WO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-----|------|-------|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | ACK | STOP | START | RUN |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:4 | reserved | WO | 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 | ACK | WO | 0 | Data Acknowledge Enable Value Description 0 The received data byte is not acknowledged automatically by the master. 1 The received data byte is acknowledged automatically by the master. See field decoding in Table 17-4 on page 696. |
| 2 | STOP | WO | 0 | Generate STOP Value Description 0 The controller does not generate the STOP condition. 1 The controller generates the STOP condition. See field decoding in Table 17-4 on page 696. |
| 1 | START | WO | 0 | Generate START Value Description 0 The controller does not generate the START condition. 1 The controller generates the START or repeated START condition. See field decoding in Table 17-4 on page 696. |
| 0 | RUN | WO | 0 | I ² C Master Enable Value Description 0 The master is disabled. 1 The master is enabled to transmit or receive data. See field decoding in Table 17-4 on page 696. |

Table 17-4. Write Field Decoding for I2CMCS[3:0] Field

| Current State | I2CMSA[0] | I2CMCS[3:0] | | | | Description |
|---|---|----------------|------|-------|------|---|
| | R/S | ACK | STOP | START | RUN | |
| Idle | 0 | X ^a | 0 | 1 | 1 | START condition followed by TRANSMIT (master goes to the Master Transmit state). |
| | 0 | X | 1 | 1 | 1 | START condition followed by a TRANSMIT and STOP condition (master remains in Idle state). |
| | 1 | 0 | 0 | 1 | 1 | START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state). |
| | 1 | 0 | 1 | 1 | 1 | START condition followed by RECEIVE and STOP condition (master remains in Idle state). |
| | 1 | 1 | 0 | 1 | 1 | START condition followed by RECEIVE (master goes to the Master Receive state). |
| | 1 | 1 | 1 | 1 | 1 | Illegal |
| | All other combinations not listed are non-operations. | | | | | NOP |
| Master Transmit | X | X | 0 | 0 | 1 | TRANSMIT operation (master remains in Master Transmit state). |
| | X | X | 1 | 0 | 0 | STOP condition (master goes to Idle state). |
| | X | X | 1 | 0 | 1 | TRANSMIT followed by STOP condition (master goes to Idle state). |
| | 0 | X | 0 | 1 | 1 | Repeated START condition followed by a TRANSMIT (master remains in Master Transmit state). |
| | 0 | X | 1 | 1 | 1 | Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state). |
| | 1 | 0 | 0 | 1 | 1 | Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state). |
| | 1 | 0 | 1 | 1 | 1 | Repeated START condition followed by a TRANSMIT and STOP condition (master goes to Idle state). |
| | 1 | 1 | 0 | 1 | 1 | Repeated START condition followed by RECEIVE (master goes to Master Receive state). |
| | 1 | 1 | 1 | 1 | 1 | Illegal. |
| All other combinations not listed are non-operations. | | | | | NOP. | |

Table 17-4. Write Field Decoding for I2CMCS[3:0] Field (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 TRANSMIT (master goes to Master Transmit state). |
| | 0 | X | 1 | 1 | 1 | Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state). |
| | All other combinations not listed are non-operations. | | | | | |

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

Register 3: I²C Master Data (I2CMDR), offset 0x008

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 Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x008

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | DATA | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | 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.

I2C Master Timer Period (I2CMTPR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x00C

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 | | | | | | | | TPR | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | TPR | R/W | 0x1 | SCL Clock Period |

This field specifies the period of the SCL clock.

$$SCL_PRD = 2 \times (1 + TPR) \times (SCL_LP + SCL_HP) \times CLK_PRD$$

where:

SCL_PRD is the SCL line period (I²C clock).

TPR is the Timer Period register value (range of 1 to 255).

SCL_LP is the SCL Low period (fixed at 6).

SCL_HP is the SCL High period (fixed at 4).

CLK_PRD is the system clock period in ns.

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 Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x010

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 | | | | | | | | | | | | | | | IM | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | IM | R/W | 0 | Interrupt Mask |
| | | | | Value Description |
| | | | | 1 The master interrupt is sent to the interrupt controller when the RIS bit in the I2CMRIS register is set. |
| | | | | 0 The RIS interrupt is suppressed and not sent to the interrupt controller. |

Register 6: I²C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

I2C Master Raw Interrupt Status (I2CMRIS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x014

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | RIS |
| 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 |
|-----------|----------|------|------------|---|
| 31:1 | 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 | RIS | RO | 0 | Raw Interrupt Status |

| Value | Description |
|-------|--------------------------------|
| 1 | A master interrupt is pending. |
| 0 | No interrupt. |

This bit is cleared by writing a 1 to the IC bit in the I2CMICR register.

Register 7: I²C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

I2C Master Masked Interrupt Status (I2CMMIS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x018

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | MIS |
| 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 |
|-----------|----------|------|------------|---|
| 31:1 | 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 | MIS | RO | 0 | Masked Interrupt Status |

Value Description

- 1 An unmasked master interrupt was signaled is pending.
- 0 An interrupt has not occurred or is masked.

This bit is cleared by writing a 1 to the IC bit in the I2CMICR register.

Register 8: I²C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x01C

Type WO, 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 | | | | | | | | | | | | | | | IC | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | IC | WO | 0 | <p>Interrupt Clear</p> <p>Writing a 1 to this bit clears the RIS bit in the I2CMRIS register and the MIS bit in the I2CMMIS register.</p> <p>A read of this register returns no meaningful data.</p> |

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 Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x020

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 | | | | | | | | | | | SFE | MFE | reserved | | LPBK | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:6 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | SFE | R/W | 0 | I ² C Slave Function Enable Value Description 1 Slave mode is enabled. 0 Slave mode is disabled. |
| 4 | MFE | R/W | 0 | I ² C Master Function Enable Value Description 1 Master mode is enabled. 0 Master mode is disabled. |
| 3:1 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | LPBK | R/W | 0 | I ² C Loopback Value Description 1 The controller in a test mode loopback configuration. 0 Normal operation. |

17.7 Register Descriptions (I²C Slave)

The remainder of this section lists and describes the I²C slave registers, in numerical order by address offset. See also “Register Descriptions (I²C Master)” on page 691.

Register 10: I²C Slave Own Address (I2CSOAR), offset 0x000

This register consists of seven address bits that identify the Stellaris® I²C device on the I²C bus.

I²C Slave Own Address (I2CSOAR)

I²C Slave 0 base: 0x4002.0800

I²C Slave 1 base: 0x4002.1800

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 | | | | | | | | | | | | | | | |
| 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 | | | | | | | | | | OAR | | | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:7 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | OAR | R/W | 0x00 | I ² C Slave Own Address This field specifies bits A6 through A0 of the slave address. |

Register 11: I²C Slave Control/Status (I2CCSR), offset 0x004

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.

Read-Only Status Register

I2C Slave Control/Status (I2CCSR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x004

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-----|------|------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | FBR | TREQ | RREQ |
| 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 |
|-----------|----------|------|------------|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | FBR | RO | 0 | <p>First Byte Received</p> <p>Value Description</p> <p>1 The first byte following the slave's own address has been received.</p> <p>0 The first byte has not been received.</p> <p>This bit is only valid when the RREQ bit is set and is automatically cleared when data has been read from the I2CSDR register.</p> <p>Note: This bit is not used for slave transmit operations.</p> |
| 1 | TREQ | RO | 0 | <p>Transmit Request</p> <p>Value Description</p> <p>1 The I²C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the I2CSDR register.</p> <p>0 No outstanding transmit request.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-----------------|
| 0 | RREQ | RO | 0 | Receive Request |

Value Description

| | |
|---|---|
| 1 | The I ² C controller has outstanding receive data from the I ² C master and is using clock stretching to delay the master until the data has been read from the I2CSDR register. |
| 0 | No outstanding receive data. |

Write-Only Control Register**I2C Slave Control/Status (I2CSCSR)**

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x004

Type WO, 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 | RO |
| Reset | 0 | 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 | | | | | | | | | | | | | | | | DA |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-------------|
|-----------|------|------|-------|-------------|

| | | | | |
|------|----------|----|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
|------|----------|----|------------|---|

| | | | | |
|---|----|----|---|---------------|
| 0 | DA | WO | 0 | Device Active |
|---|----|----|---|---------------|

Value Description

| | |
|---|--|
| 0 | Disables the I ² C slave operation. |
| 1 | Enables the I ² C slave operation. |

Register 12: I²C Slave Data (I2CSDR), offset 0x008

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 Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x008

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | DATA | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DATA | R/W | 0x00 | 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 0x00C

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Slave Interrupt Mask (I2CSIMR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x00C

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 | | | | | | | | | | | | | STOPIM | STARTIM | DATAIM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:3 | 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 | STOPIM | RO | 0 | Stop Condition Interrupt Mask Value Description 1 The STOP condition interrupt is sent to the interrupt controller when the <code>STOPRIS</code> bit in the <code>I2CSRIS</code> register is set. 0 The <code>STOPRIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 1 | STARTIM | RO | 0 | Start Condition Interrupt Mask Value Description 1 The START condition interrupt is sent to the interrupt controller when the <code>STARTRIS</code> bit in the <code>I2CSRIS</code> register is set. 0 The <code>STARTRIS</code> interrupt is suppressed and not sent to the interrupt controller. |
| 0 | DATAIM | R/W | 0 | Data Interrupt Mask Value Description 1 The data received or data requested interrupt is sent to the interrupt controller when the <code>DATARIS</code> bit in the <code>I2CSRIS</code> register is set. 0 The <code>DATARIS</code> interrupt is suppressed and not sent to the interrupt controller. |

Register 14: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x010

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|---------|----------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | STOPRIS | STARTRIS | DATARIS |
| 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 |
|-----------|----------|------|------------|--|
| 31:3 | 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 | STOPRIS | RO | 0 | <p>Stop Condition Raw Interrupt Status</p> <p>Value Description</p> <p>1 A STOP condition interrupt is pending.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register.</p> |
| 1 | STARTRIS | RO | 0 | <p>Start Condition Raw Interrupt Status</p> <p>Value Description</p> <p>1 A START condition interrupt is pending.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register.</p> |
| 0 | DATARIS | RO | 0 | <p>Data Raw Interrupt Status</p> <p>Value Description</p> <p>1 A data received or data requested interrupt is pending.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register.</p> |

Register 15: I²C Slave Masked Interrupt Status (I2CSMIS), offset 0x014

This register specifies whether an interrupt was signaled.

I2C Slave Masked Interrupt Status (I2CSMIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x014

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|---------|----------|---------|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | | | | STOPMIS | STARTMIS | DATAMIS | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | STOPMIS | R/W | 0 | <p>Stop Condition Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked STOP condition interrupt was signaled is pending.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register.</p> |
| 1 | STARTMIS | R/W | 0 | <p>Start Condition Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked START condition interrupt was signaled is pending.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register.</p> |
| 0 | DATAMIS | RO | 0 | <p>Data Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked data received or data requested interrupt was signaled is pending.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register.</p> |

Register 16: I²C Slave Interrupt Clear (I2CSICR), offset 0x018

This register clears the raw interrupt. A read of this register returns no meaningful data.

I2C Slave Interrupt Clear (I2CSICR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x018

Type WO, 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 | | | | | | | | | | | | | STOPIC | STARTIC | DATAIC | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | STOPIC | WO | 0 | Stop Condition Interrupt Clear Writing a 1 to this bit clears the <i>STOPRIS</i> bit in the I2CSRIS register and the <i>STOPMIS</i> bit in the I2CSMIS register. A read of this register returns no meaningful data. |
| 1 | STARTIC | WO | 0 | Start Condition Interrupt Clear Writing a 1 to this bit clears the <i>STOPRIS</i> bit in the I2CSRIS register and the <i>STOPMIS</i> bit in the I2CSMIS register. A read of this register returns no meaningful data. |
| 0 | DATAIC | WO | 0 | Data Interrupt Clear Writing a 1 to this bit clears the <i>STOPRIS</i> bit in the I2CSRIS register and the <i>STOPMIS</i> bit in the I2CSMIS register. A read of this register returns no meaningful data. |

18 Inter-Integrated Circuit Sound (I²S) Interface

The I²S module is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I²S as well as Left-Justified and Right-Justified serial audio formats. Data can be in one of four modes: Stereo, Mono, Compact 16-bit Stereo and Compact 8-Bit Stereo.

The transmit and receive modules each have an 8-entry audio-sample FIFO. An audio sample can consist of a Left and Right Stereo sample, a Mono sample, or a Left and Right Compact Stereo sample. In Compact 16-Bit Stereo, each FIFO entry contains both the 16-bit left and 16-bit right samples, allowing efficient data transfers and requiring less memory space. In Compact 8-bit Stereo, each FIFO entry contains an 8-bit left and an 8-bit right sample, reducing memory requirements further.

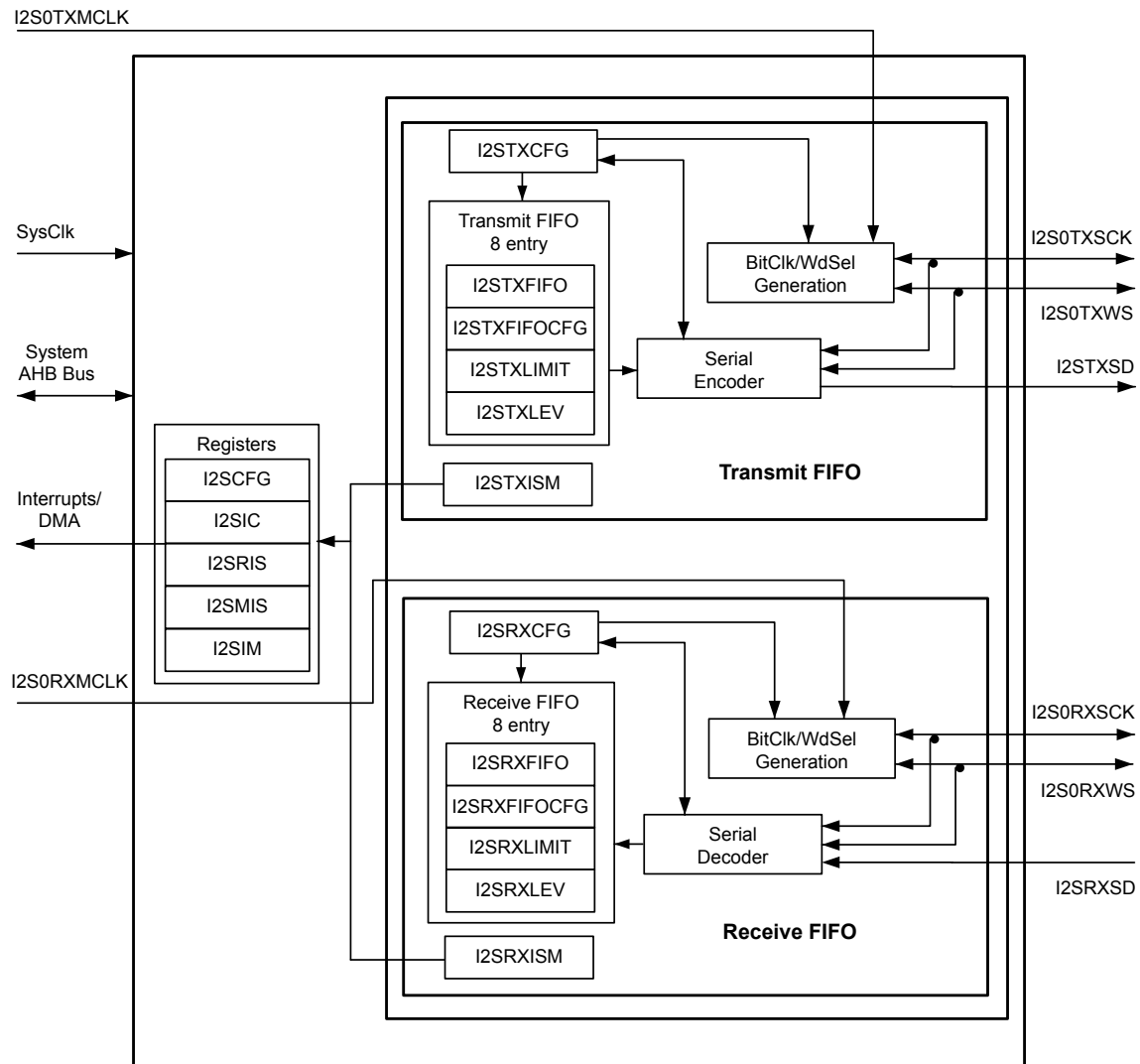
Both the transmitter and receiver are capable of being a master or a slave.

The Stellaris® I²S module has the following features:

- Configurable audio format supporting I²S, Left-justification, and Right-justification
- Configurable sample size from 8 to 32 bits
- Mono and Stereo support
- 8-, 16-, and 32-bit FIFO interface for packing memory
- Independent transmit and receive 8-entry FIFOs
- Configurable FIFO-level interrupt and μ DMA requests
- Independent transmit and receive MCLK direction control
- Transmit and receive internal MCLK sources
- Independent transmit and receive control for serial clock and word select
- MCLK and SCLK can be independently set to master or slave
- Configurable transmit zero or last sample when FIFO empty
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Burst requests
 - Channel requests asserted when FIFO contains required amount of data

18.1 Block Diagram

Figure 18-1. I²S Block Diagram



18.2 Signal Description

Table 18-1 on page 715 lists the external signals of the I²S module and describes the function of each. The I²S module signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²S signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the I²S function. The number in parentheses is the encoding that must be programmed into the PMC_n field in the **GPIO Port Control (GPIOCTL)** register (page 358) to assign the I²S signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318.

Table 18-1. Signals for I2S

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|------------|----------------|-------------------------------|----------|--------------------------|--|
| I2S0RXMCLK | 29 98 | PA3 (9) PD5 (8) | I/O | TTL | I ² S module 0 receive master clock. |
| I2S0RXSCK | 10 | PD0 (8) | I/O | TTL | I ² S module 0 receive clock. |
| I2S0RXSD | 28 97 | PA2 (9) PD4 (8) | I/O | TTL | I ² S module 0 receive data. |
| I2S0RXWS | 11 | PD1 (8) | I/O | TTL | I ² S module 0 receive word select. |
| I2S0TXMCLK | 61 | PF1 (8) | I/O | TTL | I ² S module 0 transmit master clock. |
| I2S0TXSCK | 30 90 99 | PA4 (9) PB6 (9) PD6 (8) | I/O | TTL | I ² S module 0 transmit clock. |
| I2S0TXSD | 5 47 | PE5 (9) PF0 (8) | I/O | TTL | I ² S module 0 transmit data. |
| I2S0TXWS | 6 31 100 | PE4 (9) PA5 (9) PD7 (8) | I/O | TTL | I ² S module 0 transmit word select. |

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

18.3 Functional Description

The Inter-Integrated Circuit Sound (I²S) module contains separate transmit and receive engines. Each engine consists of the following:

- Serial encoder for the transmitter; serial decoder for the receiver
- 8-entry FIFO to store sample data
- Independent configuration of all programmable settings

The basic programming model of the I²S block is as follows:

- Configuration
 - Overall I²S module configuration in the **I²S Module Configuration (I2SCFG)** register. This register is used to select the MCLK source and enable the receiver and transmitter.
 - Transmit and receive configuration in the **I²S Transmit Module Configuration (I2STXCFG)** and **I²S Receive Module Configuration (I2SRXCFG)** registers. These registers set the basic parameters for the receiver and transmitter such as data configuration (justification, delay, read mode, sample size, and system data size); SCLK (polarity and source); and word select polarity.
 - Transmit and receive FIFO configuration in the **I²S Transmit FIFO Configuration (I2STXFIFOCFG)** and **I²S Receive FIFO Configuration (I2SRXFIFOCFG)** registers. These registers select the Compact Stereo mode size (16-bit or 8-bit), provide indication of whether the next sample is Left or Right, and select mono mode for the receiver.
- FIFO
 - Transmit and receive FIFO data in the **I²S Transmit FIFO Data (I2STXFIFO)** and **I²S Receive FIFO Data (I2SRXFIFO)** registers

- Information on FIFO data levels in the **I²S Transmit FIFO Level (I2STXLEV)** and **I²S Receive FIFO Level (I2SRXLEV)** registers
- Configuration for FIFO service requests based on FIFO levels in the **I²S Transmit FIFO Limit (I2STXLIMIT)** and **I²S Receive FIFO Limit (I2SRXLIM)** registers
- **Interrupt Control**
 - Interrupt masking configuration in the **I²S Interrupt Mask (I2SIM)** register
 - Raw and masked interrupt status in the **I²S Raw Interrupt Status (I2SRIS)** and **I²S Masked Interrupt Status (I2SMIS)** registers
 - Interrupt clearing through the **I²S Interrupt Clear (I2SIC)** register
 - Configuration for FIFO service requests interrupts and transmit/receive error interrupts in the **I²S Transmit Interrupt Status and Mask (I2STXISM)** and **I²S Receive Interrupt Status and Mask (I2SRXISM)** registers

Figure 18-2 on page 716 provides an example of an I²S data transfer. Figure 18-3 on page 716 provides an example of an Left-Justified data transfer. Figure 18-4 on page 717 provides an example of an Right-Justified data transfer.

Figure 18-2. I²S Data Transfer

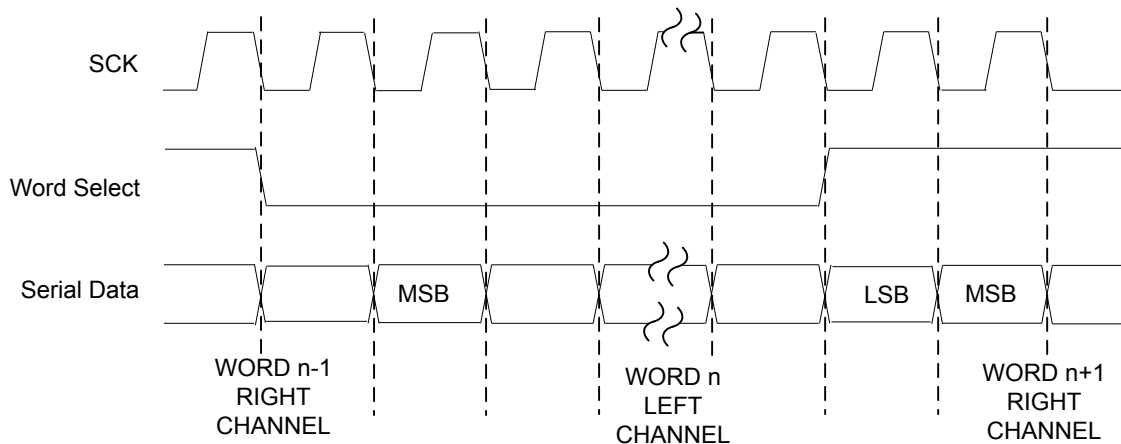


Figure 18-3. Left-Justified Data Transfer

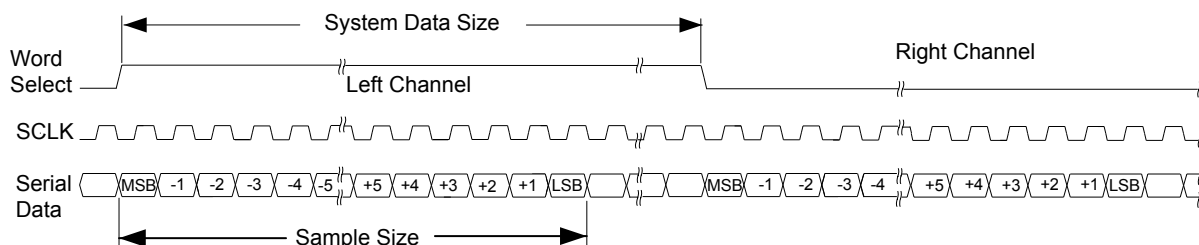
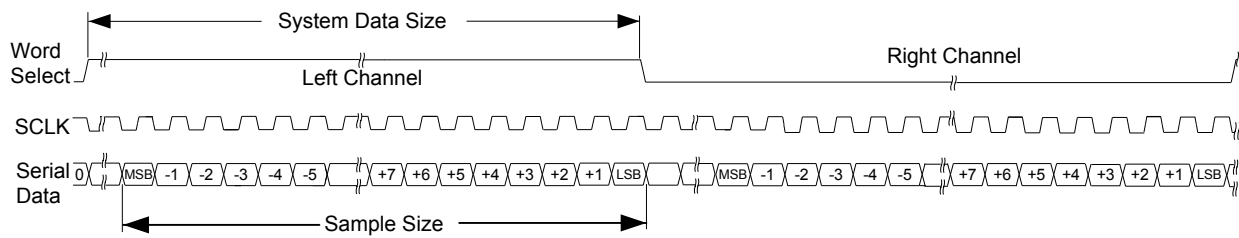


Figure 18-4. Right-Justified Data Transfer



18.3.1 Transmit

The transmitter consists of a serial encoder, an 8-entry FIFO, and control logic. The transmitter has independent MCLK ($I2S0TXMCLK$), SCLK ($I2S0TXSCK$), and Word-Select ($I2S0TXWS$) signals.

18.3.1.1 Serial Encoder

The serial encoder reads audio samples from the receive FIFO and converts them into an audio stream. By configuring the serial encoder, common audio formats I²S, Left-Justified, and Right-Justified are supported. The MSB is transmitted first. The sample size and system data size are configurable with the *SSZ* and *SDSZ* bits in the **I²S Transmit Module Configuration (I2STXCFG)** register. The sample size is the number of bits of data being transmitted, and the system data size is the number of $I2S0TXSCK$ transitions between the word select transitions. The system data size must be large enough to accommodate the maximum sample size. In Mono mode, the sample data is repeated in both the left and right channels. When the FIFO is empty, the user may select either transmission of zeros or of the last sample. The serial encoder is enabled using the *TXEN* bit in the **I²S Module Configuration (I2SCFG)** register.

18.3.1.2 FIFO Operation

The transmit FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I²S Transmit FIFO Data (I2STXFIFO)** register. The FIFO interface for the audio data is different based on the Write mode, defined by the **I²S Transmit FIFO Configuration (I2STXFIFOCFG)** Compact Stereo Sample Size bit (*CSS*) and the **I2STXCFG** Write Mode field (*WM*). All data samples are MSB-aligned. Table 18-2 on page 717 defines the interface for each Write mode. Stereo samples are written first left then right. The next sample (right or left) to be written is indicated by the *LRS* bit in the **I2STXFIFOCFG** register.

Table 18-2. I²S Transmit FIFO Interface

| <i>WM</i> field in I2STXCFG | <i>CSS</i> bit in I2STXFIFOCFG | Write Mode | Sample Width | Samples per FIFO Write | Data Alignment |
|-----------------------------|--------------------------------|-------------------------|--------------|------------------------|--------------------------------|
| 0x0 | don't care | Stereo | 8-32 bits | 1 | MSB |
| 0x1 | 0 | Compact Stereo - 16 bit | 8-16 bits | 2 | MSB Right [31:16], Left [15:0] |
| 0x1 | 1 | Compact Stereo - 8 bit | 8 bits | 2 | Right [15:8], Left[7:0] |
| 0x2 | don't care | Mono | 8-32 bits | 1 | MSB |

The number of samples in the transmit FIFO can be read using the **I²S Transmit FIFO Level (I2STXLEV)** register. The value ranges from 0 to 16. Stereo and compact stereo sample pairs are counted as two. The mono samples also increment the count by two, therefore, four mono samples will have a count of eight.

18.3.1.3 Clock Control

The transmitter MCLK and SCLK can be independently programmed to be the master or slave. The transmitter is programmed to be the master or slave of the SCLK using the `MSL` bit in the `I2STXCFG` register. When the transmitter is the master, the `I2S0TXSCK` frequency is the specified `I2S0TXMCLK` divided by four. The `I2S0TXSCK` may be inverted using the `SCP` bit in the `I2STXCFG` register.

The transmitter can also be the master or slave of the MCLK. When the transmitter is the master, the PLL must be active and a fractional clock divider must be programmed. See page 139 for the setup for the master `I2S0TXMCLK` source. An external transmit `I2S0TXMCLK` does not require the use of the PLL and is selected using the `TXSLV` bit in the `I2SCFG` register.

The following tables show combinations of the `TXINT` and `TXFRAC` bits in the **I²S MCLK Configuration (I2SMCLKCFG)** register that provide MCLK frequencies within acceptable error limits. In the table, `Fs` is the sampling frequency in kHz and possible crystal frequencies are shown in MHz across the top row of the table. The words "not supported" in the table mean that it is not possible to obtain the specified sampling frequencies with the specified crystal frequency within the error tolerance of 0.3%. The values in the table are based on the following values:

$$\text{MCLK} = F_s \times 256 \text{ PLL} = 400 \text{ MHz}$$

The Integer value is taken from the result of the following calculation:

$$\text{ROUND}(\text{PLL}/\text{MCLK})$$

The remaining fractional component is converted to binary, and the first four bits are the Fractional value.

Table 18-3. Crystal Frequency (Values from 3.5795 MHz to 5 MHz)

| Sampling Frequency F _s (kHz) | Crystal Frequency (MHz) | | | | | | | | | | | |
|--|-------------------------|------------|---------------|------------|---------|------------|---------|------------|---------------|------------|---------|------------|
| | 3.5795 | | 3.6864 | | 4 | | 4.096 | | 4.9152 | | 5 | |
| | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional |
| 8 | 195 | 12 | 194 | 6 | 195 | 5 | 196 | 0 | 194 | 6 | 195 | 5 |
| 11.025 | 142 | 1 | 141 | 1 | 141 | 12 | 142 | 4 | 141 | 1 | 141 | 12 |
| 12 | 130 | 8 | 129 | 10 | 130 | 3 | 130 | 11 | 129 | 10 | 130 | 3 |
| 16 | 97 | 14 | 97 | 3 | 97 | 10 | 98 | 0 | 97 | 3 | 97 | 10 |
| 22.05 | 71 | 0 | 70 | 8 | 70 | 14 | 71 | 2 | 70 | 8 | 70 | 14 |
| 24 | 65 | 4 | 64 | 13 | 65 | 2 | 65 | 5 | 64 | 13 | 65 | 2 |
| 32 | 48 | 15 | 48 | 10 | 48 | 13 | 49 | 0 | 48 | 10 | 48 | 13 |
| 44.1 | 35 | 8 | 35 | 4 | 35 | 7 | 35 | 9 | 35 | 4 | 35 | 7 |
| 48 | 32 | 10 | 32 | 6 | 32 | 9 | 32 | 11 | 32 | 6 | 32 | 9 |
| 64 | 24 | 8 | 24 | 5 | 24 | 7 | 24 | 8 | 24 | 5 | 24 | 7 |
| 88.2 | 17 | 12 | 17 | 10 | 17 | 11 | 17 | 12 | 17 | 10 | 17 | 11 |
| 96 | 16 | 5 | 16 | 3 | 16 | 4 | 16 | 5 | 16 | 3 | 16 | 4 |
| 128 | 12 | 4 | 12 | 2 | 12 | 3 | 12 | 4 | 12 | 2 | 12 | 3 |
| 176.4 | 8 | 14 | 8 | 13 | 8 | 14 | 8 | 14 | 8 | 13 | 8 | 14 |
| 192 | Not supported | | Not supported | | 8 | 2 | 8 | 3 | Not supported | | 8 | 2 |

Table 18-4. Crystal Frequency (Values from 5.12 MHz to 8.192 MHz)

| Sampling Frequency Fs (kHz) | Crystal Frequency (MHz) | | | | | | | | | | | |
|--------------------------------|-------------------------|------------|---------|------------|---------------|------------|---------------|------------|---------|------------|---------|------------|
| | 5.12 | | 6 | | 6.144 | | 7.3728 | | 8 | | 8.192 | |
| | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional |
| 8 | 195 | 0 | 195 | 5 | 195 | 0 | 194 | 6 | 195 | 5 | 194 | 11 |
| 11.025 | 141 | 8 | 141 | 12 | 141 | 8 | 141 | 1 | 141 | 12 | 141 | 4 |
| 12 | 130 | 0 | 130 | 3 | 130 | 0 | 129 | 10 | 130 | 3 | 129 | 12 |
| 16 | 97 | 8 | 97 | 10 | 97 | 8 | 97 | 3 | 97 | 10 | 97 | 5 |
| 22.05 | 70 | 12 | 70 | 14 | 70 | 12 | 70 | 8 | 70 | 14 | 70 | 10 |
| 24 | 65 | 0 | 65 | 2 | 65 | 0 | 64 | 13 | 65 | 2 | 64 | 14 |
| 32 | 48 | 12 | 48 | 13 | 48 | 12 | 48 | 10 | 48 | 13 | 48 | 11 |
| 44.1 | 35 | 6 | 35 | 7 | 35 | 6 | 35 | 4 | 35 | 7 | 35 | 5 |
| 48 | 32 | 8 | 32 | 9 | 32 | 8 | 32 | 6 | 32 | 9 | 32 | 7 |
| 64 | 24 | 6 | 24 | 7 | 24 | 6 | 24 | 5 | 24 | 7 | 24 | 5 |
| 88.2 | 17 | 11 | 17 | 11 | 17 | 11 | 17 | 10 | 17 | 11 | 17 | 11 |
| 96 | 16 | 4 | 16 | 4 | 16 | 4 | 16 | 3 | 16 | 4 | 16 | 4 |
| 128 | 12 | 3 | 12 | 3 | 12 | 3 | 12 | 2 | 12 | 3 | 12 | 3 |
| 176.4 | Not supported | | 8 | 14 | Not supported | | 8 | 13 | 8 | 14 | 8 | 13 |
| 192 | 8 | 2 | 8 | 2 | 8 | 2 | Not supported | | 8 | 2 | 8 | 2 |

Table 18-5. Crystal Frequency (Values from 10 MHz to 14.3181 MHz)

| Sampling Frequency Fs (kHz) | Crystal Frequency (MHz) | | | | | | | | | |
|--------------------------------|-------------------------|------------|---------|------------|---------|------------|---------------|------------|---------------|------------|
| | 10 | | 12 | | 12.288 | | 13.56 | | 14.3181 | |
| | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional | Integer | Fractional |
| 8 | 195 | 5 | 195 | 5 | 196 | 0 | 194 | 3 | 195 | 12 |
| 11.025 | 141 | 12 | 141 | 12 | 142 | 4 | 140 | 15 | 142 | 1 |
| 12 | 130 | 3 | 130 | 3 | 130 | 11 | 129 | 8 | 130 | 8 |
| 16 | 97 | 10 | 97 | 10 | 98 | 0 | 97 | 2 | 97 | 14 |
| 22.05 | 70 | 14 | 70 | 14 | 71 | 2 | 70f | 7 | 71 | 0 |
| 24 | 65 | 2 | 65 | 2 | 65 | 5 | 64 | 12 | 65 | 4 |
| 32 | 48 | 13 | 48 | 13 | 49 | 0 | 48 | 9 | 48 | 15 |
| 44.1 | 35 | 7 | 35 | 7 | 35 | 9 | 35 | 4 | 35 | 8 |
| 48 | 32 | 9 | 32 | 9 | 32 | 11 | 32 | 6 | 32 | 10 |
| 64 | 24 | 7 | 24 | 7 | 24 | 8 | 24 | 4 | 24 | 8 |
| 88.2 | 17 | 11 | 17 | 11 | 17 | 12 | 17 | 10 | 17 | 12 |
| 96 | 16 | 4 | 16 | 4 | 16 | 5 | 16 | 3 | 16 | 5 |
| 128 | 12 | 3 | 12 | 3 | 12 | 4 | 12 | 2 | 12 | 4 |
| 176.4 | 8 | 14 | 8 | 14 | 8 | 14 | 8 | 13 | 8 | 14 |
| 192 | 8 | 2 | 8 | 2 | 8 | 3 | Not supported | | Not supported | |

Table 18-6. Crystal Frequency (Values from 16 MHz to 16.384 MHz)

| Sampling Frequency Fs (kHz) | Crystal Frequency (MHz) | | | |
|--------------------------------|-------------------------|------------|---------|------------|
| | 16 | | 16.384 | |
| | Integer | Fractional | Integer | Fractional |
| 8 | 195 | 5 | 192 | 0 |
| 11.025 | 141 | 12 | 139 | 5 |
| 12 | 130 | 3 | 128 | 0 |
| 16 | 97 | 10 | 96 | 0 |
| 22.05 | 70 | 14 | 69 | 11 |
| 24 | 65 | 2 | 64 | 0 |
| 32 | 48 | 13 | 48 | 0 |
| 44.1 | 35 | 7 | 34 | 13 |
| 48 | 32 | 9 | 32 | 0 |
| 64 | 24 | 7 | 24 | 0 |
| 88.2 | 17 | 11 | 17 | 7 |
| 96 | 16 | 4 | 16 | 0 |
| 128 | 12 | 3 | 12 | 0 |
| 176.4 | 8 | 14 | 8 | 11 |
| 192 | 8 | 2 | 8 | 0 |

18.3.1.4 Interrupt Control

A single interrupt is asserted to the CPU whenever any of the transmit or receive sources is asserted. The transmit module has two interrupt sources: the FIFO service request and write error. The interrupts may be masked using the `TXSRIM` and `TXWEIM` bits in the **I²S Interrupt Mask (I2SIM)** register. The status of the interrupt source is indicated by the **I²S Raw Interrupt Status (I2SRIS)** register. The status of enabled interrupts is indicated by the **I²S Masked Interrupt Status (I2SMIS)** register. The FIFO level interrupt has a second level of masking using the `FFM` bit in the **I²S Transmit Interrupt Status and Mask (I2STXISM)** register.

The FIFO service request interrupt is asserted when the FIFO level (indicated by the `LEVEL` field in the **I²S Transmit FIFO Level (I2STXLEV)** register) is below the FIFO limit (programmed using the **I²S Transmit FIFO Limit (I2STXLIMIT)** register) and both the `TXSRIM` and `FFM` bits are set. If software attempts to write to a full FIFO, a Transmit FIFO Write error occurs (indicated by the `TXWERIS` bit in the **I²S Raw Interrupt Status (I2SRIS)** register). The `TXWERIS` bit in the **I2SRIS** register and the `TXWEMIS` bit in the **I2SMIS** register are cleared by setting the `TXWEIC` bit in the **I²S Interrupt Clear (I2SIC)** register.

18.3.1.5 DMA Support

The μ DMA can be used to more efficiently stream data to and from the I²S bus. The I²S transmit and receive modules have separate μ DMA channels. The FIFO Interrupt Mask bit (`FFM`) in the **I2STXISM** register must be set for the request signaling to propagate to the μ DMA module. See “Micro Direct Memory Access (μ DMA)” on page 258 for channel configuration.

The I²S module uses the μ DMA burst request signal, not the single request. Thus each time a μ DMA request is made, the μ DMA controller transfers the number of items specified as the burst size for the μ DMA channel. Therefore, the μ DMA channel burst size and the I²S FIFO service request limit must be set to the same value (using the `LIMIT` field in the **I2STXLIMIT** register).

18.3.2 Receive

The receiver consists of a serial decoder, an 8-entry FIFO, and control logic. The receiver has independent MCLK (I2S0RXMCLK), SCLK (I2S0RXSCK), and Word-Select (I2S0RXWS) signals.

18.3.2.1 Serial Decoder

The serial decoder accepts incoming audio stream data and places the sample data in the receive FIFO. By configuring the serial decoder, common audio formats I²S, Left-Justified, and Right-Justified are supported. The MSB is transmitted first. The sample size and system data size are configurable with the SSZ and SDSZ bits in the **I²S Receive Module Configuration (I2SRXCFG)** register. The sample size is the number of bits of data being received, and the system data size is the number of I2S0RXSCK transitions between the word select transitions. The system data size must be large enough to accommodate the maximum sample size. Any bits received after the LSB are 0s. If the FIFO is full, the incoming sample (in Mono) or sample-pairs (Stereo) are dropped until the FIFO has space. The serial decoder is enabled using the RXEN bit in the **I2SCFG** register.

18.3.2.2 FIFO Operation

The receive FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I²S Receive FIFO Data (I2SRXFIFO)** register. Table 18-7 on page 721 defines the interface for each Read mode. All data is stored MSB-aligned. The Stereo data is read left sample then right.

In Mono mode, the FIFO interface can be configured to read the right or left channel by setting the FIFO Mono Mode bit (FMM) in the **I²S Receive FIFO Configuration (I2SRXFIFOCFG)** register. This enables reads from a single channel, where the channel selected can be either the right or left as determined by the LRP bit in the **I2SRXCFG** register.

Table 18-7. I²S Receive FIFO Interface

| RM bit in I2RXCFG | CSS bit in I2SRXFIFOCFG | Read Mode | Sample Width | Samples per FIFO Write | Data Alignment |
|-------------------|-------------------------|--|--------------|------------------------|--------------------------------|
| 0 | don't care | Stereo | 8-32 bits | 1 | MSB |
| 1 | 0 | Compact Stereo - 16 bit | 8-16 bits | 2 | MSB Right [31:15], Left [15:0] |
| 1 | 1 | Compact Stereo - 8 bit | 8 bits | 2 | Right [15:8] Left[7:0] |
| 0 | don't care | Mono (FMM bit in the I2SRXFIFOCFG register must be set.) | 8-32 bits | 1 | MSB |

The number of samples in the receive FIFO can be read using the **I²S Receive FIFO Level (I2SRXLEV)** register. The value ranges from 0 to 16. Stereo and compact stereo sample pairs are counted as two. The mono samples also increment the count by two, therefore four Mono samples will have a count of eight.

18.3.2.3 Clock Control

The receiver MCLK and SCLK can be independently programmed to be the master or slave. The receiver is programmed to be the master or slave of the SCLK using the MSL bit in the **I2SRXCFG** register. When the receiver is the master, the I2S0RXSCK frequency is the specified I2S0RXMCLK divided by four. The I2S0RXSCK may be inverted using the SCP bit in the **I2SRXCFG** register.

The receiver can also be the master or slave of the MCLK. When the receiver is the master, the PLL must be active and a fractional clock divider must be programmed. See page 139 for the setup

for the master I2S0RXMCLK source. An external transmit I2S0RXMCLK does not require the use of the PLL and is selected using the RXSLV bit in the **I2SCFG** register.

Refer to “Clock Control” on page 718 for combinations of the RXINT and RXFRAC bits in the **I²S MCLK Configuration (I2SMCLKCFG)** register that provide MCLK frequencies within acceptable error limits. In the table, Fs is the sampling frequency in kHz and possible crystal frequencies are shown in MHz across the top row of the table. The words “not supported” in the table mean that it is not possible to obtain the specified sampling frequencies with the specified crystal frequency within the error tolerance of 0.3%.

18.3.2.4 Interrupt Control

A single interrupt is asserted to the CPU whenever any of the transmit or receive sources is asserted. The receive module has two interrupt sources: the FIFO service request and read error. The interrupts may be masked using the RXSRIM and RXREIM bits in the **I2SIM** register. The status of the interrupt source is indicated by the **I2SRIS** register. The status of enabled interrupts is indicated by the **I2SMIS** register. The FIFO service request interrupt has a second level of masking using the FFM bit in the **I²S Receive Interrupt Status and Mask (I2SRXISM)** register. The sources may be masked using the **I2SIM** register.

The FIFO service request interrupt is asserted when the FIFO level (indicated by the LEVEL field in the **I²S Receive FIFO Level (I2SRXLEV)** register) is above the FIFO limit (programmed using the **I²S Receive FIFO Limit (I2SRXLIMIT)** register) and both the RXSRIM and FFM bits are set. An error occurs when reading an empty FIFO or if a stereo sample pair is not read left then right. To clear an interrupt, write a 1 to the appropriate bit in the **I2SIC** register. If software attempts to read an empty FIFO or if a stereo sample pair is not read left then right, a Receive FIFO Read error occurs (indicated by the RXRERIS bit in the **I2SRIS** register). The RXRERIS bit in the **I2SRIS** register and the RXREMIS bit in the **I2SMIS** register are cleared by setting the RXREIC bit in the **I2SIC** register.

18.3.2.5 DMA Support

The μ DMA can be used to more efficiently stream data to and from the I²S bus. The I²S transmit and receive modules have separate μ DMA channels. The FIFO Interrupt Mask bit (FFM) in the **I2SRXISM** register must be set for the request signaling to propagate to the μ DMA module. See “Micro Direct Memory Access (μ DMA)” on page 258 for channel configuration.

The I²S module uses the μ DMA burst request signal, not the single request. Thus each time a μ DMA request is made, the μ DMA controller transfers the number of items specified as the burst size for the μ DMA channel. Therefore, the μ DMA channel burst size and the I²S FIFO service request limit must be set to the same value (using the LIMIT field in the **I2SRXLIMIT** register).

18.4 Initialization and Configuration

The default setup for the I²S transmit and receive is to use external MCLK, external SCLK, Stereo, I²S audio format, and 32-bit data samples. The following example shows how to configure a system using the internal MCLK, internal SCLK, Compact Stereo, and Left-Justified audio format with 16-bit data samples.

1. Enable the I²S peripheral clock by writing a value of 0x1000.0000 to the **RCGC1** register in the System Control module (see page 175).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 184). To find out which GPIO port to enable, refer to Table 24-5 on page 1035.

3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 340). To determine which GPIOs to configure, see Table 24-4 on page 1028.
4. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the I²S signals to the appropriate pins (see page 358 and Table 24-5 on page 1035).
5. Set up the MCLK sources for a 48-kHz sample rate. The input crystal is assumed to be 6 MHz for this example (internal source).
 - Enable the PLL by clearing the **PWRDWN** bit in the **RCC** register in the System Control module (see page 121).
 - Set the MCLK dividers and enable them by writing 0x0208.0208 to the **I2SMCLKCFG** register in the System Control module (see page 139).
 - Enable the MCLK internal sources by writing 0x8208.8208 to the **I2SMCLKCFG** register in the System Control module.

To allow an external MCLK to be used, set bits 4 and 5 of the **I2SCFG** register. Starting up the PLL and enabling the MCLK sources is not required.
6. Set up the Serial Bit Clock SCLK source. By default, the SCLK is externally sourced.
 - Receiver: Masters the **I2S0RXSCK** by ORing 0x0040.0000 into the **I2SRXCFG** register.
 - Transmitter: Masters the **I2S0TXSCK** by ORing 0x0040.0000 into the **I2STXCFG** register.
7. Configure the Serial Encoder/Decoder (Left-Justified, Compact Stereo, 16-bit samples, 32-bit system data size).
 - Set the audio format using the Justification (**JST**), Data Delay (**DLY**), SCLK polarity (**SCP**), and Left-Right Polarity (**LRP**) bits written to the **I2STXCFG** and **I2SRXCFG** registers. The settings are shown in the table below.

Table 18-8. Audio Formats Configuration

| Audio Format | I2STXCFG/I2SRXCFG Register Bit | | | |
|------------------|--------------------------------|-----|-----|-----|
| | JST | DLY | SCP | LRP |
| I ² S | 0 | 1 | 0 | 1 |
| Left-Justified | 0 | 0 | 0 | 0 |
| Right-Justified | 1 | 0 | 0 | 0 |

- Write 0x0140.3DF0 to both the **I2STXCFG** and **I2SRXCFG** registers to program the following configurations:
 - Set the sample size to 16 bits using the **SSZ** field of the **I2STXCFG** and **I2SRXCFG** registers.
 - Set the system data size to 32 bits using the **SDSZ** field of the **I2STXCFG** and **I2SRXCFG** registers.
 - Set the Write and Read modes using the **WM** and **RM** fields in the **I2STXCFG** and **I2SRXCFG** registers, respectively.

8. Set up the FIFO limits for triggering interrupts (also used for μ DMA)
 - Set up the transmit FIFO to trigger when it has less than four sample pairs by writing a 0x0000.0008 to the **I2STXLIMIT** register.
 - Set up the receive FIFO to trigger when there are more than four sample pairs by writing a 0x0000.00008 to the **I2SRXLIMIT** register.
9. Enable interrupts.
 - Enable the transmit FIFO interrupt by setting the **FFM** bit in the **I2STXISM** register (write 0x0000.0001).
 - Set up the receive FIFO interrupts by setting the **FFM** bit in the **I2SRXISM** register (write 0x0000.0001).
 - Enable the TX FIFO service request, the TX Error, the RX FIFO service request, and the RX Error interrupts to be sent to the CPU by writing a 0x0000.0033 to the **I2SSIM** register.
10. Enable the Serial Encoder and Serial Decoders by writing a 0x0000.0003 to the **I2SCFG** register.

18.5 Register Map

Table 18-9 on page 724 lists the I²S registers. The offset listed is a hexadecimal increment to the register's address, relative to the I²S interface base address of 0x4005.4000. Note that the I²S module clock must be enabled before the registers can be programmed (see page 175).

Table 18-9. Inter-Integrated Circuit Sound (I²S) Interface Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|--------------|------|-------------|--|----------|
| 0x000 | I2STXFIFO | WO | 0x0000.0000 | I2S Transmit FIFO Data | 726 |
| 0x004 | I2STXFIFOCFG | R/W | 0x0000.0000 | I2S Transmit FIFO Configuration | 727 |
| 0x008 | I2STXCFCG | R/W | 0x1400.7DF0 | I2S Transmit Module Configuration | 728 |
| 0x00C | I2STXLIMIT | R/W | 0x0000.0000 | I2S Transmit FIFO Limit | 730 |
| 0x010 | I2STXISM | R/W | 0x0000.0000 | I2S Transmit Interrupt Status and Mask | 731 |
| 0x018 | I2STXLEV | RO | 0x0000.0000 | I2S Transmit FIFO Level | 732 |
| 0x800 | I2SRXFIFO | RO | 0x0000.0000 | I2S Receive FIFO Data | 733 |
| 0x804 | I2SRXFIFOCFG | R/W | 0x0000.0000 | I2S Receive FIFO Configuration | 734 |
| 0x808 | I2SRXCFCG | R/W | 0x1400.7DF0 | I2S Receive Module Configuration | 735 |
| 0x80C | I2SRXLIMIT | R/W | 0x0000.7FFF | I2S Receive FIFO Limit | 738 |
| 0x810 | I2SRXISM | R/W | 0x0000.0000 | I2S Receive Interrupt Status and Mask | 739 |
| 0x818 | I2SRXLEV | RO | 0x0000.0000 | I2S Receive FIFO Level | 740 |
| 0xC00 | I2SCFG | R/W | 0x0000.0000 | I2S Module Configuration | 741 |
| 0xC10 | I2SIM | R/W | 0x0000.0000 | I2S Interrupt Mask | 743 |
| 0xC14 | I2SRIS | RO | 0x0000.0000 | I2S Raw Interrupt Status | 745 |

Table 18-9. Inter-Integrated Circuit Sound (I²S) Interface Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|--------|------|-------------|-----------------------------|----------|
| 0xC18 | I2SMIS | RO | 0x0000.0000 | I2S Masked Interrupt Status | 747 |
| 0xC1C | I2SIC | WO | 0x0000.0000 | I2S Interrupt Clear | 749 |

18.6 Register Descriptions

The remainder of this section lists and describes the I²S registers, in numerical order by address offset.

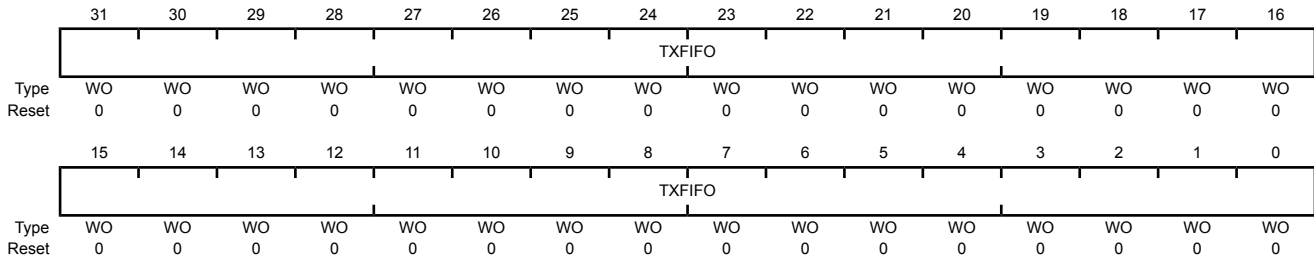
Register 1: I²S Transmit FIFO Data (I2STXFIFO), offset 0x000

This register is the 32-bit serial audio transmit data register. In Stereo mode, the data is written left, right, left, right, and so on. The *LRS* bit in the **I²S Transmit FIFO Configuration (I2STXFIFOCFG)** register can be read to verify the next position expected. In Compact 16-bit mode, bits [31:16] contain the right sample, and bits [15:0] contain the left sample. In Compact 8-bit mode, bits [15:8] contain the right sample, and bits [7:0] contain the left sample. In Mono mode, each 32-bit entry is a single sample.

Note that if the FIFO is full and a write is attempted, a transmit FIFO write error is generated.

I2S Transmit FIFO Data (I2STXFIFO)

Base 0x4005.4000
 Offset 0x000
 Type WO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------------|--|
| 31:0 | TXFIFO | WO | 0x0000.0000 | TX Data Serial audio sample data to be transmitted. |

Register 2: I²S Transmit FIFO Configuration (I2STXFIFOCFG), offset 0x004

This register configures the sample for dual-channel operation. In Stereo mode, the LRS bit toggles between left and right samples as the Transmit FIFO is written. The left sample is written first, followed by the right.

I2S Transmit FIFO Configuration (I2STXFIFOCFG)

Base 0x4005.4000
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 | | | | | | | | | | | | | | CSS | LRS |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| 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 | CSS | R/W | 0 | Compact Stereo Sample Size Value Description 0 The transmitter is in Compact 16-bit Stereo Mode with a 16-bit sample size. 1 The transmitter is in Compact 8-bit Stereo Mode with an 8-bit sample size. |
| 0 | LRS | R/W | 0 | Left-Right Sample Indicator Value Description 0 The left sample is the next position. 1 The right sample is the next position. In Mono mode and Compact stereo mode, this bit toggles as if it were in Stereo mode, but it has no meaning and should be ignored. |

Register 3: I²S Transmit Module Configuration (I2STXCFG), offset 0x008

This register controls the configuration of the Transmit module.

I2S Transmit Module Configuration (I2STXCFG)

Base 0x4005.4000
 Offset 0x008
 Type R/W, reset 0x1400.7DF0

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|------|-----|-----|-----|-----|-----|----------|----------|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | JST | DLY | SCP | LRP | WM | | FMT | MSL | reserved | | | | | |
| Type | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 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 |
| | SSZ | | | | SDSZ | | | | | | | reserved | | | | |
| 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 | RO | RO | RO | RO |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31:30 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 29 | JST | R/W | 0 | Justification of Output Data Value Description 0 The data is Left-Justified. 1 The data is Right-Justified. |
| 28 | DLY | R/W | 1 | Data Delay Value Description 0 Data is latched on the next latching edge of I2S0TXSCK as defined by the SCP bit. This bit should be clear in Left-Justified or Right-Justified mode. 1 A one-I2S0TXSCK delay from the edge of I2S0TXWS is inserted before data is latched. This bit should be set in I ² S mode. |
| 27 | SCP | R/W | 0 | SCLK Polarity Value Description 0 Data and the I2S0TXWS signal (when the MSL bit is set) are launched on the falling edge of I2S0TXSCK. 1 Data and the I2S0TXWS signal (when the MSL bit is set) are launched on the rising edge of I2S0TXSCK. |
| 26 | LRP | R/W | 1 | Left/Right Clock Polarity Value Description 0 I2S0TXWS is high during the transmission of the left channel data. 1 I2S0TXWS is high during the transmission of the right channel data. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 25:24 | WM | R/W | 0x0 | <p>Write Mode</p> <p>This bit field selects the mode in which the transmit data is stored in the FIFO and transmitted.</p> <p>Value Description</p> <p>0x0 Stereo mode</p> <p>0x1 Compact Stereo mode</p> <p>Left/Right sample packed. Refer to I2STXFIFOCFG for 8/16-bit sample size selection.</p> <p>0x2 Mono mode</p> <p>0x3 reserved</p> |
| 23 | FMT | R/W | 0 | <p>FIFO Empty</p> <p>Value Description</p> <p>0 All zeroes are transmitted if the FIFO is empty.</p> <p>1 The last sample is transmitted if the FIFO is empty.</p> |
| 22 | MSL | R/W | 0 | <p>SCLK Master/Slave</p> <p>Source of serial bit clock (I2S0TXSCK) and Word Select (I2S0TXWS).</p> <p>Value Description</p> <p>0 The transmitter is a slave using the externally driven I2S0TXSCK and I2S0TXWS signals.</p> <p>1 The transmitter is a master using the internally generated I2S0TXSCK and I2S0TXWS signals.</p> |
| 21:16 | reserved | RO | 0x00 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |
| 15:10 | SSZ | R/W | 0x1F | <p>Sample Size</p> <p>This field contains the number of bits minus one in the sample.</p> <p>Note: This field is only used in Right-Justified mode. Unused bits are not masked.</p> |
| 9:4 | SDSZ | R/W | 0x1F | <p>System Data Size</p> <p>This field contains the number of bits minus one during the high or low phase of the I2S0TXWS signal.</p> |
| 3:0 | reserved | RO | 0x0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |

Register 4: I²S Transmit FIFO Limit (I2STXLIMIT), offset 0x00C

This register sets the lower FIFO limit at which a FIFO service request is issued.

I2S Transmit FIFO Limit (I2STXLIMIT)

Base 0x4005.4000
 Offset 0x00C
 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 | | | | | | | | | | | | LIMIT | | | |
| 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 |
|-----------|----------|------|-----------|--|
| 31:5 | 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. |
| 4:0 | LIMIT | R/W | 0x00 | <p>FIFO Limit</p> <p>This field sets the FIFO level at which a FIFO service request is issued, generating an interrupt or a μDMA transfer request.</p> <p>The transmit FIFO generates a service request when the number of items in the FIFO is less than the level specified by the <code>LIMIT</code> field. For example, if the <code>LIMIT</code> field is set to 8, then a service request is generated when there are less than 8 samples remaining in the transmit FIFO.</p> |

Register 5: I²S Transmit Interrupt Status and Mask (I2STXISM), offset 0x010

This register indicates the transmit interrupt status and interrupt masking control.

I2S Transmit Interrupt Status and Mask (I2STXISM)

Base 0x4005.4000

Offset 0x010

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | FFI |
| 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 | | | | | | | | | | | | | | | FFM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:17 | 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 | FFI | RO | 0 | Transmit FIFO Service Request Interrupt Value Description 0 The FIFO level is equal to or above the FIFO limit. 1 The FIFO level is below the FIFO limit. |
| 15:1 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | FFM | R/W | 0 | FIFO Interrupt Mask Value Description 0 The FIFO interrupt is masked and not sent to the CPU. 1 The FIFO interrupt is enabled to be sent to the interrupt controller. |

Register 6: I²S Transmit FIFO Level (I2STXLEV), offset 0x018

The number of samples in the transmit FIFO can be read using the **I2STXLEV** register. The value ranges from 0 to 16. Stereo and Compact Stereo sample-pairs are counted as two. Mono samples also increment the count by two. For example, the **LEVEL** field is set to eight if there are four Mono samples.

I2S Transmit FIFO Level (I2STXLEV)

Base 0x4005.4000
 Offset 0x018
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-------|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | LEVEL | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:5 | 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. |
| 4:0 | LEVEL | RO | 0x00 | Number of Audio Samples This field contains the number of samples in the FIFO. |

Register 7: I²S Receive FIFO Data (I2SRXFIFO), offset 0x800

This register is the 32-bit serial audio receive data register. In Stereo mode, the data is read left, right, left, right, and so on. The `LRS` bit in the **I²S Receive FIFO Configuration (I2SRXFIFOCFG)** register can be read to verify the next position expected. In Compact 16-bit mode, bits [31:16] contain the right sample, and bits [15:0] contain the left sample. In Compact 8-bit mode, bits [15:8] contain the right sample, and bits [7:0] contain the left sample. In Mono mode, each 32-bit entry is a single sample. If the FIFO is empty, a read of this register returns a value of 0x0000.0000 and generates a receive FIFO read error.

I2S Receive FIFO Data (I2SRXFIFO)

Base 0x4005.4000
Offset 0x800
Type RO, reset 0x0000.0000

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | RXFIFO | | | | | | | | | | | | | | | |
| 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 |
| | RXFIFO | | | | | | | | | | | | | | | |
| 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 | RXFIFO | RO | 0x0000.0000 | RX Data |
|------|--------|----|-------------|---------|

Serial audio sample data received.

The read of an empty FIFO returns a value of 0x0.

Register 8: I²S Receive FIFO Configuration (I2SRXFIFOCFG), offset 0x804

This register configures the sample for dual-channel operation. In Stereo mode, the LRS bit toggles between Left and Right as the samples are read from the receive FIFO. In Mono mode, both the left and right samples are stored in the FIFO. The FMM bit can be used to read only the left or right sample as determined by the LRP bit. In Compact Stereo 8- or 16-bit mode, both the left and right samples are read in one access from the FIFO.

I2S Receive FIFO Configuration (I2SRXFIFOCFG)

Base 0x4005.4000
 Offset 0x804
 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 | | | | | | | | | | | | | FMM | CSS | LRS |
| 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 |

| 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 | FMM | R/W | 0 | FIFO Mono Mode Value Description 0 The receiver is in Stereo Mode. 1 The receiver is in Mono mode. If the LRP bit in the I2SRXCFG register is clear, data is read while the I2SORXWS signal is low (Right Channel); if the LRP bit is set, data is read while the I2SORXWS signal is high (Left Channel). |
| 1 | CSS | R/W | 0 | Compact Stereo Sample Size Value Description 0 The receiver is in Compact 16-bit Stereo Mode with a 16-bit sample size. 1 The receiver is in Compact 8-bit Stereo Mode with a 8-bit sample size. |
| 0 | LRS | R/W | 0 | Left-Right Sample Indicator Value Description 0 The left sample is the next position to be read. 1 The right sample is the next position to be read. This bit is only meaningful in Compact Stereo Mode. |

Register 9: I²S Receive Module Configuration (I2SRXCFG), offset 0x808

This register controls the configuration of the receive module.

I2S Receive Module Configuration (I2SRXCFG)

Base 0x4005.4000

Offset 0x808

Type R/W, reset 0x1400.7DF0

| | | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|------|----------|-----|----------|-----|----------|----------|-----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | JST | DLY | SCP | LRP | reserved | RM | reserved | MSL | reserved | | | | | | |
| Type | RO | RO | R/W | R/W | R/W | R/W | RO | R/W | RO | R/W | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 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 | |
| | SSZ | | | | | SDSZ | | | | | | reserved | | | | | |
| 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 | RO | RO | RO | RO |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 31:30 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 29 | JST | R/W | 0 | Justification of Input Data Value Description 0 The data is Left-Justified. 1 The data is Right-Justified. |
| 28 | DLY | R/W | 1 | Data Delay Value Description 0 Data is latched on the next latching edge of I2S0RXSCK as defined by the SCP bit. This bit should be clear in Left-Justified or Right-Justified mode. 1 A one-I2S0RXSCK delay from the edge of I2S0RXWS is inserted before data is latched. This bit should be set in I ² S mode. |
| 27 | SCP | R/W | 0 | SCLK Polarity Value Description 0 Data is latched on the rising edge and the I2S0RXWS signal (when the MSL bit is set) is launched on the falling edge of I2S0RXSCK. 1 Data is latched on the falling edge and the I2S0RXWS signal (when the MSL bit is set) is launched on the rising edge of I2S0RXSCK. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 26 | LRP | R/W | 1 | <p>Left/Right Clock Polarity</p> <p>Value Description</p> <p>0 In Stereo mode, I2S0RXWS is high during the transmission of the left channel data.</p> <p>In Mono mode, data is read while the I2S0RXWS signal is low (Right Channel).</p> <p>1 In Stereo mode, I2S0RXWS is high during the transmission of the right channel data.</p> <p>In Mono mode, data is read while the I2S0RXWS signal is high (Left Channel).</p> |
| 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 | RM | R/W | 0 | <p>Read Mode</p> <p>This bit selects the mode in which the receive data is received and stored in the FIFO.</p> <p>Value Description</p> <p>0 Stereo/Mono mode</p> <p>I2SRXFIFOCFG FMM bit specifies Stereo or Mono FIFO read behavior.</p> <p>1 Compact Stereo mode</p> <p>Left/Right sample packed. Refer to I2SRXFIFOCFG for 8/16-bit sample size selection.</p> |
| 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 | MSL | R/W | 0 | <p>SCLK Master/Slave</p> <p>Value Description</p> <p>0 The receiver is a slave and uses the externally driven I2S0RXSCK and I2S0RXWS signals.</p> <p>1 The receiver is a master and uses the internally generated I2S0RXSCK and I2S0RXWS signals.</p> |
| 21:16 | 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 | SSZ | R/W | 0x1F | <p>Sample Size</p> <p>This field contains the number of bits minus one in the sample.</p> |
| 9:4 | SDSZ | R/W | 0x1F | <p>System Data Size</p> <p>This field contains the number of bits minus one during the high or low phase of the I2S0RXWS signal.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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 10: I²S Receive FIFO Limit (I2SRXLIMIT), offset 0x80C

This register sets the upper FIFO limit at which a FIFO service request is issued.

I2S Receive FIFO Limit (I2SRXLIMIT)

Base 0x4005.4000
 Offset 0x80C
 Type R/W, reset 0x0000.7FFF

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|-----|-------|-----|-----|-----|
| | 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 | | | | | | | | | | | | LIMIT | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | 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 |
|-----------|----------|------|--------|---|
| 31:16 | 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. |
| 15:5 | reserved | RO | 0x7FF | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4:0 | LIMIT | R/W | 0x1F | FIFO Limit This field sets the FIFO level at which a FIFO service request is issued, generating an interrupt or a μ DMA transfer request. The receive FIFO generates a service request when the number of items in the FIFO is greater than the level specified by the LIMIT field. For example, if the LIMIT field is set to 4, then a service request is generated when there are less than 4 samples remaining in the transmit FIFO. |

Register 11: I²S Receive Interrupt Status and Mask (I2SRXISM), offset 0x810

This register indicates the receive interrupt status and interrupt masking control.

I2S Receive Interrupt Status and Mask (I2SRXISM)

Base 0x4005.4000

Offset 0x810

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | FFI |
| 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 | | | | | | | | | | | | | | | FFM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:17 | 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 | FFI | RO | 0 | Receive FIFO Service Request Interrupt |
| | | | | Value Description |
| | | | | 0 The FIFO level is equal to or below the FIFO limit. |
| | | | | 1 The FIFO level is above the FIFO limit. |
| 15:1 | reserved | RO | 0x000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | FFM | R/W | 0 | FIFO Interrupt Mask |
| | | | | Value Description |
| | | | | 0 The FIFO interrupt is masked and not sent to the CPU. |
| | | | | 1 The FIFO interrupt is enabled to be sent to the interrupt controller. |

Register 12: I²S Receive FIFO Level (I2SRXLEV), offset 0x818

The number of samples in the receive FIFO can be read using the **I2SRXLEV** register. The value ranges from 0 to 16. Stereo and Compact Stereo sample pairs are counted as two. Mono samples also increment the count by two. For example, the LEVEL field is set to eight if there are four Mono samples.

I2S Receive FIFO Level (I2SRXLEV)

Base 0x4005.4000
 Offset 0x818
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|-------|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | LEVEL | | | |
| 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 |
|-----------|----------|------|-----------|---|
| 31:5 | 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. |
| 4:0 | LEVEL | RO | 0x00 | Number of Audio Samples This field contains the number of samples in the FIFO. |

Register 13: I²S Module Configuration (I2SCFG), offset 0xC00

This register enables the transmit and receive serial engines and sets the source of the I2S0TXMCLK and I2S0RXMCLK signals.

I2S Module Configuration (I2SCFG)

Base 0x4005.4000
Offset 0xC00
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 | | | | | | | | | | | RXSLV | TXSLV | reserved | | RXEN | TXEN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | 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 |
|-----------|----------|------|-----------|---|
| 31:6 | 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 | RXSLV | R/W | 0 | Use External I2S0RXMCLK Value Description 0 The receiver uses the internally generated MCLK as the I2S0RXMCLK signal. See "Clock Control" on page 718 for information on how to program the I2S0RXMCLK. 1 The receiver uses the externally driven I2S0RXMCLK signal. |
| 4 | TXSLV | R/W | 0 | Use External I2S0TXMCLK Value Description 0 The transmitter uses the internally generated MCLK as the I2S0TXMCLK signal. See "Clock Control" on page 718 for information on how to program the I2S0TXMCLK. 1 The transmitter uses the externally driven I2S0TXMCLK signal. |
| 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 | RXEN | R/W | 0 | Serial Receive Engine Enable Value Description 0 Disables the serial receive engine. 1 Enables the serial receive engine. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 0 | TXEN | R/W | 0 | Serial Transmit Engine Enable |
| | | | | Value Description |
| | | | | 0 Disables the serial transmit engine. |
| | | | | 1 Enables the serial transmit engine. |

Register 14: I²S Interrupt Mask (I2SIM), offset 0xC10

This register masks the interrupts to the CPU.

I2S Interrupt Mask (I2SIM)

Base 0x4005.4000
Offset 0xC10
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 | | | | | | | | | | | RXREIM | RXSRIM | reserved | | TXWEIM | TXSRIM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | RO | 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 |
|-----------|----------|------|-----------|---|
| 31:6 | 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 | RXREIM | R/W | 0 | Receive FIFO Read Error Value Description 0 The receive FIFO read error interrupt is masked and not sent to the CPU. 1 The receive FIFO read error is enabled to be sent to the interrupt controller. |
| 4 | RXSRIM | R/W | 0 | Receive FIFO Service Request Value Description 0 The receive FIFO service request interrupt is masked and not sent to the CPU. 1 The receive FIFO service request is enabled to be sent to the interrupt controller. |
| 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 | TXWEIM | R/W | 0 | Transmit FIFO Write Error Value Description 0 The transmit FIFO write error interrupt is masked and not sent to the CPU. 1 The transmit FIFO write error is enabled to be sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 0 | TXSRIM | R/W | 0 | Transmit FIFO Service Request |
| | | | | Value Description |
| | | | | 0 The transmit FIFO service request interrupt is masked and not sent to the CPU. |
| | | | | 1 The transmit FIFO service request is enabled to be sent to the interrupt controller. |

Register 15: I²S Raw Interrupt Status (I2SRIS), offset 0xC14

This register reads the unmasked interrupt status.

I2S Raw Interrupt Status (I2SRIS)

Base 0x4005.4000
Offset 0xC14
Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|---------|---------|----------|----|---------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | | RXRERIS | RXSRRIS | reserved | | TXWERIS | TXSRRIS |
| 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 |
|-----------|----------|------|-----------|---|
| 31:6 | 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 | RXRERIS | RO | 0 | Receive FIFO Read Error Value Description 1 A receive FIFO read error interrupt has occurred. 0 No interrupt This bit is cleared by setting the <code>RXREIC</code> bit in the <code>I2SIC</code> register. |
| 4 | RXSRRIS | RO | 0 | Receive FIFO Service Request Value Description 1 A receive FIFO service request interrupt has occurred. 0 No interrupt This bit is cleared when the level in the receive FIFO has risen to a value greater than the value programmed in the <code>LIMIT</code> field in the <code>I2SRXLIMIT</code> 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 | TXWERIS | RO | 0 | Transmit FIFO Write Error Value Description 1 A transmit FIFO write error interrupt has occurred. 0 No interrupt This bit is cleared by setting the <code>TXWEIC</code> bit in the <code>I2SIC</code> register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 0 | TXSRRIS | RO | 0 | Transmit FIFO Service Request Value Description 1 A transmit FIFO service request interrupt has occurred. 0 No interrupt This bit is cleared when the level in the transmit FIFO has fallen to a value less than the value programmed in the <code>LIMIT</code> field in the I2STXLIMIT register. |

Register 16: I²S Masked Interrupt Status (I2SMIS), offset 0xC18

This register reads the masked interrupt status. The mask is defined in the **I2SIM** register.

I2S Masked Interrupt Status (I2SMIS)

Base 0x4005.4000
Offset 0xC18
Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|---------|---------|----------|----|---------|---------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | | | | | | RXREMIS | RXSRMIS | reserved | | TXWEMIS | TXSRMIS |
| 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 |
|-----------|----------|------|-----------|---|
| 31:6 | 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 | RXREMIS | RO | 0 | Receive FIFO Read Error Value Description 1 An unmasked interrupt was signaled due to a receive FIFO read error. 0 An interrupt has not occurred or is masked. This bit is cleared by setting the RXREIC bit in the I2SIC register. |
| 4 | RXSRMIS | RO | 0 | Receive FIFO Service Request Value Description 1 An unmasked interrupt was signaled due to a receive FIFO service request. 0 An interrupt has not occurred or is masked. This bit is cleared when the level in the receive FIFO has risen to a value greater than the value programmed in the LIMIT field in the I2SRXLIMIT register. |
| 3:2 | reserved | RO | 0s0 | Software should not rely on the value of 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 | TXWEMIS | RO | 0 | Transmit FIFO Write Error Value Description 1 An unmasked interrupt was signaled due to a transmit FIFO write error. 0 An interrupt has not occurred or is masked. This bit is cleared by setting the TXWEIC bit in the I2SIC register. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 0 | TXSRMIS | RO | 0 | Transmit FIFO Service Request |
| | | | | Value Description |
| | | | | 1 An unmasked interrupt was signaled due to a transmit FIFO service request. |
| | | | | 0 An interrupt has not occurred or is masked. |
| | | | | This bit is cleared when the level in the transmit FIFO has fallen to a value less than the value programmed in the <code>LIMIT</code> field in the I2STXLIMIT register. |

Register 17: I²S Interrupt Clear (I2SIC), offset 0xC1C

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

I2S Interrupt Clear (I2SIC)

Base 0x4005.4000
Offset 0xC1C
Type WO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|--------|----------|----|----|--------|----------|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | RXREIC | reserved | | | TXWEIC | reserved |
| Type | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:6 | reserved | WO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 5 | RXREIC | WO | 0 | Receive FIFO Read Error Writing a 1 to this bit clears the <code>RXRERIS</code> bit in the I2CRIS register and the <code>RXREMIS</code> bit in the I2CMIS register. |
| 4:2 | reserved | WO | 0x0 | Software should not rely on the value of 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 | TXWEIC | WO | 0 | Transmit FIFO Write Error Writing a 1 to this bit clears the <code>TXWERIS</code> bit in the I2CRIS register and the <code>TXWEMIS</code> bit in the I2CMIS register. |
| 0 | 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. |

19 Controller Area Network (CAN) Module

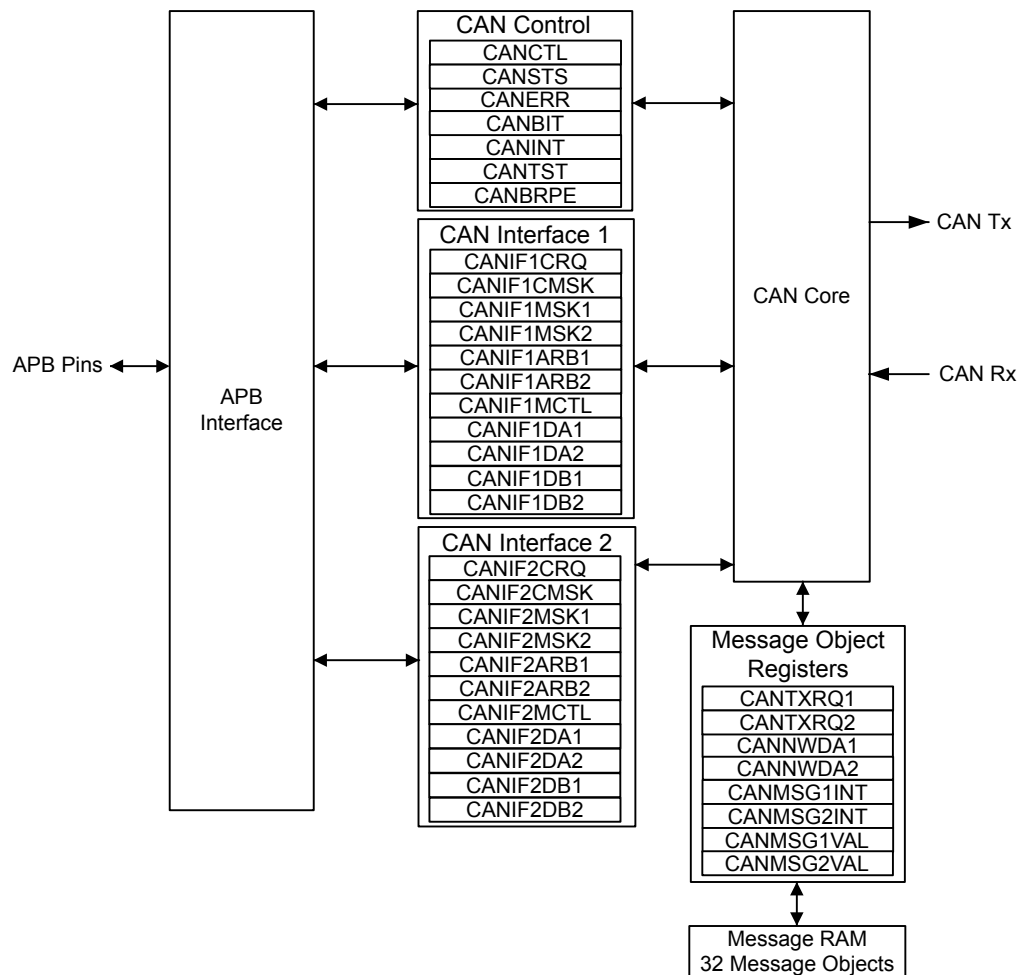
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally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

The Stellaris[®] LM3S9B90 microcontroller includes two CAN units with the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable Loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the CAN_nTX and CAN_nRX signals

19.1 Block Diagram

Figure 19-1. CAN Controller Block Diagram



19.2 Signal Description

Table 19-1 on page 752 lists the external signals of the CAN controller and describes the function of each. The CAN controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the CAN signals. The **AFSEL** bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the CAN controller function. The number in parentheses is the encoding that must be programmed into the **PMC_n** field in the **GPIO Port Control (GPIOCTL)** register (page 358) to assign the CAN signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318.

Table 19-1. Signals for Controller Area Network

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|----------------------|--|----------|--------------------------|------------------------|
| CAN0Rx | 10 30 34 92 | PD0 (2) PA4 (5) PA6 (6) PB4 (5) | I | TTL | CAN module 0 receive. |
| CAN0Tx | 11 31 35 91 | PD1 (2) PA5 (5) PA7 (6) PB5 (5) | O | TTL | CAN module 0 transmit. |
| CAN1Rx | 47 | PF0 (1) | I | TTL | CAN module 1 receive. |
| CAN1Tx | 61 | PF1 (1) | O | TTL | CAN module 1 transmit. |

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

19.3 Functional Description

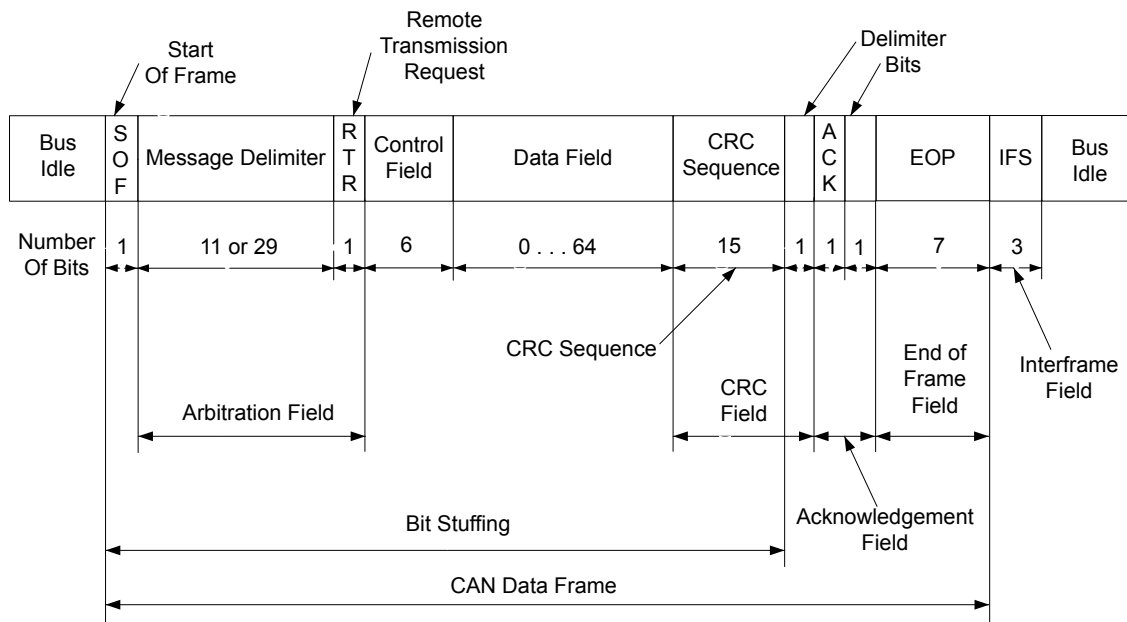
The Stellaris[®] CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

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 19-2.

Figure 19-2. CAN Data/Remote Frame



The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These memory blocks are accessed via either of the CAN message object register interfaces.

The message memory is not directly accessible in the 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. The message object memory cannot be directly accessed, so 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.

19.3.1 Initialization

To use the CAN controller, the peripheral clock must be enabled using the **RCGC0** register (see page 166). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register (see page 184). To find out which GPIO port to enable, refer to Table 24-4 on page 1028. Set the GPIO **AFSEL** bits for the appropriate pins (see page 340). Configure the **PMCn** fields in the **GPIOPCTL** register to assign the CAN signals to the appropriate pins. See page 358 and Table 24-5 on page 1035.

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.

19.3.2 Operation

Two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**) are used to access the message objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The two sets are independent and identical and can be used to

queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

Once the CAN module is initialized and the `INIT` bit in the **CANCTL** register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the message handler's filtering process, and if it passes through the filter, is stored in the message object specified by the `MNUM` bit in the **CAN IFn Command Request (CANIFnCRQ)** register. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the `MSK` bits in the **CAN IFn Mask 1** and **CAN IFn Mask 2 (CANIFnMSKn)** registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time via the CAN Interface Registers. The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects is under the control of the software that is managing the CAN hardware. Message objects can be used for one-time data transfers or can be permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. At the start of transmission, the appropriate `TXRQST` bit in the **CAN Transmission Request n (CANTXRQn)** register and the `NEWDAT` bit in the **CAN New Data n (CANNWDAn)** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier (`MNUM`) for the message object, with 1 being the highest priority and 32 being the lowest priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the `RMTEEN` bit in the **CAN IFn Message Control (CANIFnMCTL)** register. A matching received remote frame causes the `TXRQST` bit to be set, and the message object automatically transfers its data or generates an interrupt indicating a remote frame was requested. A remote frame can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are identified as remote frame requests. The `UMASK` bit in the **CANIFnMCTL** register enables the `MSK` bits in the **CANIFnMSKn** register to filter which frames are identified as a remote frame request. The `MXTD` bit in the **CANIFnMSK2** register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

19.3.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if a data transfer is not occurring between the CAN Interface Registers and message RAM, the valid message object with the highest priority that has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's `NEWDAT` bit in the **CANNWDAn** register is cleared. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the `TXRQST` bit in the **CANTXRQn** register is cleared. If the CAN controller is configured to interrupt on a successful transmission of a message object, (the `TXIE` bit in the **CAN IFn Message Control (CANIFnMCTL)** register is set), the `INTPND` bit in the **CANIFnMCTL** register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is

re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

19.3.4 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 on the reception of a matching remote frame allowing automatic transmission
 - Set the `EOB` bit for a single message object
 - Configure the `DLC[3:0]` field to specify the size of the data frame. Take care during this configuration not to set the `NEWDAT`, `MSGLST`, `INTPND` or `TXRQST` bits.
7. Load the data to be transmitted into the **CAN IFn Data (CANIFnDA1, CANIFnDA2, CANIFnDB1, CANIFnDB2)** registers. Byte 0 of the CAN data frame is stored in `DATA[7:0]` in the **CANIFnDA1** register.
 8. Program the number of the message object to be transmitted in the `MNUM` field in the **CAN IFn Command Request (CANIFnCRQ)** register.
 9. When everything is properly configured, set the `TXRQST` bit in the **CANIFnMCTL** register. Once this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Note that setting the `RMTEN` bit in the **CANIFnMCTL** register can also start message transmission if a matching remote frame has been received.

19.3.5 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.

19.3.6 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.

19.3.7 Receiving a Data Frame

The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the DLC bits are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used. The NEWDAT bit of the **CANIFnMCTL** register is set to indicate that new data has been received. The CPU should clear this bit when it reads the message object to indicate to the controller that the message has been received, and the buffer is free to receive more messages. If the CAN controller receives a message and the NEWDAT bit is already set, the MSGLST bit in the **CANIFnMCTL** register is set to indicate that the previous data was lost. If the system requires an interrupt on successful reception of a frame, the RXIE bit of the **CANIFnMCTL** register should be set. In this case, the INTPND bit of the same register is set, causing the **CANINT** register to point to the message object that just received a message. The TXRQST bit of this message object should be cleared to prevent the transmission of a remote frame.

19.3.8 Receiving a Remote Frame

A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object have to be considered:

Table 19-2. Message Object Configurations

| Configuration in CANIFnMCTL | Description |
|---|--|
| <ul style="list-style-type: none"> ■ 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. |
| <ul style="list-style-type: none"> ■ 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 nothing indicates that the remote frame ever happened. |
| <ul style="list-style-type: none"> ■ 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 mode 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. |

19.3.9 Receive/Transmit Priority

The receive/transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This prioritization is separate from that of the message identifier which is enforced by the CAN bus. As a result, if message object 1 and message object 2 both have valid messages to be transmitted, message object 1 is always transmitted first regardless of the message identifier in the message object itself.

19.3.10 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 755 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 755 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 755 section to program **XTD** and **ID** bits for the message identifier to be received; set the **MSGVAL** bit to indicate a valid message; and clear the **DIR** bit to specify receive.
5. In the **CANIFnMCTL** register:
 - Optionally set the **UMASK** bit to enable the mask (**MSK**, **MXTD**, and **MDIR** specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
 - Optionally set the **RXIE** bit to enable the **INTPND** bit to be set after a successful reception
 - Clear the **RMTEN** bit to leave the **TXRQST** bit unchanged
 - Set the **EOB** bit for a single message object
 - Configure the **DLC[3:0]** field to specify the size of the data frameTake 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.

19.3.11 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.

19.3.11.1 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 758). 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.

19.3.11.2 Reception of Messages with FIFO Buffers

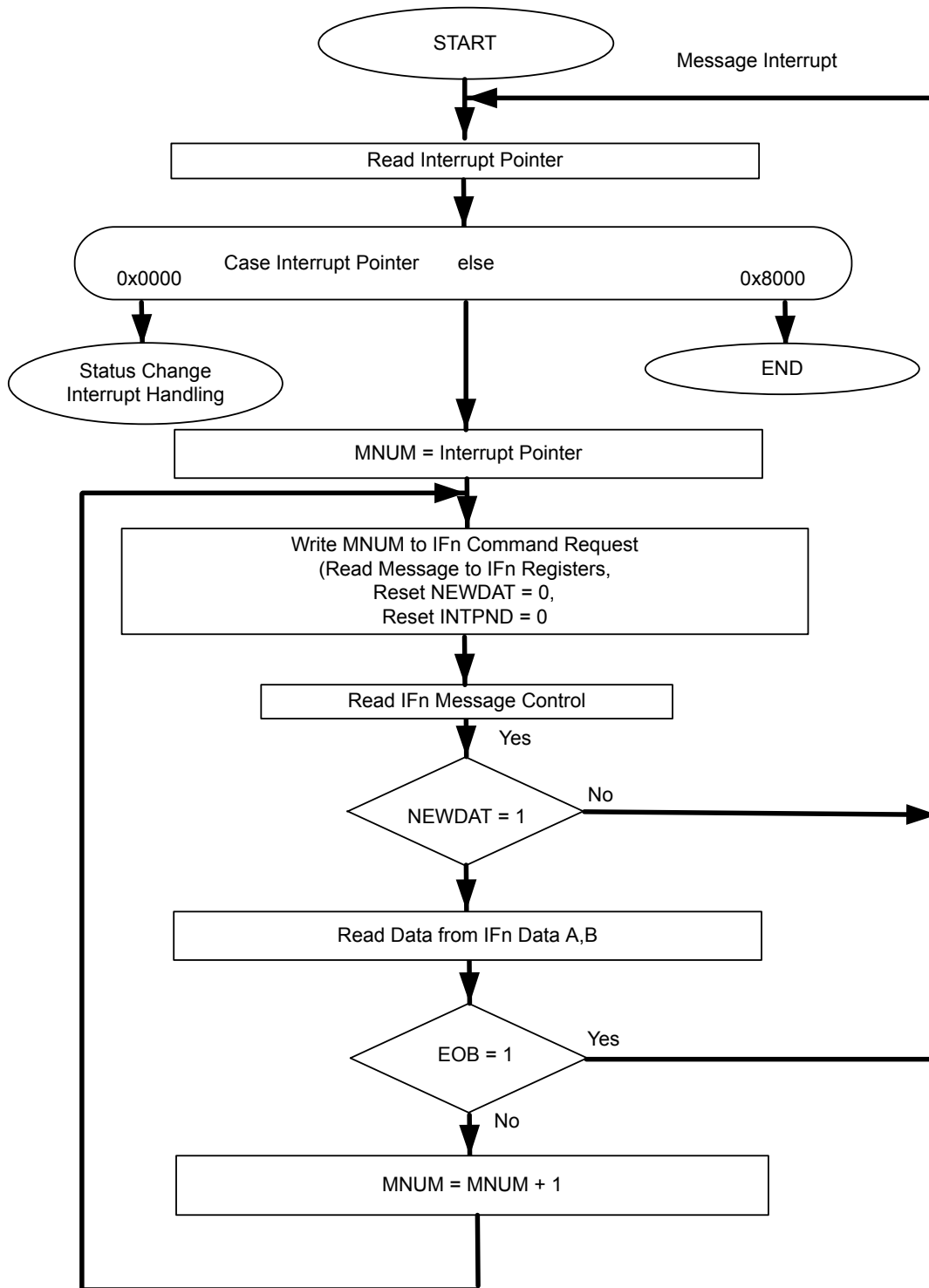
Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the `NEWDAT` of the **CANIFnMCTL** register bit of this message object is set. By setting

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 are written into the last message object of the FIFO buffer and therefore overwrite previous messages.

19.3.11.3 Reading from a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the **CANIFnCRQ** register, the TXRQST and CLRINTPND bits in the **CANIFnCMSK** register should be set such that the NEWDAT and INTPEND bits in the **CANIFnMCTL** register are cleared after the read. The values of these bits in the **CANIFnMCTL** register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. Figure 19-3 on page 761 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 19-3. Message Objects in a FIFO Buffer



19.3.12 Handling of Interrupts

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. The status interrupt has the highest

priority. Among the message interrupts, the message object's interrupt with the lowest message number has the highest priority. A message interrupt is cleared by clearing the message object's `INTPND` bit in the **CANIFnMCTL** register or by reading the **CAN Status (CANSTS)** register. The status Interrupt is cleared by reading the **CANSTS** register.

The interrupt identifier `INTID` in the **CANINT** register indicates the cause of the interrupt. When no interrupt is pending, the register reads as 0x0000. If the value of the `INTID` field is different from 0, then an interrupt is pending. If the `IE` bit is set in the **CANCTL** register, the interrupt line to the interrupt controller is active. The interrupt line remains active until the `INTID` field is 0, meaning that all interrupt sources have been cleared (the cause of the interrupt is reset), or until `IE` is cleared, which disables interrupts from the CAN controller.

The `INTID` field of the **CANINT** register points to the pending message interrupt with the highest interrupt priority. The `SIE` bit in the **CANCTL** register controls whether a change of the `RXOK`, `TXOK`, and `LEC` bits in the **CANSTS** register can cause an interrupt. The `EIE` bit in the **CANCTL** register controls whether a change of the `BOFF` and `EWARN` bits in the **CANSTS** register can cause an interrupt. The `IE` bit in the **CANCTL** register controls whether any interrupt from the CAN controller actually generates an interrupt to the interrupt controller. The **CANINT** register is updated even when the `IE` bit in the **CANCTL** register is clear, but the interrupt is not indicated to the CPU.

A value of 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed, the **CANSTS** register, indicating that either an error or status interrupt has been generated. A write access to the **CANSTS** register can clear the `RXOK`, `TXOK`, and `LEC` bits in that same register; however, the only way to clear the source of a status interrupt is to read the **CANSTS** register.

The source of an interrupt can be determined in two ways during interrupt handling. The first is to read the `INTID` bit in the **CANINT** register to determine the highest priority interrupt that is pending, and the second is to read the **CAN Message Interrupt Pending (CANMSGnINT)** register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and clear the message object's `INTPND` bit at the same time by setting the `CLRINTPND` bit in the **CANIFnCMSK** register. Once the `INTPND` bit has been cleared, the **CANINT** register contains the message number for the next message object with a pending interrupt.

19.3.13 Test Mode

A Test Mode is provided which allows various diagnostics to be performed. Test Mode is entered by setting the `TEST` bit in the **CANCTL** register. Once in Test Mode, the `TX[1:0]`, `LBACK`, `SILENT` and `BASIC` bits in the **CAN Test (CANTST)** register can be used to put the CAN controller into the various diagnostic modes. The `RX` bit in the **CANTST** register allows monitoring of the `CANnRX` signal. All **CANTST** register functions are disabled when the `TEST` bit is cleared.

19.3.13.1 Silent Mode

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The CAN Controller is put in Silent Mode setting the `SILENT` bit in the **CANTST** register. In Silent Mode, the CAN controller is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag, or active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus remains in recessive state.

19.3.13.2 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.

19.3.13.3 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.

19.3.13.4 Basic Mode

Basic Mode allows the CAN Controller to be operated without the Message RAM. In Basic Mode, The CANIF1 registers are used as the transmit buffer. The transmission of the contents of the IF1 registers is requested by setting the `BUSY` bit of the `CANIF1CRQ` register. The CANIF1 registers are locked while the `BUSY` bit is set. The `BUSY` bit indicates that a transmission is pending. As soon the CAN bus is idle, the CANIF1 registers are loaded into the shift register of the CAN Controller and transmission is started. When the transmission has completed, the `BUSY` bit is cleared and the locked CANIF1 registers are released. A pending transmission can be aborted at any time by clearing the `BUSY` bit in the `CANIF1CRQ` register while the CANIF1 registers are locked. If the CPU has cleared the `BUSY` bit, a possible retransmission in case of lost arbitration or an error is disabled.

The CANIF2 Registers are used as a receive buffer. After the reception of a message, the contents of the shift register are stored in the CANIF2 registers, without any acceptance filtering. Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read message object is initiated by setting the `BUSY` bit of the `CANIF2CRQ` register, the contents of the shift register are stored into the CANIF2 registers.

In Basic Mode, all message-object-related control and status bits and of the control bits of the `CANIFnCMSK` registers are not evaluated. The message number of the `CANIFnCRQ` registers is also not evaluated. In the `CANIF2MCTL` register, the `NEWDAT` and `MSGLST` bits retain their function, the `DLC[3:0]` field shows the received DLC, the other control bits are cleared.

Basic Mode is enabled by setting the `BASIC` bit in the `CANTST` register.

19.3.13.5 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.

19.3.14 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.

19.3.15 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 19-4 on page 765): 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 19-3 on page 765). 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 system clock (f_{sys}) and the Baud Rate Prescaler (`BRP`):

$$t_q = \text{BRP} / f_{sys}$$

The CAN module's system clock f_{sys} is the frequency of its CAN module clock input.

The Synchronization Segment `Sync_Seg` 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_Seg` and the `Sync_Seg` is called the *phase error* of that edge.

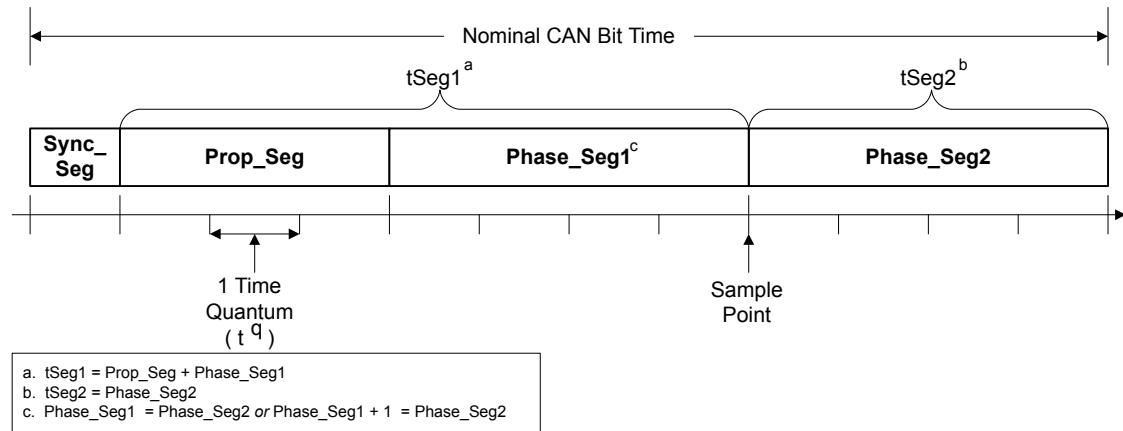
The Propagation Time Segment `Prop_Seg` is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments `Phase_Seg1` and `Phase_Seg2` 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 19-4. CAN Bit Time

Table 19-3. CAN Protocol Ranges^a

| Parameter | Range | Remark |
|------------|-------------------------|--|
| BRP | [1 .. 32] | Defines the length of the time quantum t _q |
| Sync_Seg | 1 t _q | Fixed length, synchronization of bus input to system clock |
| Prop_Seg | [1 .. 8] t _q | Compensates for the physical delay times |
| Phase_Seg1 | [1 .. 8] t _q | May be lengthened temporarily by synchronization |
| Phase_Seg2 | [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. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, and SJW and BRP are combined in the other byte.

In these bit timing registers, the four components TSEG1, TSEG2, 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. Therefore, the length of the bit time is (programmed values):

$$[TSEG1 + TSEG2 + 3] \times t_q$$

or (functional values):

$$[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] \times t_q$$

The data in the **CANBIT** register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) defines the length of the time quantum, the basic time unit of the bit time; the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. In addition, the controller generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. The bit value is received or transmitted at the sample point. The information processing time

(IPT) is the time after the sample point needed to calculate the next bit to be transmitted on the CAN bus. The IPT includes any of the following: retrieving the next data bit, handling a CRC bit, determining if bit stuffing is required, generating an error flag or simply going idle.

The IPT is application-specific but may not be longer than $2 t_q$; the CAN's IPT is $0 t_q$. The IPT is the lower limit of the programmed length of `Phase_Seg2`. In case of synchronization, `Phase_Seg2` may be shortened to a value less than IPT, which does not affect bus timing.

19.3.16 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 the `Prop_Seg`. 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_Seg` is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The `Sync_Seg` is $1 t_q$ long (fixed), which leaves $(\text{bit time} - \text{Prop_Seg} - 1) t_q$ 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, `Phase_Seg2 = Phase_Seg1`, else `Phase_Seg2 = Phase_Seg1 + 1`.

The minimum nominal length of `Phase_Seg2` has to be regarded as well. `Phase_Seg2` may not be shorter than the CAN controller's IPT, which is t_q .

The length of the synchronization jump width is set to its maximum value, which is the minimum of 4 and `Phase_Seg1`.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times f_{nom} \leq f_{osc} \leq (1 + df) \times f_{nom}$$

where:

- df = Maximum tolerance of oscillator frequency
- f_{osc} = Actual oscillator frequency
- f_{nom} = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(\text{Phase_seg1}, \text{Phase_seg2}) \min}{2 \times (13 \times t_{bit} - \text{Phase_Seg2})}$$

$$df \max = 2 \times df \times f_{nom}$$

where:

- Phase_Seg1 and Phase_Seg2 are from Table 19-3 on page 765
- 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.

19.3.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 25 MHz, and the bit rate is 1 Mbps.

```
tq 200 ns = (BRP + 1)/CAN Clock
delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 400 ns = 2 × tq
tSJW 200 ns = 1 × tq
tTSeg1 600 ns = tProp + tSJW
tTSeg2 200 ns = (Information Processing Time + 1) × tq
tSync-Seg 200 ns = 1 × tq
bit time 1000 ns = tSync-Seg + tTSeg1 + tTSeg2
```

In the above example, the bit field values for the **CANBIT** register are: TSEG2=1, TSEG1=2, SJW=0 and BRP=3. This makes the final value programmed into the **CANBIT** register = 0x3FC0.

19.3.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 50 MHz, and the bit rate is 100 Kbps.

```
tq 500 ns = (BRP + 1)/CAN clock
delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 4.5 μs = 9 × tq
tSJW 2 μs = 4 × tq
tTSeg1 6.5 μs = tProp + tSJW
tTSeg2 3 μs = (Information Processing Time + 6) × tq
tSync-Seg 500 ns = 1 × tq
bit time 10 μs = tSync-Seg + tTSeg1 + tTSeg2
```

In the above example, the bit field values for the **CANBIT** register are: TSEG2=5, TSEG1=12, SJW=3 and BRP=24. This makes the final value programmed into the **CANBIT** register = 0x5CD8.

19.4 Register Map

Table 19-4 on page 768 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000
- CAN1: 0x4004.1000

Note that the CAN controller clock must be enabled before the registers can be programmed (see page 166).

Table 19-4. CAN Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|------------|------|-------------|-----------------------------------|----------|
| 0x000 | CANCTL | R/W | 0x0000.0001 | CAN Control | 770 |
| 0x004 | CANSTS | R/W | 0x0000.0000 | CAN Status | 772 |
| 0x008 | CANERR | RO | 0x0000.0000 | CAN Error Counter | 775 |
| 0x00C | CANBIT | R/W | 0x0000.2301 | CAN Bit Timing | 776 |
| 0x010 | CANINT | RO | 0x0000.0000 | CAN Interrupt | 778 |
| 0x014 | CANTST | R/W | 0x0000.0000 | CAN Test | 779 |
| 0x018 | CANBRPE | R/W | 0x0000.0000 | CAN Baud Rate Prescaler Extension | 781 |
| 0x020 | CANIF1CRQ | R/W | 0x0000.0001 | CAN IF1 Command Request | 782 |
| 0x024 | CANIF1CMSK | R/W | 0x0000.0000 | CAN IF1 Command Mask | 783 |
| 0x028 | CANIF1MSK1 | R/W | 0x0000.FFFF | CAN IF1 Mask 1 | 786 |
| 0x02C | CANIF1MSK2 | R/W | 0x0000.FFFF | CAN IF1 Mask 2 | 787 |
| 0x030 | CANIF1ARB1 | R/W | 0x0000.0000 | CAN IF1 Arbitration 1 | 789 |
| 0x034 | CANIF1ARB2 | R/W | 0x0000.0000 | CAN IF1 Arbitration 2 | 790 |
| 0x038 | CANIF1MCTL | R/W | 0x0000.0000 | CAN IF1 Message Control | 792 |
| 0x03C | CANIF1DA1 | R/W | 0x0000.0000 | CAN IF1 Data A1 | 795 |
| 0x040 | CANIF1DA2 | R/W | 0x0000.0000 | CAN IF1 Data A2 | 795 |
| 0x044 | CANIF1DB1 | R/W | 0x0000.0000 | CAN IF1 Data B1 | 795 |
| 0x048 | CANIF1DB2 | R/W | 0x0000.0000 | CAN IF1 Data B2 | 795 |
| 0x080 | CANIF2CRQ | R/W | 0x0000.0001 | CAN IF2 Command Request | 782 |
| 0x084 | CANIF2CMSK | R/W | 0x0000.0000 | CAN IF2 Command Mask | 783 |
| 0x088 | CANIF2MSK1 | R/W | 0x0000.FFFF | CAN IF2 Mask 1 | 786 |
| 0x08C | CANIF2MSK2 | R/W | 0x0000.FFFF | CAN IF2 Mask 2 | 787 |
| 0x090 | CANIF2ARB1 | R/W | 0x0000.0000 | CAN IF2 Arbitration 1 | 789 |
| 0x094 | CANIF2ARB2 | R/W | 0x0000.0000 | CAN IF2 Arbitration 2 | 790 |
| 0x098 | CANIF2MCTL | R/W | 0x0000.0000 | CAN IF2 Message Control | 792 |

Table 19-4. CAN Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|------------|------|-------------|---------------------------------|----------|
| 0x09C | CANIF2DA1 | R/W | 0x0000.0000 | CAN IF2 Data A1 | 795 |
| 0x0A0 | CANIF2DA2 | R/W | 0x0000.0000 | CAN IF2 Data A2 | 795 |
| 0x0A4 | CANIF2DB1 | R/W | 0x0000.0000 | CAN IF2 Data B1 | 795 |
| 0x0A8 | CANIF2DB2 | R/W | 0x0000.0000 | CAN IF2 Data B2 | 795 |
| 0x100 | CANTXRQ1 | RO | 0x0000.0000 | CAN Transmission Request 1 | 796 |
| 0x104 | CANTXRQ2 | RO | 0x0000.0000 | CAN Transmission Request 2 | 796 |
| 0x120 | CANNWDA1 | RO | 0x0000.0000 | CAN New Data 1 | 797 |
| 0x124 | CANNWDA2 | RO | 0x0000.0000 | CAN New Data 2 | 797 |
| 0x140 | CANMSG1INT | RO | 0x0000.0000 | CAN Message 1 Interrupt Pending | 798 |
| 0x144 | CANMSG2INT | RO | 0x0000.0000 | CAN Message 2 Interrupt Pending | 798 |
| 0x160 | CANMSG1VAL | RO | 0x0000.0000 | CAN Message 1 Valid | 799 |
| 0x164 | CANMSG2VAL | RO | 0x0000.0000 | CAN Message 2 Valid | 799 |

19.5 CAN Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers that are used to access the Message Objects in the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

Register 1: CAN Control (CANCTL), offset 0x000

This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or clearing `INIT`. If the device goes bus-off, it sets `INIT`, stopping all bus activities. Once `INIT` has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 * 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after `INIT` is cleared, each time a sequence of 11 High bits has been monitored, a `BITERROR0` code is written to the **CANSTS** register (the `LEC` field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

CAN Control (CANCTL)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 Offset 0x000
 Type 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 | | | | | | |
| | | | | <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The CAN controller is operating normally.</td> </tr> <tr> <td>1</td> <td>The CAN controller is in test mode.</td> </tr> </table> | Value | Description | 0 | The CAN controller is operating normally. | 1 | The CAN controller is in test mode. |
| Value | Description | | | | | | | | | |
| 0 | The CAN controller is operating normally. | | | | | | | | | |
| 1 | The CAN controller is in test mode. | | | | | | | | | |
| 6 | CCE | R/W | 0 | Configuration Change Enable | | | | | | |
| | | | | <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Write accesses to the CANBIT register are not allowed.</td> </tr> <tr> <td>1</td> <td>Write accesses to the CANBIT register are allowed if the <code>INIT</code> bit is 1.</td> </tr> </table> | Value | Description | 0 | Write accesses to the CANBIT register are not allowed. | 1 | Write accesses to the CANBIT register are allowed if the <code>INIT</code> bit is 1. |
| Value | Description | | | | | | | | | |
| 0 | Write accesses to the CANBIT register are not allowed. | | | | | | | | | |
| 1 | Write accesses to the CANBIT register are allowed if the <code>INIT</code> bit is 1. | | | | | | | | | |
| 5 | DAR | R/W | 0 | Disable Automatic-Retransmission | | | | | | |
| | | | | <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Auto-retransmission of disturbed messages is enabled.</td> </tr> <tr> <td>1</td> <td>Auto-retransmission is disabled.</td> </tr> </table> | Value | Description | 0 | Auto-retransmission of disturbed messages is enabled. | 1 | Auto-retransmission is disabled. |
| Value | Description | | | | | | | | | |
| 0 | Auto-retransmission of disturbed messages is enabled. | | | | | | | | | |
| 1 | Auto-retransmission is disabled. | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|--|------|-------|---|-------|-------------|---|---|---|--|
| 4 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | |
| 3 | EIE | R/W | 0 | Error Interrupt Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error status interrupt is generated.</td> </tr> <tr> <td>1</td> <td>A change in the <i>BOFF</i> or <i>EWARN</i> bits in the CANSTS register generates an interrupt.</td> </tr> </tbody> </table> | Value | Description | 0 | No error status interrupt is generated. | 1 | A change in the <i>BOFF</i> or <i>EWARN</i> bits in the CANSTS register generates an interrupt. |
| Value | Description | | | | | | | | | |
| 0 | No error status interrupt is generated. | | | | | | | | | |
| 1 | A change in the <i>BOFF</i> or <i>EWARN</i> bits in the CANSTS register generates an interrupt. | | | | | | | | | |
| 2 | SIE | R/W | 0 | Status Interrupt Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No status interrupt is generated.</td> </tr> <tr> <td>1</td> <td>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 <i>TXOK</i>, <i>RXOK</i> or <i>LEC</i> bits in the CANSTS register generates an interrupt.</td> </tr> </tbody> </table> | Value | Description | 0 | No status interrupt is generated. | 1 | An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the <i>TXOK</i> , <i>RXOK</i> or <i>LEC</i> bits in the CANSTS register generates an interrupt. |
| Value | Description | | | | | | | | | |
| 0 | No status interrupt is generated. | | | | | | | | | |
| 1 | An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the <i>TXOK</i> , <i>RXOK</i> or <i>LEC</i> bits in the CANSTS register generates an interrupt. | | | | | | | | | |
| 1 | IE | R/W | 0 | CAN Interrupt Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupts disabled.</td> </tr> <tr> <td>1</td> <td>Interrupts enabled.</td> </tr> </tbody> </table> | Value | Description | 0 | Interrupts disabled. | 1 | Interrupts enabled. |
| Value | Description | | | | | | | | | |
| 0 | Interrupts disabled. | | | | | | | | | |
| 1 | Interrupts enabled. | | | | | | | | | |
| 0 | INIT | R/W | 1 | Initialization <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal operation.</td> </tr> <tr> <td>1</td> <td>Initialization started.</td> </tr> </tbody> </table> | Value | Description | 0 | Normal operation. | 1 | Initialization started. |
| Value | Description | | | | | | | | | |
| 0 | Normal operation. | | | | | | | | | |
| 1 | Initialization started. | | | | | | | | | |

Register 2: CAN Status (CANSTS), offset 0x004

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

The LEC field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error. The unused error code 0x7 may be written by the CPU to manually set this field to an invalid error so that it can be checked for a change later.

An error interrupt is generated by the BOFF and EWARN bits, and a status interrupt is generated by the RXOK, TXOK, and LEC bits, if the corresponding enable bits in the **CAN Control (CANCTL)** register are set. A change of the EPASS bit or a write to the RXOK, TXOK, or LEC bits does not generate an interrupt.

Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Status (CANSTS)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 Offset 0x004
 Type 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 |
| | | | Value | Description |
| | | | 0 | The CAN controller is not in bus-off state. |
| | | | 1 | The CAN controller is in bus-off state. |
| 6 | EWARN | RO | 0 | Warning Status |
| | | | Value | Description |
| | | | 0 | Both error counters are below the error warning limit of 96. |
| | | | 1 | At least one of the error counters has reached the error warning limit of 96. |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|--|------|-------|--|-------|-------------|---|---|---|--|
| 5 | EPASS | RO | 0 | <p>Error Passive</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127.</td> </tr> <tr> <td>1</td> <td>The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.</td> </tr> </tbody> </table> | Value | Description | 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. |
| Value | Description | | | | | | | | | |
| 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. | | | | | | | | | |
| 4 | RXOK | R/W | 0 | <p>Received a Message Successfully</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Since this bit was last cleared, no message has been successfully received.</td> </tr> <tr> <td>1</td> <td>Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering.</td> </tr> </tbody> </table> <p>This bit must be cleared by writing a 0 to it.</p> | Value | Description | 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. |
| Value | Description | | | | | | | | | |
| 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. | | | | | | | | | |
| 3 | TXOK | R/W | 0 | <p>Transmitted a Message Successfully</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Since this bit was last cleared, no message has been successfully transmitted.</td> </tr> <tr> <td>1</td> <td>Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node.</td> </tr> </tbody> </table> <p>This bit must be cleared by writing a 0 to it.</p> | Value | Description | 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. |
| Value | Description | | | | | | | | | |
| 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. | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|---|------|-------|--|-------|-------------|-----|----------|-----|-------------|--|---|-----|--------------|--|---|-----|-----------|--|---|-----|-------------|--|---|--|--|-----|-------------|--|---|--|---|-----|-----------|--|---|-----|--------|--|---|
| 2:0 | LEC | R/W | 0x0 | <p>Last Error Code</p> <p>This is the type of the last error to occur on the CAN bus.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No Error</td> </tr> <tr> <td>0x1</td> <td>Stuff Error</td> </tr> <tr> <td></td> <td>More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.</td> </tr> <tr> <td>0x2</td> <td>Format Error</td> </tr> <tr> <td></td> <td>A fixed format part of the received frame has the wrong format.</td> </tr> <tr> <td>0x3</td> <td>ACK Error</td> </tr> <tr> <td></td> <td>The message transmitted was not acknowledged by another node.</td> </tr> <tr> <td>0x4</td> <td>Bit 1 Error</td> </tr> <tr> <td></td> <td>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.</td> </tr> <tr> <td></td> <td>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).</td> </tr> <tr> <td>0x5</td> <td>Bit 0 Error</td> </tr> <tr> <td></td> <td>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).</td> </tr> <tr> <td></td> <td>During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. By checking for this status, software can monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus.</td> </tr> <tr> <td>0x6</td> <td>CRC Error</td> </tr> <tr> <td></td> <td>The CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data.</td> </tr> <tr> <td>0x7</td> <td>Unused</td> </tr> <tr> <td></td> <td>When the LEC bit shows this value, no CAN bus event was detected since this value was written to the LEC field.</td> </tr> </tbody> </table> | Value | Description | 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. By checking for this status, software can monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus. | 0x6 | CRC Error | | 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 | Unused | | When the LEC bit shows this value, no CAN bus event was detected since this value was written to the LEC field. |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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. By checking for this status, software can monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | CRC Error | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 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 | Unused | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | When the LEC bit shows this value, no CAN bus event was detected since this value was written to the LEC field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Register 3: CAN Error Counter (CANERR), offset 0x008

This register contains the error counter values, which can be used to analyze the cause of an error.

CAN Error Counter (CANERR)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x008

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|-----|----|----|----|----|----|-----|----|----|----|----|----|----|----|
| | 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 |
| | RP | | REC | | | | | | TEC | | | | | | | |
| 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 | | | | | | |
|-----------|---|------|--------|--|-------|-------------|---|---|---|---|
| 31:16 | 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 | | | | | | |
| | | | | <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The Receive Error counter is below the Error Passive level (127 or less).</td> </tr> <tr> <td>1</td> <td>The Receive Error counter has reached the Error Passive level (128 or greater).</td> </tr> </tbody> </table> | Value | Description | 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). |
| Value | Description | | | | | | | | | |
| 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 This field contains the state of the receiver error counter (0 to 127). | | | | | | |
| 7:0 | TEC | RO | 0x00 | Transmit Error Counter This field contains the state of the transmit error counter (0 to 255). | | | | | | |

Register 4: CAN Bit Timing (CANBIT), offset 0x00C

This register is used to program the bit width and bit quantum. Values are programmed to the system clock frequency. This register is write-enabled by setting the `CCE` and `INIT` bits in the `CANCTL` register. See “Bit Time and Bit Rate” on page 764 for more information.

CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 Offset 0x00C
 Type R/W, reset 0x0000.2301

| | | | | | | | | | | | | | | | | |
|-------|----------|-------|-----|-----|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 | TSEG2 | | | TSEG1 | | | SJW | | | BRP | | | | | |
| Type | RO | 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 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| 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, the reset value of 0x2 means that 3 (2+1) bit time quanta are defined for <code>Phase_Seg2</code> (see Figure 19-4 on page 765). The bit time quanta is defined by the <code>BRP</code> 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 means that 4 (3+1) bit time quanta are defined for <code>Phase_Seg1</code> (see Figure 19-4 on page 765). The bit time quanta is defined by the <code>BRP</code> 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 <code>TSEG2</code> or <code>TSEG1</code> by the value in <code>SJW</code> . So the reset value of 0 adjusts the length by 1 bit time quanta. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 5:0 | BRP | R/W | 0x1 | <p>Baud Rate Prescaler</p> <p>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.</p> <p>0x00-0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.</p> <p>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).</p> <p>The CANBRPE register can be used to further divide the bit time.</p> |

Register 5: CAN Interrupt (CANINT), offset 0x010

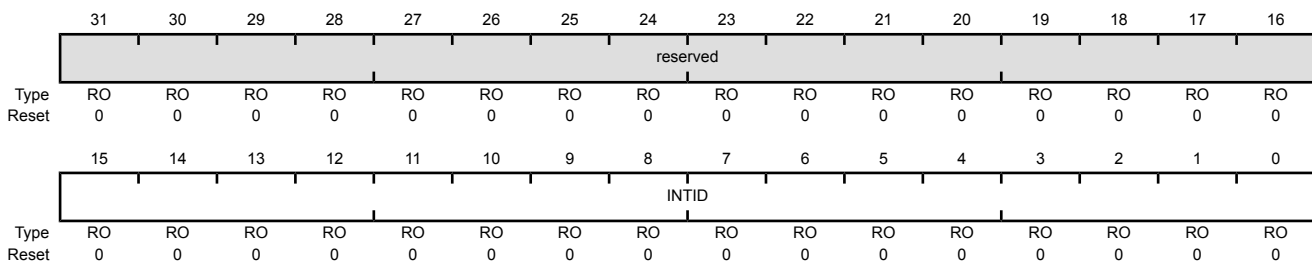
This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the **INTID** field is not 0x0000 (the default) and the **IE** bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the **INTID** field is cleared by reading the **CANSTS** register, or until the **IE** bit in the **CANCTL** register is cleared.

Note: Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 Offset 0x010
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|---------------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | INTID | RO | 0x0000 | Interrupt Identifier The number in this field indicates the source of the interrupt. |
| | | | Value | Description |
| | | | 0x0000 | No interrupt pending |
| | | | 0x0001-0x0020 | Number of the message object that caused the interrupt |
| | | | 0x0021-0x7FFF | Unused |
| | | | 0x8000 | Status Interrupt |
| | | | 0x8001-0xFFFF | Unused |

Register 6: CAN Test (CANTST), offset 0x014

This register is used for self-test and external pin access. It is write-enabled by setting the `TEST` bit in the `CANCTL` register. Different test functions may be combined, however, CAN transfers are affected if the `TX` bits in this register are not zero.

CAN Test (CANTST)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 Offset 0x014
 Type 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 | | | | | | | | RX | TX | LBACK | SILENT | BASIC | reserved | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|--|------|-----------|--|-------|-------------|-----|--|-----|--|-----|---|-----|--|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 7 | RX | RO | 0 | Receive Observation <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The <code>CANnRx</code> pin is low.</td> </tr> <tr> <td>1</td> <td>The <code>CANnRx</code> pin is high.</td> </tr> </table> | Value | Description | 0 | The <code>CANnRx</code> pin is low. | 1 | The <code>CANnRx</code> pin is high. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | The <code>CANnRx</code> pin is low. | | | | | | | | | | | | | |
| 1 | The <code>CANnRx</code> pin is high. | | | | | | | | | | | | | |
| 6:5 | TX | R/W | 0x0 | Transmit Control Overrides control of the <code>CANnTx</code> pin. <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0x0</td> <td><code>CANnTx</code> is controlled by the CAN module; default operation</td> </tr> <tr> <td>0x1</td> <td>The sample point is driven on the <code>CANnTx</code> signal. This mode is useful to monitor bit timing.</td> </tr> <tr> <td>0x2</td> <td><code>CANnTx</code> drives a low value. This mode is useful for checking the physical layer of the CAN bus.</td> </tr> <tr> <td>0x3</td> <td><code>CANnTx</code> drives a high value. This mode is useful for checking the physical layer of the CAN bus.</td> </tr> </table> | Value | Description | 0x0 | <code>CANnTx</code> is controlled by the CAN module; default operation | 0x1 | The sample point is driven on the <code>CANnTx</code> signal. This mode is useful to monitor bit timing. | 0x2 | <code>CANnTx</code> drives a low value. This mode is useful for checking the physical layer of the CAN bus. | 0x3 | <code>CANnTx</code> drives a high value. This mode is useful for checking the physical layer of the CAN bus. |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | <code>CANnTx</code> is controlled by the CAN module; default operation | | | | | | | | | | | | | |
| 0x1 | The sample point is driven on the <code>CANnTx</code> signal. This mode is useful to monitor bit timing. | | | | | | | | | | | | | |
| 0x2 | <code>CANnTx</code> drives a low value. This mode is useful for checking the physical layer of the CAN bus. | | | | | | | | | | | | | |
| 0x3 | <code>CANnTx</code> drives a high value. This mode is useful for checking the physical layer of the CAN bus. | | | | | | | | | | | | | |
| 4 | LBACK | R/W | 0 | Loopback Mode <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Loopback mode is disabled.</td> </tr> <tr> <td>1</td> <td>Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.</td> </tr> </table> | Value | Description | 0 | Loopback mode is disabled. | 1 | Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | Loopback mode is disabled. | | | | | | | | | | | | | |
| 1 | Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored. | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | |
|-----------|---|------|-------|---|--------------------------|
| 3 | SILENT | R/W | 0 | Silent Mode | |
| | | | | Value | Description |
| | | | | 0 | Silent mode is disabled. |
| 1 | Silent mode is enabled. In silent mode, the CAN controller does not transmit data but instead monitors the bus. This mode is also known as Bus Monitor mode. | | | | |
| 2 | BASIC | R/W | 0 | Basic Mode | |
| | | | | Value | Description |
| | | | | 0 | Basic mode is disabled. |
| 1 | Basic mode is enabled. In basic mode, software should use the CANIF1 registers as the transmit buffer and use the CANIF2 registers as the receive buffer. | | | | |
| 1:0 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | |

Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018

This register is used to further divide the bit time set with the `BRP` bit in the `CANBIT` register. It is write-enabled by setting the `CCE` bit in the `CANCTL` register.

CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x018

Type 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 | | | | | | | | | | | | BRPE | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:4 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3:0 | BRPE | R/W | 0x0 | Baud Rate Prescaler Extension 0x00-0x0F: Extend the <code>BRP</code> bit in the <code>CANBIT</code> register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by <code>BRPE</code> (MSBs) and <code>BRP</code> (LSBs). |

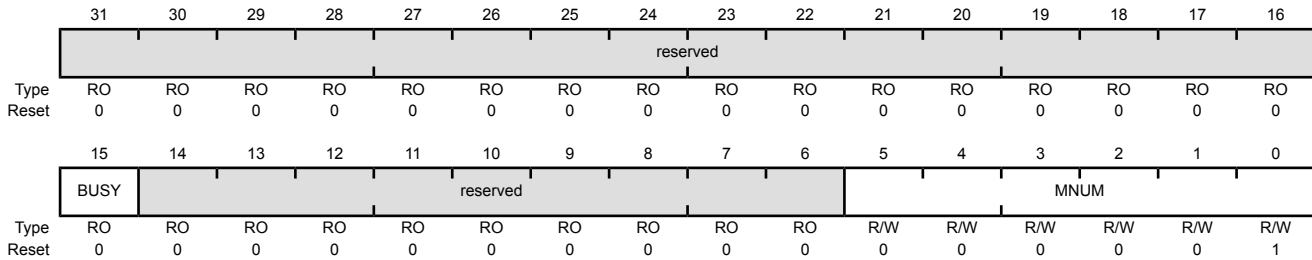
Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

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
 CAN1 base: 0x4004.1000
 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 | | | | | | | | |
| | | | | <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>This bit is cleared when read/write action has finished.</td> </tr> <tr> <td>1</td> <td>This bit is set when a write occurs to the message number in this register.</td> </tr> </tbody> </table> | Value | Description | 0 | This bit is cleared when read/write action has finished. | 1 | This bit is set when a write occurs to the message number in this register. | | |
| Value | Description | | | | | | | | | | | |
| 0 | This bit is cleared when read/write action has finished. | | | | | | | | | | | |
| 1 | This bit is set when a write occurs to the message number in this register. | | | | | | | | | | | |
| 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. | | | | | | | | |
| | | | | <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>0 is not a valid message number; it is interpreted as 0x20, or object 32.</td> </tr> <tr> <td>0x01-0x20</td> <td>Indicates specified message object 1 to 32.</td> </tr> <tr> <td>0x21-0x3F</td> <td>Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.</td> </tr> </tbody> </table> | Value | Description | 0x00 | 0 is not a valid message number; it is interpreted as 0x20, or object 32. | 0x01-0x20 | Indicates specified message object 1 to 32. | 0x21-0x3F | Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F. |
| Value | Description | | | | | | | | | | | |
| 0x00 | 0 is not a valid message number; it is interpreted as 0x20, or object 32. | | | | | | | | | | | |
| 0x01-0x20 | Indicates specified message object 1 to 32. | | | | | | | | | | | |
| 0x21-0x3F | 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
CAN1 base: 0x4004.1000
Offset 0x024
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 | | | | | | | | WRNRD | MASK | ARB | CONTROL | CLRINTPND | NEWDAT / TXRQST | DATAA | DATAB |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7 | WRNRD | R/W | 0 | Write, Not Read |
| | | | | Value Description |
| | | | | 0 Transfer the data in the CANIFn registers to the CAN message object specified by the MNUM field in the CAN Command Request (CANIFnCRQ) . |
| | | | | 1 Transfer the data in the CAN message object specified by the the MNUM field in the CANIFnCRQ register into the CANIFn 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 |
| | | | | Value Description |
| | | | | 0 Mask bits unchanged. |
| | | | | 1 Transfer IDMASK + DIR + MXTD of the message object into the Interface registers. |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | |
|-----------|---|---|-------|---|-------|-------------|-------------|-----------------------------|---|--|---|---|---|---|---|---|---|---|--|
| 5 | ARB | R/W | 0 | <p>Access Arbitration Bits</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Arbitration bits unchanged.</td> </tr> <tr> <td>1</td> <td>Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.</td> </tr> </tbody> </table> | Value | Description | 0 | Arbitration bits unchanged. | 1 | Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers. | | | | | | | | | |
| Value | Description | | | | | | | | | | | | | | | | | | |
| 0 | Arbitration bits unchanged. | | | | | | | | | | | | | | | | | | |
| 1 | Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers. | | | | | | | | | | | | | | | | | | |
| 4 | CONTROL | R/W | 0 | <p>Access Control Bits</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Control bits unchanged.</td> </tr> <tr> <td>1</td> <td>Transfer control bits from the CANIFnMCTL register into the Interface registers.</td> </tr> </tbody> </table> | Value | Description | 0 | Control bits unchanged. | 1 | Transfer control bits from the CANIFnMCTL register into the Interface registers. | | | | | | | | | |
| Value | Description | | | | | | | | | | | | | | | | | | |
| 0 | Control bits unchanged. | | | | | | | | | | | | | | | | | | |
| 1 | Transfer control bits from the CANIFnMCTL register into the Interface registers. | | | | | | | | | | | | | | | | | | |
| 3 | CLRINTPND | R/W | 0 | <p>Clear Interrupt Pending Bit</p> <p>The function of this bit depends on the configuration of the WRNRD bit as follows:</p> <table border="1"> <thead> <tr> <th>WRNRD</th> <th>CLRINTPND</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The interrupt pending status is transferred from the message buffer into the CANIFnMCTL register.</td> </tr> <tr> <td>0</td> <td>1</td> <td>The interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing.</td> </tr> <tr> <td>1</td> <td>0</td> <td>The INTPND bit in the message object remains unchanged.</td> </tr> <tr> <td>1</td> <td>1</td> <td>The INTPND bit is cleared in the message object.</td> </tr> </tbody> </table> | WRNRD | CLRINTPND | Description | 0 | 0 | The interrupt pending status is transferred from the message buffer into the CANIFnMCTL register. | 0 | 1 | The interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. | 1 | 0 | The INTPND bit in the message object remains unchanged. | 1 | 1 | The INTPND bit is cleared in the message object. |
| WRNRD | CLRINTPND | Description | | | | | | | | | | | | | | | | | |
| 0 | 0 | The interrupt pending status is transferred from the message buffer into the CANIFnMCTL register. | | | | | | | | | | | | | | | | | |
| 0 | 1 | The interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. | | | | | | | | | | | | | | | | | |
| 1 | 0 | The INTPND bit in the message object remains unchanged. | | | | | | | | | | | | | | | | | |
| 1 | 1 | The INTPND bit is cleared in the message object. | | | | | | | | | | | | | | | | | |
| 2 | NEWDAT / TXRQST | R/W | 0 | <p>NEWDAT / TXRQST Bit</p> <p>The function of this bit depends on the configuration of the WRNRD bit as follows:</p> <table border="1"> <thead> <tr> <th>WRNRD</th> <th>CLRINTPND</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The value of the new data status is transferred from the message buffer into the CANIFnMCTL register.</td> </tr> <tr> <td>0</td> <td>1</td> <td>The new data status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing.</td> </tr> <tr> <td>1</td> <td>0</td> <td>Transmission is not requested</td> </tr> <tr> <td>1</td> <td>1</td> <td>Request a transmission. Note that when this bit is set, the TXRQST bit in the CANIFnMCTL register is ignored.</td> </tr> </tbody> </table> | WRNRD | CLRINTPND | Description | 0 | 0 | The value of the new data status is transferred from the message buffer into the CANIFnMCTL register. | 0 | 1 | The new data status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. | 1 | 0 | Transmission is not requested | 1 | 1 | Request a transmission. Note that when this bit is set, the TXRQST bit in the CANIFnMCTL register is ignored. |
| WRNRD | CLRINTPND | Description | | | | | | | | | | | | | | | | | |
| 0 | 0 | The value of the new data status is transferred from the message buffer into the CANIFnMCTL register. | | | | | | | | | | | | | | | | | |
| 0 | 1 | The new data status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. | | | | | | | | | | | | | | | | | |
| 1 | 0 | Transmission is not requested | | | | | | | | | | | | | | | | | |
| 1 | 1 | Request a transmission. Note that when this bit is set, the TXRQST bit in the CANIFnMCTL register is ignored. | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | |
|-----------|-----------|---|-------|--|-------|-----------|-------------|---|---|-------------------------------|---|---|---|---|---|--|
| 1 | DATAA | R/W | 0 | <p>Access Data Byte 0 to 3</p> <p>The function of this bit depends on the configuration of the WRNRD bit as follows:</p> <table border="1"> <thead> <tr> <th>WRNRD</th> <th>CLRINTPND</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>X</td> <td>0</td> <td>Data bytes 0-3 are unchanged.</td> </tr> <tr> <td>0</td> <td>1</td> <td>Transfer data bytes 0-3 in CANIFnDA1 and CANIFnDA2 to the message object.</td> </tr> <tr> <td>1</td> <td>1</td> <td>Transfer data bytes 0-3 in message object to CANIFnDA1 and CANIFnDA2.</td> </tr> </tbody> </table> | WRNRD | CLRINTPND | Description | X | 0 | Data bytes 0-3 are unchanged. | 0 | 1 | Transfer data bytes 0-3 in CANIFnDA1 and CANIFnDA2 to the message object. | 1 | 1 | Transfer data bytes 0-3 in message object to CANIFnDA1 and CANIFnDA2 . |
| WRNRD | CLRINTPND | Description | | | | | | | | | | | | | | |
| X | 0 | Data bytes 0-3 are unchanged. | | | | | | | | | | | | | | |
| 0 | 1 | Transfer data bytes 0-3 in CANIFnDA1 and CANIFnDA2 to the message object. | | | | | | | | | | | | | | |
| 1 | 1 | Transfer data bytes 0-3 in message object to CANIFnDA1 and CANIFnDA2 . | | | | | | | | | | | | | | |
| 0 | DATAB | R/W | 0 | <p>Access Data Byte 4 to 7</p> <p>The function of this bit depends on the configuration of the WRNRD bit as follows:</p> <table border="1"> <thead> <tr> <th>WRNRD</th> <th>CLRINTPND</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>X</td> <td>0</td> <td>Data bytes 4-7 are unchanged.</td> </tr> <tr> <td>0</td> <td>1</td> <td>Transfer data bytes 4-7 in CANIFnDA1 and CANIFnDA2 to the message object.</td> </tr> <tr> <td>1</td> <td>1</td> <td>Transfer data bytes 4-7 in message object to CANIFnDA1 and CANIFnDA2.</td> </tr> </tbody> </table> | WRNRD | CLRINTPND | Description | X | 0 | Data bytes 4-7 are unchanged. | 0 | 1 | Transfer data bytes 4-7 in CANIFnDA1 and CANIFnDA2 to the message object. | 1 | 1 | Transfer data bytes 4-7 in message object to CANIFnDA1 and CANIFnDA2 . |
| WRNRD | CLRINTPND | Description | | | | | | | | | | | | | | |
| X | 0 | Data bytes 4-7 are unchanged. | | | | | | | | | | | | | | |
| 0 | 1 | Transfer data bytes 4-7 in CANIFnDA1 and CANIFnDA2 to the message object. | | | | | | | | | | | | | | |
| 1 | 1 | Transfer data bytes 4-7 in message object to CANIFnDA1 and CANIFnDA2 . | | | | | | | | | | | | | | |

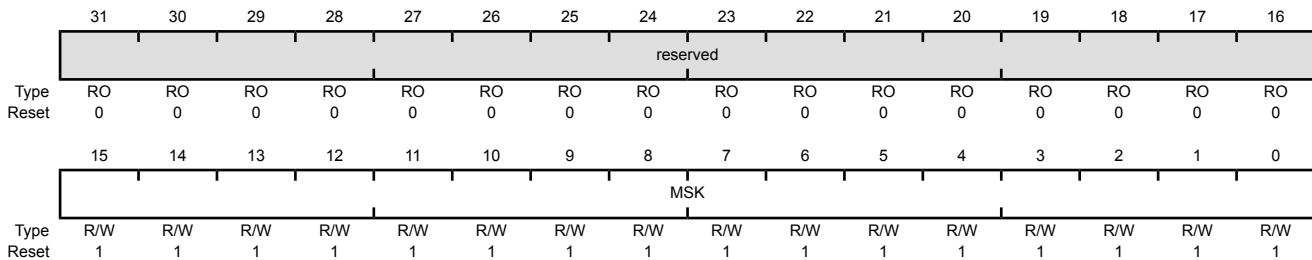
Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the **ID** bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

CAN IF1 Mask 1 (CANIF1MSK1)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 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.

| Value | Description |
|-------|--|
| 0 | The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering. |
| 1 | The corresponding identifier field (ID) is used for acceptance filtering. |

Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C**Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C**

This register holds extended mask information that accompanies the **CANIFnMSK1** register.

CAN IF1 Mask 2 (CANIF1MSK2)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x02C

Type R/W, reset 0x0000.FFFF

| | | | | | | | | | | | | | | | | |
|-------|----------|------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |
| | MXTD | MDIR | reserved | | | | | | | | | | | | | |
| Type | R/W | R/W | RO | 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 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| 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 | 1 | Mask Extended Identifier <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The extended identifier bit (<i>X</i>TD in the CANIFnARB2 register) has no effect on the acceptance filtering.</td> </tr> <tr> <td>1</td> <td>The extended identifier bit <i>X</i>TD is used for acceptance filtering.</td> </tr> </tbody> </table> | Value | Description | 0 | The extended identifier bit (<i>X</i> TD in the CANIFnARB2 register) has no effect on the acceptance filtering. | 1 | The extended identifier bit <i>X</i> TD is used for acceptance filtering. |
| Value | Description | | | | | | | | | |
| 0 | The extended identifier bit (<i>X</i> TD in the CANIFnARB2 register) has no effect on the acceptance filtering. | | | | | | | | | |
| 1 | The extended identifier bit <i>X</i> TD is used for acceptance filtering. | | | | | | | | | |
| 14 | MDIR | R/W | 1 | Mask Message Direction <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The message direction bit (<i>D</i>IR in the CANIFnARB2 register) has no effect for acceptance filtering.</td> </tr> <tr> <td>1</td> <td>The message direction bit <i>D</i>IR is used for acceptance filtering.</td> </tr> </tbody> </table> | Value | Description | 0 | The message direction bit (<i>D</i> IR in the CANIFnARB2 register) has no effect for acceptance filtering. | 1 | The message direction bit <i>D</i> IR is used for acceptance filtering. |
| Value | Description | | | | | | | | | |
| 0 | The message direction bit (<i>D</i> IR in the CANIFnARB2 register) has no effect for acceptance filtering. | | | | | | | | | |
| 1 | The message direction bit <i>D</i> IR is used for acceptance filtering. | | | | | | | | | |
| 13 | reserved | RO | 1 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|--|------|-------|---|-------|-------------|---|--|---|--|
| 12:0 | MSK | R/W | 0xFF | <p>Identifier Mask</p> <p>When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The <code>MSK</code> field in the <code>CANIFnMSK1</code> register are used for bits [15:0] of the ID. When using an 11-bit identifier, <code>MSK[12:2]</code> are used for bits [10:0] of the ID.</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>The corresponding identifier field (<code>ID</code>) in the message object cannot inhibit the match in acceptance filtering.</td></tr><tr><td>1</td><td>The corresponding identifier field (<code>ID</code>) is used for acceptance filtering.</td></tr></tbody></table> | Value | Description | 0 | The corresponding identifier field (<code>ID</code>) in the message object cannot inhibit the match in acceptance filtering. | 1 | The corresponding identifier field (<code>ID</code>) is used for acceptance filtering. |
| Value | Description | | | | | | | | | |
| 0 | The corresponding identifier field (<code>ID</code>) in the message object cannot inhibit the match in acceptance filtering. | | | | | | | | | |
| 1 | The corresponding identifier field (<code>ID</code>) 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

CAN1 base: 0x4004.1000

Offset 0x030

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 |
| | ID | | | | | | | | | | | | | | | |
| 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. |
| 15:0 | ID | R/W | 0x0000 | <p>Message Identifier</p> <p>This bit field is used with the <code>ID</code> field in the <code>CANIFnARB2</code> register to create the message identifier.</p> <p>When using a 29-bit identifier, bits 15:0 of the <code>CANIFnARB1</code> register are [15:0] of the ID, while bits 12:0 of the <code>CANIFnARB2</code> register are [28:16] of the ID.</p> <p>When using an 11-bit identifier, these bits are not used.</p> |

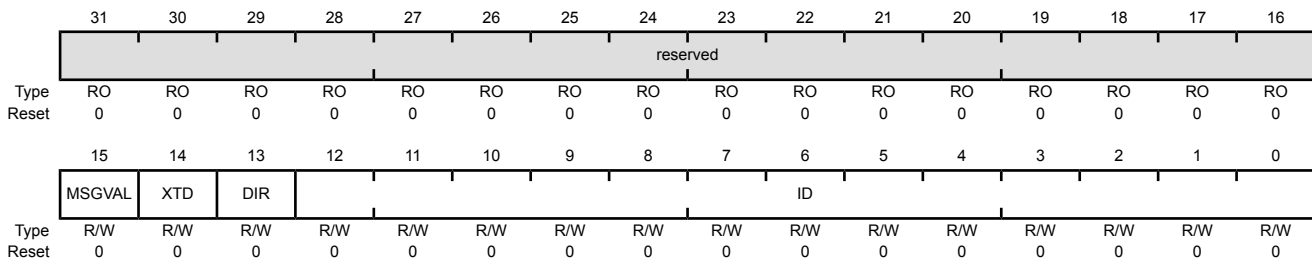
Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034

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
 CAN1 base: 0x4004.1000
 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 |
| | Value | Description | | |
| | 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 |
| | Value | Description | | |
| | 0 | An 11-bit Standard Identifier is used for this message object. | | |
| | 1 | A 29-bit Extended Identifier is used for this message object. | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 13 | DIR | R/W | 0 | <p>Message Direction</p> <p>Value Description</p> <p>0 Receive. When the <code>TXRQST</code> 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.</p> <p>1 Transmit. When the <code>TXRQST</code> 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 <code>TXRQST</code> bit of this message object is set (if <code>RMTEN=1</code>).</p> |
| 12:0 | ID | R/W | 0x000 | <p>Message Identifier</p> <p>This bit field is used with the <code>ID</code> field in the CANIFnARB2 register to create the message identifier.</p> <p>When using a 29-bit identifier, <code>ID[15:0]</code> of the CANIFnARB1 register are [15:0] of the ID, while these bits, <code>ID[12:0]</code>, are [28:16] of the ID.</p> <p>When using an 11-bit identifier, <code>ID[12:2]</code> are used for bits [10:0] of the ID. The <code>ID</code> field in the CANIFnARB1 register is ignored.</p> |

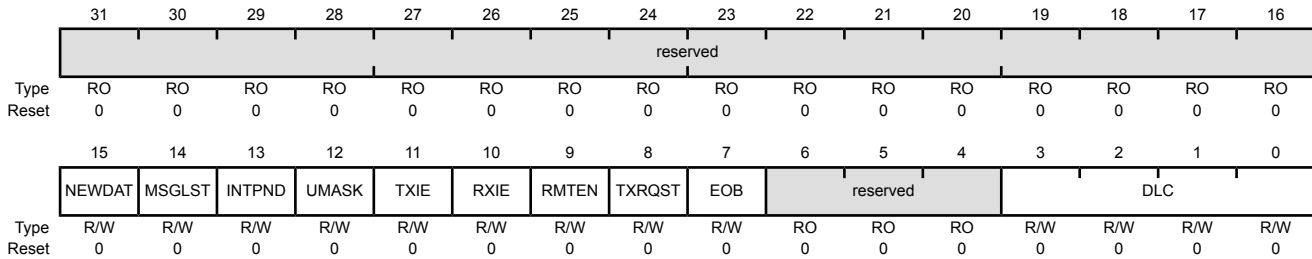
Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038

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
 CAN1 base: 0x4004.1000
 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 <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>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.</td> </tr> <tr> <td>1</td> <td>The message handler or the CPU has written new data into the data portion of this message object.</td> </tr> </table> | Value | Description | 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. |
| Value | Description | | | | | | | | | |
| 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 <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>No message was lost since the last time this bit was cleared by the CPU.</td> </tr> <tr> <td>1</td> <td>The message handler stored a new message into this object when NEWDAT was set; the CPU has lost a message.</td> </tr> </table> <p>This bit is only valid for message objects when the DIR bit in the CANIFnARB2 register is clear (receive).</p> | Value | Description | 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. |
| Value | Description | | | | | | | | | |
| 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. | | | | | | | | | |
| 13 | INTPND | R/W | 0 | Interrupt Pending <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>This message object is not the source of an interrupt.</td> </tr> <tr> <td>1</td> <td>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.</td> </tr> </table> | Value | Description | 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. |
| Value | Description | | | | | | | | | |
| 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. | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|--|------|-------|--|-------|-------------|---|--|---|--|
| 12 | UMASK | R/W | 0 | Use Acceptance Mask <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Mask is ignored.</td> </tr> <tr> <td>1</td> <td>Use mask (MSK, MXTD, and MDIR bits in the CANIFnMSKn registers) for acceptance filtering.</td> </tr> </tbody> </table> | Value | Description | 0 | Mask is ignored. | 1 | Use mask (MSK, MXTD, and MDIR bits in the CANIFnMSKn registers) for acceptance filtering. |
| Value | Description | | | | | | | | | |
| 0 | Mask is ignored. | | | | | | | | | |
| 1 | Use mask (MSK, MXTD, and MDIR bits in the CANIFnMSKn registers) for acceptance filtering. | | | | | | | | | |
| 11 | TXIE | R/W | 0 | Transmit Interrupt Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The INTPND bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame.</td> </tr> <tr> <td>1</td> <td>The INTPND bit in the CANIFnMCTL register is set after a successful transmission of a frame.</td> </tr> </tbody> </table> | Value | Description | 0 | The INTPND bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame. | 1 | The INTPND bit in the CANIFnMCTL register is set after a successful transmission of a frame. |
| Value | Description | | | | | | | | | |
| 0 | The INTPND bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame. | | | | | | | | | |
| 1 | The INTPND bit in the CANIFnMCTL register is set after a successful transmission of a frame. | | | | | | | | | |
| 10 | RXIE | R/W | 0 | Receive Interrupt Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The INTPND bit in the CANIFnMCTL register is unchanged after a successful reception of a frame.</td> </tr> <tr> <td>1</td> <td>The INTPND bit in the CANIFnMCTL register is set after a successful reception of a frame.</td> </tr> </tbody> </table> | Value | Description | 0 | The INTPND bit in the CANIFnMCTL register is unchanged after a successful reception of a frame. | 1 | The INTPND bit in the CANIFnMCTL register is set after a successful reception of a frame. |
| Value | Description | | | | | | | | | |
| 0 | The INTPND bit in the CANIFnMCTL register is unchanged after a successful reception of a frame. | | | | | | | | | |
| 1 | The INTPND bit in the CANIFnMCTL register is set after a successful reception of a frame. | | | | | | | | | |
| 9 | RMTEN | R/W | 0 | Remote Enable <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is left unchanged.</td> </tr> <tr> <td>1</td> <td>At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is set.</td> </tr> </tbody> </table> | Value | Description | 0 | At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is left unchanged. | 1 | At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is set. |
| Value | Description | | | | | | | | | |
| 0 | At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is left unchanged. | | | | | | | | | |
| 1 | At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is set. | | | | | | | | | |
| 8 | TXRQST | R/W | 0 | Transmit Request <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>This message object is not waiting for transmission.</td> </tr> <tr> <td>1</td> <td>The transmission of this message object is requested and is not yet done.</td> </tr> </tbody> </table> <p>Note: If the WRNRD and TXRQST bits in the CANIFnCMSK register are set, this bit is ignored.</p> | Value | Description | 0 | This message object is not waiting for transmission. | 1 | The transmission of this message object is requested and is not yet done. |
| Value | Description | | | | | | | | | |
| 0 | This message object is not waiting for transmission. | | | | | | | | | |
| 1 | The transmission of this message object is requested and is not yet done. | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|---|------|-------|---|-------|-------------|---------|---|---------|--|
| 7 | EOB | R/W | 0 | <p>End of Buffer</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.</td> </tr> <tr> <td>1</td> <td>Single message object or last message object of a FIFO Buffer.</td> </tr> </tbody> </table> <p>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.</p> | Value | Description | 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. |
| Value | Description | | | | | | | | | |
| 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. | | | | | | | | | |
| 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 | <p>Data Length Code</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0-0x8</td> <td>Specifies the number of bytes in the data frame.</td> </tr> <tr> <td>0x9-0xF</td> <td>Defaults to a data frame with 8 bytes.</td> </tr> </tbody> </table> <p>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.</p> | Value | Description | 0x0-0x8 | Specifies the number of bytes in the data frame. | 0x9-0xF | Defaults to a data frame with 8 bytes. |
| Value | Description | | | | | | | | | |
| 0x0-0x8 | Specifies the number of bytes in the data frame. | | | | | | | | | |
| 0x9-0xF | Defaults to a data frame with 8 bytes. | | | | | | | | | |

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
CAN1 base: 0x4004.1000
Offset 0x03C
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 |
| | DATA | | | | | | | | | | | | | | | |
| 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. |
| 15:0 | 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

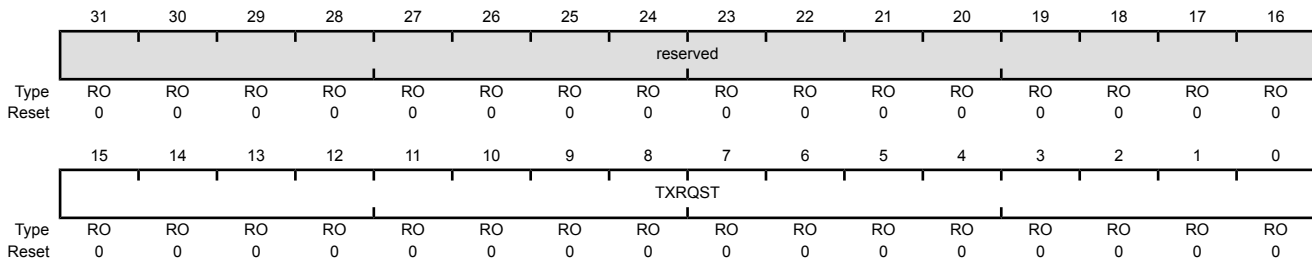
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104

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
 CAN1 base: 0x4004.1000
 Offset 0x100
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | TXRQST | RO | 0x0000 | Transmission Request Bits |
| | | | | Value Description |
| | | | | 0 The corresponding message object is not waiting for transmission. |
| | | | | 1 The transmission of the corresponding message object is requested and is not yet done. |

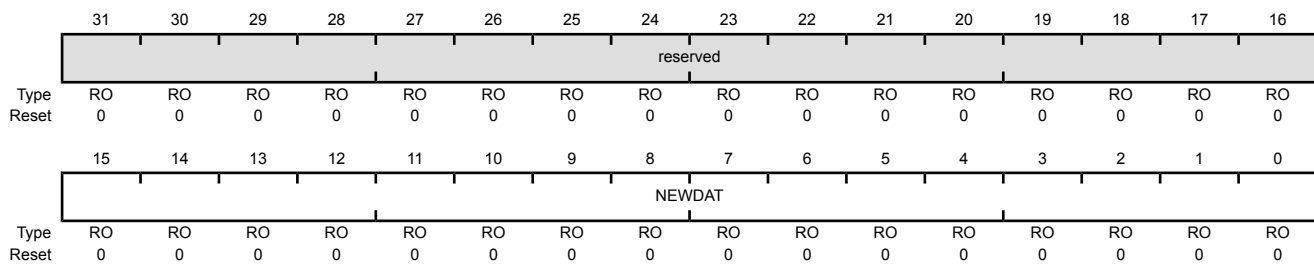
Register 32: CAN New Data 1 (CANNWDA1), offset 0x120**Register 33: CAN New Data 2 (CANNWDA2), offset 0x124**

The **CANNWDA1** and **CANNWDA2** registers hold the **NEWDAT** bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The **NEWDAT** bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a data frame, or (3) the message handler state machine after a successful transmission.

The **CANNWDA1** register contains the **NEWDAT** bits of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the **NEWDAT** bits of the second 16 message objects.

CAN New Data 1 (CANNWDA1)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 Offset 0x120
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | NEWDAT | RO | 0x0000 | New Data Bits |
| | | | | Value Description |
| | | | | 0 No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU. |
| | | | | 1 The message handler or the CPU has written new data into the data portion of the corresponding message object. |

Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

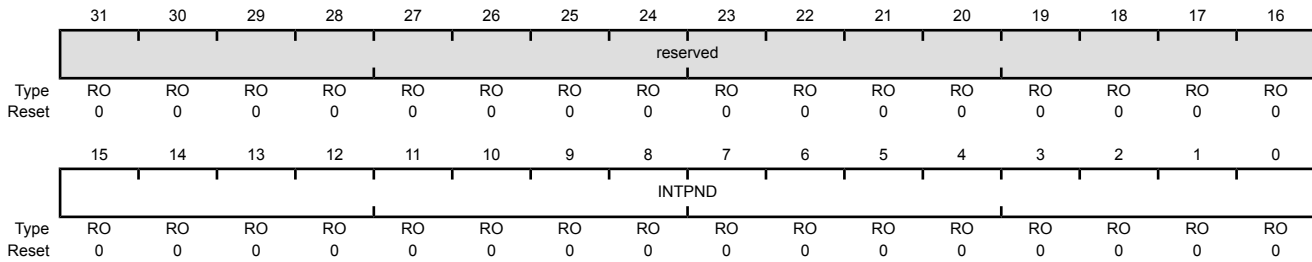
The **CANMSG1INT** and **CANMSG2INT** registers hold the **INTPND** bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The **INTPND** bit of a specific message object can be changed through two sources: (1) the CPU via the **CANIFnMCTL** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CANINT** register.

The **CANMSG1INT** register contains the **INTPND** bits of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the **INTPND** bits of the second 16 message objects.

CAN Message 1 Interrupt Pending (CANMSG1INT)

CAN0 base: 0x4004.0000
 CAN1 base: 0x4004.1000
 Offset 0x140
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | INTPND | RO | 0x0000 | Interrupt Pending Bits |
| | | | Value | Description |
| | | | 0 | The corresponding message object is not the source of an interrupt. |
| | | | 1 | The corresponding message object is the source of an interrupt. |

Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160**Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164**

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the **MSGVAL** bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message valid bit of a specific message object can be changed with the **CANIFnARB2** register.

The **CANMSG1VAL** register contains the **MSGVAL** bits of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the **MSGVAL** bits of the second 16 message objects in the message RAM.

CAN Message 1 Valid (CANMSG1VAL)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x160

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | MSGVAL | | | | | | | | | | | | | | | |
| 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 |
|-----------|----------|------|--------|---|
| 31:16 | reserved | RO | 0x0000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 15:0 | MSGVAL | RO | 0x0000 | Message Valid Bits |
| | | | | Value Description |
| | | | | 0 The corresponding message object is not configured and is ignored by the message handler. |
| | | | | 1 The corresponding message object is configured and should be considered by the message handler. |

20 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
 - MDI/MDI-X cross-over support through software assist
 - Register-programmable transmit amplitude
 - Automatic polarity correction and 10BASE-T signal reception
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive channel request asserted on packet receipt
 - Transmit channel request asserted on empty transmit FIFO

20.1 Block Diagram

As shown in Figure 20-1 on page 801, 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 20-1. Ethernet Controller

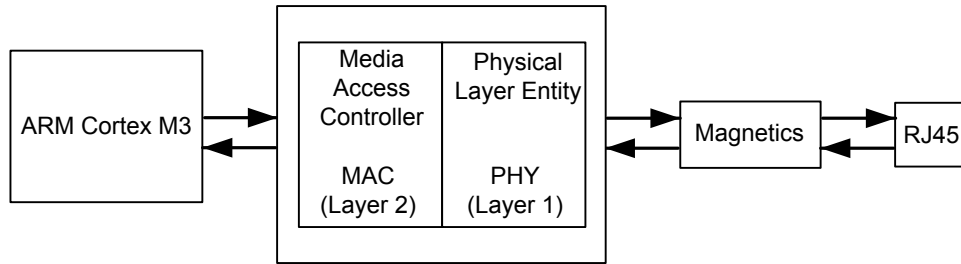
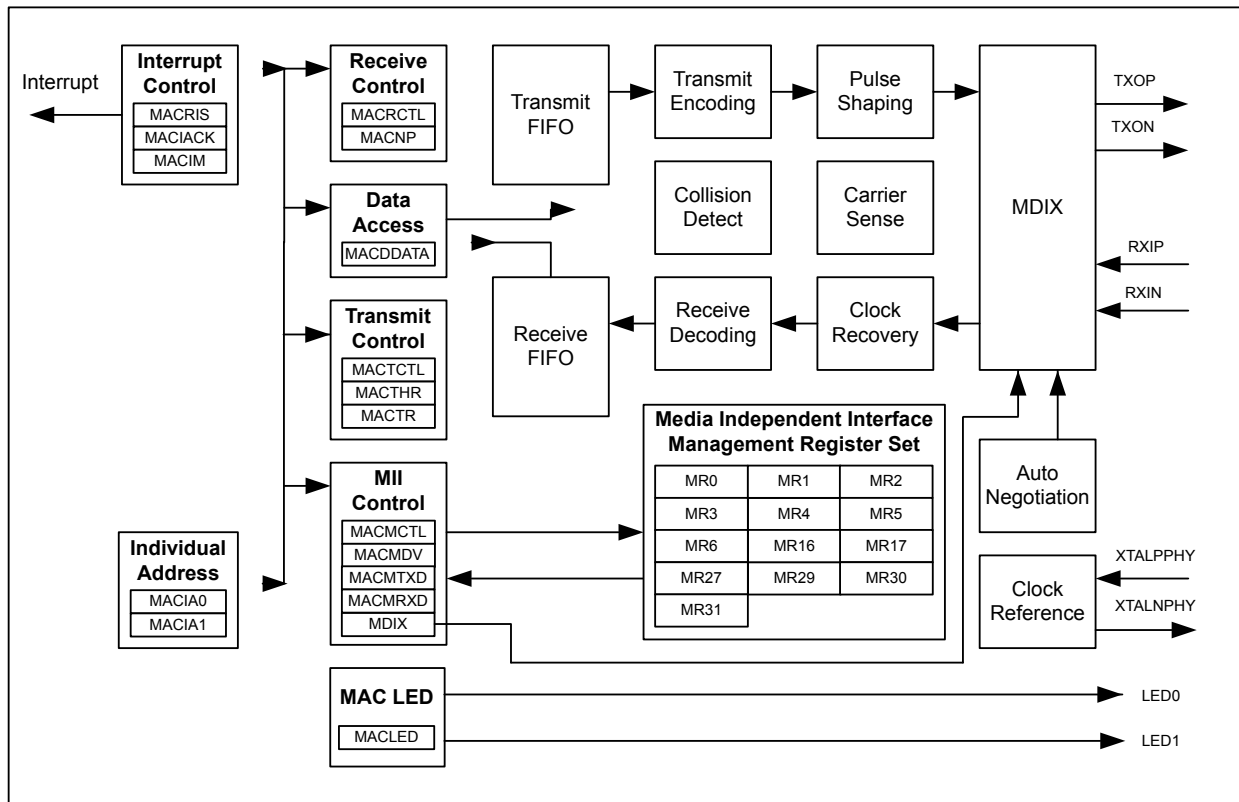


Figure 20-2 on page 801 shows more detail of the internal structure of the Ethernet Controller and how the register set relates to various functions.

Figure 20-2. Ethernet Controller Block Diagram



20.2 Signal Description

Table 20-1 on page 802 lists the external signals of the Ethernet controller and describes the function of each. The LED signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the LED signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the LED function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPTL)** register (page 358) to assign the `LED0` and `LED1` signals to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 20-1. Signals for Ethernet PHY

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|---|
| ERBIAS | 33 | fixed | O | Analog | 12.4-kΩ resistor (1% precision) used internally for Ethernet PHY. |
| LED0 | 59 | PF3 (1) | O | TTL | MII LED 0. |
| LED1 | 60 | PF2 (1) | O | TTL | MII LED 1. |
| MDIO | 58 | fixed | I/O | OD | MDIO of the Ethernet PHY. |
| RXIN | 37 | fixed | I | Analog | RXIN of the Ethernet PHY. |
| RXIP | 40 | fixed | I | Analog | RXIP of the Ethernet PHY. |
| TXON | 46 | fixed | O | TTL | TXON of the Ethernet PHY. |
| TXOP | 43 | fixed | O | TTL | TXOP of the Ethernet PHY. |
| XTALNPHY | 17 | fixed | I | Analog | XTALN of the Ethernet PHY. |
| XTALPPHY | 16 | fixed | O | Analog | XTALP of the Ethernet PHY. |

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

20.3 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.

20.3.1 MAC Operation

The following sections describe the operation of the MAC unit, including an overview of the Ethernet frame format, the MAC layer FIFOs, Ethernet transmission and reception options, and LED indicators.

20.3.1.1 Ethernet Frame Format

Ethernet data is carried by Ethernet frames. The basic frame format is shown in Figure 20-3 on page 803.

Figure 20-3. Ethernet Frame

| | | | | | | |
|------------|-----------|---------------------|----------------|-----------------|--------------------|------------|
| Preamble | SFD | Destination Address | Source Address | Length/ Type | Data | FCS |
| 7 Bytes | 1 Byte | 6 Bytes | 6 Bytes | 2 Bytes | 46 - 1500 Bytes | 4 Bytes |

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 1010.1011b.

- 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 20-2 on page 804) 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 encodes the 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.

20.3.1.2 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 20-2 on page 804. 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 20-2. 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 |

Table 20-2. TX & RX FIFO Organization (continued)

| FIFO Word Read/Write Sequence | Word Bit Fields | TX FIFO (Write) | RX FIFO (Read) |
|-------------------------------|-----------------|---------------------------------|----------------|
| 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 ^a | |
| | 15:8 | FCS 2 ^a | |
| | 23:16 | FCS 3 ^a | |
| | 31:24 | FCS 4 ^a | |

a. 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.

20.3.1.3 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. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.

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.

20.3.1.4 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.

20.3.1.5 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 Ethernet Controller to drive these signals, they must be reconfigured to their hardware function. See “General-Purpose Input/Outputs (GPIOs)” on page 318 for additional details. The function of these pins is programmable using the **Ethernet MAC LED Encoding (MACLED)** register. Refer to page 835 for additional details on how to program these LED functions.

20.3.2 Internal MII Operation

For the MII management interface to function properly, the `MDIO` signal must be connected through a 10 k 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 830 for more details about the use of this register.

20.3.3 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.

20.3.3.1 Clock Selection

The Ethernet Controller can be clocked from an on-chip crystal oscillator which can also be driven by an external oscillator. When using the on-chip crystal oscillator, 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 should be left unconnected. See “Ethernet Controller” on page 1057 for more information regarding the specifications of the Ethernet Controller.

20.3.3.2 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 `SPEED` bit in the **Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31)** register reflects the actual speed. The `AUTODONE` bit in **MR31** is set to indicate that auto-negotiation is complete. Setting the `RANEG` bit in the **MR0** register also causes auto-negotiation to restart.

20.3.3.3 Polarity Correction

The Ethernet Controller is capable of automatic polarity reversal for 10BASE-T and auto-negotiation functions. The `XPOL` bit in the **Ethernet PHY Management Register 27 –Special Control/Status (MR27)** register is set to indicate the polarity has automatically been reversed.

20.3.3.4 MDI/MDI-X Configuration

The Ethernet Controller supports the MDI/MDI-X configuration as defined in *IEEE 802.3-2002 specification* through software assistance. The MDI/MDI-X configuration eliminates the need for cross-over cables when connecting to another device, such as a hub. Software can implement the MDI/MDI-X configuration by using any available timer resource such as SysTick (see “System Timer (SysTick)” on page 74 for more information) to implement this functionality. Once the Ethernet Controller has been configured and enabled, software should check to see if the `LINK` bit in the **Ethernet PHY Management Register 1 - Status (MR1)** has been set within approximately 1 s; if not, set the `EN` bit of the **Ethernet PHY MDIX (MDIX)** register to switch the reverse the transmit and receive lines to the PHY layer.

20.3.3.5 Power Management

The PHY has two power-saving modes:

- Power-Down
- Energy Detect Power-Down

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. When the `PWRDN` bit is cleared, the PHY powers up and is automatically reset.

The energy detect power-down mode is activated by setting the `EDPD` bit in the **MR17** register. In this mode of operation, when no energy is present on the line, the PHY is powered down, except for the management interface, the SQUELCH circuit and the ENERGYON logic. The ENERGYON logic is used to detect the presence of valid energy from 100BASE-T, 10BASE-T, or auto-negotiation signals. While the PHY is powered down, nothing is transmitted. When link pulses or packets are received, the PHY powers-up. The PHY automatically resets itself into the state it had prior to power down and sets the `EONIS` bit in the **MR29** register. The first and possibly the second packet to activate the ENERGYON mode may be lost.

20.3.4 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 Partner Acknowledge
- Parallel Detect Fault
- Page Received

Refer to **Ethernet PHY Management Register 29 - Interrupt Source Flags (MR29)** (see page 852) for additional details regarding PHY interrupts.

20.3.5 DMA Operation

The Ethernet peripheral provides request signals to the μ DMA controller and has a dedicated channel for transmit and one for receive. The request is a single type for both channels. Burst requests are not supported. The RX channel request is asserted when a packet is received while the TX channel request is asserted when the transmit FIFO becomes empty.

No special configuration is needed to enable the Ethernet peripheral for use with the μ DMA controller.

Because the size of a received packet is not known until the header is examined, it is best to set up the initial μ DMA transfer to copy the first 4 words including the packet length plus the Ethernet header from the RX FIFO when the RX request occurs. The μ DMA causes an interrupt when this transfer is complete. Upon entering the interrupt handler, the packet length in the FIFO and the Ethernet header are in a buffer and can be examined. Once the packet length is known, then another μ DMA transfer can be set up to transfer the remaining received packet payload from the FIFO into a buffer. This transfer should be initiated by software. Another interrupt occurs when this transfer is done.

Even though the TX channel generates a TX empty request, the recommended way to handle μ DMA transfers for transmitting packets is to set up the transfer from the buffer containing the packet to the transmit FIFO, and then to initiate the transfer with a software request. An interrupt occurs when this transfer is complete. For both channels, the "auto-request" transfer mode should be used. See "Micro Direct Memory Access (μ DMA)" on page 258 for more details about programming the μ DMA controller.

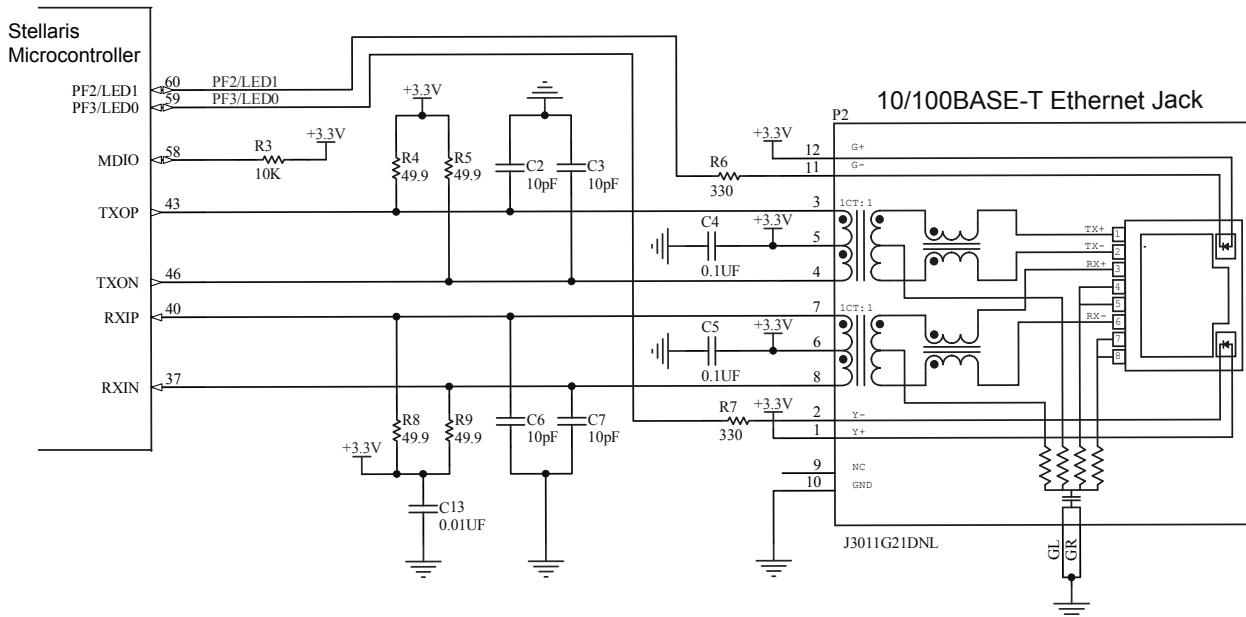
20.4 Initialization and Configuration

The following sections describe the hardware and software configuration required to set up the Ethernet Controller.

20.4.1 Hardware Configuration

Figure 20-4 on page 809 shows the proper method for interfacing the Ethernet Controller to a 10/100BASE-T Ethernet jack.

Figure 20-4. 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

20.4.2 Software Configuration

To use the Ethernet Controller, it must be enabled by setting the `EPHY0` and `EMAC0` bits in the **RCGC2** register (see page 184). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module. See page 184. To find out which GPIO port to enable, refer to Table 24-4 on page 1028. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the Ethernet signals to the appropriate pins. See page 358 and Table 24-5 on page 1035.

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. When the frame (including the FCS field) has been read, the **NPR** field decrements by one. When there are no more frames in the RX FIFO, the **NPR** field reads 0.

20.5 Ethernet Register Map

Table 20-3 on page 810 lists the Ethernet MAC and MII Management registers. The MAC register addresses given are relative to the Ethernet base address of 0x4004.8000. The MII Management registers are accessed using the **MACMCTL** register. Note that the Ethernet controller clocks must be enabled before the registers can be programmed (see page 184).

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 20-3 on page 810 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 20-3. Ethernet Register Map

| Offset | Name | Type | Reset | Description | See page |
|---------------------------------------|----------------|-------|-------------|---|----------|
| Ethernet MAC (Ethernet Offset) | | | | | |
| 0x000 | MACRIS/MACIACK | R/W1C | 0x0000.0000 | Ethernet MAC Raw Interrupt Status/Acknowledge | 813 |
| 0x004 | MACIM | R/W | 0x0000.007F | Ethernet MAC Interrupt Mask | 816 |
| 0x008 | MACRCTL | R/W | 0x0000.0008 | Ethernet MAC Receive Control | 818 |
| 0x00C | MACTCTL | R/W | 0x0000.0000 | Ethernet MAC Transmit Control | 820 |
| 0x010 | MACDATA | R/W | 0x0000.0000 | Ethernet MAC Data | 822 |

Table 20-3. Ethernet Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|---|---------|------|-------------|--|----------|
| 0x014 | MACIA0 | R/W | 0x0000.0000 | Ethernet MAC Individual Address 0 | 824 |
| 0x018 | MACIA1 | R/W | 0x0000.0000 | Ethernet MAC Individual Address 1 | 825 |
| 0x01C | MACTHR | R/W | 0x0000.003F | Ethernet MAC Threshold | 826 |
| 0x020 | MACMCTL | R/W | 0x0000.0000 | Ethernet MAC Management Control | 828 |
| 0x024 | MACMDV | R/W | 0x0000.0080 | Ethernet MAC Management Divider | 830 |
| 0x02C | MACMTXD | R/W | 0x0000.0000 | Ethernet MAC Management Transmit Data | 831 |
| 0x030 | MACMRXD | R/W | 0x0000.0000 | Ethernet MAC Management Receive Data | 832 |
| 0x034 | MACNP | RO | 0x0000.0000 | Ethernet MAC Number of Packets | 833 |
| 0x038 | MACTR | R/W | 0x0000.0000 | Ethernet MAC Transmission Request | 834 |
| 0x040 | MACLED | R/W | 0x0000.0010 | Ethernet MAC LED Encoding | 835 |
| 0x044 | MDIX | R/W | 0x0000.0000 | Ethernet PHY MDIX | 836 |
| MII Management (Accessed through the MACMCTL register) | | | | | |
| - | MR0 | R/W | 0x3100 | Ethernet PHY Management Register 0 – Control | 837 |
| - | MR1 | RO | 0x7849 | Ethernet PHY Management Register 1 – Status | 839 |
| - | MR2 | RO | 0x0161 | Ethernet PHY Management Register 2 – PHY Identifier 1 | 841 |
| - | MR3 | RO | 0xB410 | Ethernet PHY Management Register 3 – PHY Identifier 2 | 842 |
| - | MR4 | R/W | 0x01E1 | Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement | 843 |
| - | MR5 | RO | 0x0000 | Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability | 845 |
| - | MR6 | RO | 0x0000 | Ethernet PHY Management Register 6 – Auto-Negotiation Expansion | 847 |
| - | MR16 | RO | 0x0040 | Ethernet PHY Management Register 16 – Vendor-Specific | 848 |
| - | MR17 | R/W | 0x0002 | Ethernet PHY Management Register 17 – Mode Control/Status | 849 |
| - | MR27 | RO | - | Ethernet PHY Management Register 27 – Special Control/Status | 851 |
| - | MR29 | RO | 0x0000 | Ethernet PHY Management Register 29 – Interrupt Status | 852 |
| - | MR30 | R/W | 0x0000 | Ethernet PHY Management Register 30 – Interrupt Mask | 854 |
| - | MR31 | R/W | 0x00040 | Ethernet PHY Management Register 31 – PHY Special Control/Status | 856 |

20.6 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 836.

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.

Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK)

Base 0x4004.8000

Offset 0x000

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 | | | | | | | | | | PHYINT | MDINT | RXER | FOV | TXEMP | TXER | RXINT |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RW1C | RW1C | RW1C | RW1C | RW1C | RW1C | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|--|
| 31:7 | 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 | RW1C | 0 | PHY Interrupt Value Description 1 An enabled interrupt in the PHY layer has occurred. MR29 in the PHY must be read to determine the specific PHY event that triggered this interrupt. 0 No interrupt. This bit is cleared by writing a 1 to it. |
| 5 | MDINT | RW1C | 0 | MII Transaction Complete Value Description 1 A transaction (read or write) on the MII interface has completed successfully. 0 No interrupt. This bit is cleared by writing a 1 to it. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|---|
| 4 | RXER | RW1C | 0 | <p>Receive Error</p> <p>Value Description</p> <p>1 An error was encountered on the receiver. The possible errors that can cause this interrupt bit to be set are:</p> <ul style="list-style-type: none"> ■ 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. <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to it.</p> |
| 3 | FOV | RW1C | 0 | <p>FIFO Overrun</p> <p>Value Description</p> <p>1 An overrun was encountered on the receive FIFO.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to it.</p> |
| 2 | TXEMP | RW1C | 0 | <p>Transmit FIFO Empty</p> <p>Value Description</p> <p>1 The packet was transmitted and that the TX FIFO is empty.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to it.</p> |
| 1 | TXER | RW1C | 0 | <p>Transmit Error</p> <p>Value Description</p> <p>1 An error was encountered on the transmitter. The possible errors that can cause this interrupt bit to be set are:</p> <ul style="list-style-type: none"> ■ 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. <p>0 No interrupt.</p> <p>Writing a 1 to this bit clears it and resets the TX FIFO write pointer.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|---|
| 0 | RXINT | RW1C | 0 | Packet Received |
| | | | | Value Description |
| | | | | 1 At least one packet has been received and is stored in the receiver FIFO. |
| | | | | 0 No interrupt. |
| | | | | This bit is cleared by writing a 1 to it. |

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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|---------|--------|-------|------|--------|-------|--------|
| | 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 | | | | | | | | | | PHYINTM | MDINTM | RXERM | FOVM | TXEMPM | TXERM | RXINTM |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 | 1 | 1 | 1 | 1 | 1 | 1 | |

| 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 Value Description 1 An interrupt is sent to the interrupt controller when the PHYINT bit in the MACRIS/MACIACK register is set. 0 The PHYINT interrupt is suppressed and not sent to the interrupt controller. |
| 5 | MDINTM | R/W | 1 | Mask MII Transaction Complete Value Description 1 An interrupt is sent to the interrupt controller when the MDINT bit in the MACRIS/MACIACK register is set. 0 The MDINT interrupt is suppressed and not sent to the interrupt controller. |
| 4 | RXERM | R/W | 1 | Mask Receive Error Value Description 1 An interrupt is sent to the interrupt controller when the RXER bit in the MACRIS/MACIACK register is set. 0 The RXER interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 3 | FOVM | R/W | 1 | Mask FIFO Overrun Value Description 1 An interrupt is sent to the interrupt controller when the FOV bit in the MACRIS/MACIACK register is set. 0 The FOV interrupt is suppressed and not sent to the interrupt controller. |
| 2 | TXEMPM | R/W | 1 | Mask Transmit FIFO Empty Value Description 1 An interrupt is sent to the interrupt controller when the TXEMP bit in the MACRIS/MACIACK register is set. 0 The TXEMP interrupt is suppressed and not sent to the interrupt controller. |
| 1 | TXERM | R/W | 1 | Mask Transmit Error Value Description 1 An interrupt is sent to the interrupt controller when the TXER bit in the MACRIS/MACIACK register is set. 0 The TXER interrupt is suppressed and not sent to the interrupt controller. |
| 0 | RXINTM | R/W | 1 | Mask Packet Received Value Description 1 An interrupt is sent to the interrupt controller when the RXINT bit in the MACRIS/MACIACK register is set. 0 The RXINT interrupt is suppressed and not sent to the interrupt controller. |

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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|---------|--------|------|------|------|
| | 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 | | | | | | | | | | | | RSTFIFO | BADCRC | PRMS | AMUL | RXEN |
| Type | RO | 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 | 1 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:5 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4 | RSTFIFO | R/W | 0 | <p>Clear Receive FIFO</p> <p>Value Description</p> <p>1 Clear the receive FIFO. The receive FIFO should be cleared when software initialization is performed.</p> <p>0 No effect.</p> <p>This bit is automatically cleared when read.</p> <p>The receiver should be disabled (<code>RXEN = 0</code>), before a reset is initiated (<code>RSTFIFO = 1</code>). This sequence flushes and resets the RX FIFO.</p> |
| 3 | BADCRC | R/W | 1 | <p>Enable Reject Bad CRC</p> <p>Value Description</p> <p>1 Enables the rejection of frames with an incorrectly calculated CRC. If a bad CRC is encountered, the <code>RXER</code> bit in the MACRIS register is set and the receiver FIFO is reset.</p> <p>0 Disables the rejection of frames with an incorrectly calculated CRC.</p> |
| 2 | PRMS | R/W | 0 | <p>Enable Promiscuous Mode</p> <p>Value Description</p> <p>1 Enables Promiscuous mode, which accepts all valid frames, regardless of the specified Destination Address.</p> <p>0 Disables Promiscuous mode, accepting only frames with the programmed Destination Address.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 1 | AMUL | R/W | 0 | Enable Multicast Frames Value Description 1 Enables the reception of multicast frames. 0 Disables the reception of multicast frames. |
| 0 | RXEN | R/W | 0 | Enable Receiver Value Description 1 Enables the Ethernet receiver. 0 Disables the receiver. 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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|-----|--------|----------|-----|-------|------|
| | 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 | | | | | | | | | | | | DUPLEX | reserved | CRC | PADEN | TXEN |
| Type | RO | RO | RO | RO | 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 |
|-----------|----------|------|------------|--|
| 31:5 | 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 Value Description 1 Enables Duplex mode, allowing simultaneous transmission and reception. 0 Disables Duplex mode. |
| 3 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | CRC | R/W | 0 | Enable CRC Generation Value Description 1 Enables the automatic generation of the CRC and its placement at the end of the packet. 0 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 Value Description 1 Enables the automatic padding of packets that do not meet the minimum frame size. 0 Disables automatic padding. Note that this bit should generally be set. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-----------------------------|
| 0 | TXEN | R/W | 0 | Enable Transmitter |
| | | | | Value Description |
| | | | | 1 Enables the transmitter. |
| | | | | 0 Disables the transmitter. |

Register 5: Ethernet MAC Data (MACDATA), offset 0x010

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 returns indeterminate data and does 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 then auto incremented to the next TX FIFO location. Writing more data into the TX FIFO than indicated in the length field results in the data being lost. Writing less data into the TX FIFO than indicated in the length field results 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 results in the data being lost.

Bytes may not be randomly accessed 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 |

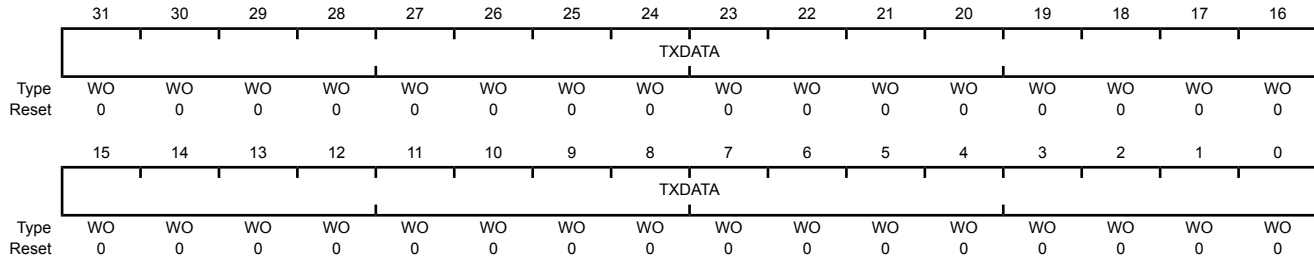
The `RXDATA` bits represent the next word of data stored in the RX FIFO.

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

| | | | | | | | | | | | | | | | | |
|-------|---------|-----|-----|-----|-----|-----|-----|-----|---------|-----|-----|-----|-----|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | MACOCT4 | | | | | | | | MACOCT3 | | | | | | | |
| 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 |
| | MACOCT2 | | | | | | | | MACOCT1 | | | | | | | |
| 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 |
|-----------|---------|------|-------|---|
| 31:24 | 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

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|---------|-----|-----|-----|-----|-----|-----|-----|
| | 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 |
| | MACOCT6 | | | | | | | | MACOCT5 | | | | | | | |
| 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. |
| 15:8 | 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.

Caution – 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 field 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 the THRESH field is clear, 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:

$$\text{Number of Bytes} \geq 4 ((\text{THRESH} \times 8) + 1)$$

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 underrun 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, which 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

| | 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 | | | | | | | | | | THRESH | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | 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 | 1 | 1 | 1 | 1 | 1 | 1 |

| 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 20-3 on page 810 and in “MII Management Register Descriptions” on page 836.

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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|--------|-----|-----|-----|----------|-------|-------|-----|
| | 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 | | | | | | | | REGADR | | | | reserved | WRITE | START | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | 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 |
|-----------|----------|------|-----------|---|
| 31:8 | 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 <code>REGADR</code> bit field represents the MII Management register address for the next MII management interface transaction. Refer to Table 20-3 on page 810 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 Value Description 1 The next operation of the next MII management interface is a write transaction. 0 The next operation of the next MII management interface is a read transaction. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|--|
| 0 | START | R/W | 0 | MII Register Transaction Enable |
| | | | | Value Description |
| | | | 1 | The MII register located at REGADR is read (WRITE=0) or written (WRITE=1). |
| | | | 0 | No effect. |

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:

$$F_{mdc} = \frac{F_{ipclk}}{2 \times (\text{MACMDV} + 1)}$$

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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 | | | | | | | | DIV | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-----------|---|
| 31:8 | reserved | RO | 0x0000.00 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 7:0 | DIV | R/W | 0x80 | Clock Divider The <code>DIV</code> 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

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |
| | MDTX | | | | | | | | | | | | | | | |
| 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. |
| 15:0 | MDTX | R/W | 0x0000 | MII Register Transmit Data The MDTX bits represent the data to 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

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 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 |
| | MDRX | | | | | | | | | | | | | | | |
| 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. |
| 15:0 | 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.

Ethernet MAC Number of Packets (MACNP)

Base 0x4004.8000

Offset 0x034

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|-----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | NPR | | | | |
| 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 |
|-----------|----------|------|-----------|--|
| 31:6 | 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 <code>NPR</code> bits represent the number of packets stored in the RX FIFO. While the <code>NPR</code> field is greater than 0, the <code>RXINT</code> 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|
| | 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 | | | | | | | | | | | | | | | NEWTX |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | NEWTX | R/W | 0 | New Transmission |
| | | | | Value Description |
| | | | | 1 Initiates an Ethernet transmission once the packet has been placed in the TX FIFO. |
| | | | | 0 The transmission has completed. |
| | | | | If early transmission is being used (see the MACTHR register), this bit does not need to be set. |

Register 15: Ethernet MAC LED Encoding (MACLED), offset 0x040

This register enables software to select the source that causes the LED1 and LED0 signal to toggle.

Ethernet MAC LED Encoding (MACLED)

Base 0x4004.8000

Offset 0x040

Type R/W, reset 0x0000.0010

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|------|-----|-----|-----|------|-----|-----|-----|
| | 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 | | | | | | | | LED1 | | | | LED0 | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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. | | | | | | | | | | | | | | | | | | |
| 7:4 | LED1 | R/W | 0x1 | <p>LED1 Source</p> <p>The LED1 field selects the source that toggles the LED1 signal.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Link OK</td> </tr> <tr> <td>0x1</td> <td>RX or TX Activity (Default LED1)</td> </tr> <tr> <td>0x2-0x4</td> <td>Reserved</td> </tr> <tr> <td>0x5</td> <td>100BASE-TX mode</td> </tr> <tr> <td>0x6</td> <td>10BASE-T mode</td> </tr> <tr> <td>0x7</td> <td>Full-Duplex</td> </tr> <tr> <td>0x8</td> <td>Link OK & Blink=RX or TX Activity</td> </tr> <tr> <td>0x9-0xF</td> <td>Reserved</td> </tr> </tbody> </table> | Value | Description | 0x0 | Link OK | 0x1 | RX or TX Activity (Default LED1) | 0x2-0x4 | Reserved | 0x5 | 100BASE-TX mode | 0x6 | 10BASE-T mode | 0x7 | Full-Duplex | 0x8 | Link OK & Blink=RX or TX Activity | 0x9-0xF | Reserved |
| Value | Description | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | Link OK | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | RX or TX Activity (Default LED1) | | | | | | | | | | | | | | | | | | | | | |
| 0x2-0x4 | Reserved | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | 100BASE-TX mode | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | 10BASE-T mode | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Full-Duplex | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | Link OK & Blink=RX or TX Activity | | | | | | | | | | | | | | | | | | | | | |
| 0x9-0xF | Reserved | | | | | | | | | | | | | | | | | | | | | |
| 3:0 | LED0 | R/W | 0x0 | <p>LED0 Source</p> <p>The LED0 field selects the source that toggles the LED0 signal.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Link OK (Default LED0)</td> </tr> <tr> <td>0x1</td> <td>RX or TX Activity</td> </tr> <tr> <td>0x2-0x4</td> <td>Reserved</td> </tr> <tr> <td>0x5</td> <td>100BASE-TX mode</td> </tr> <tr> <td>0x6</td> <td>10BASE-T mode</td> </tr> <tr> <td>0x7</td> <td>Full-Duplex</td> </tr> <tr> <td>0x8</td> <td>Link OK & Blink=RX or TX Activity</td> </tr> <tr> <td>0x9-0xF</td> <td>Reserved</td> </tr> </tbody> </table> | Value | Description | 0x0 | Link OK (Default LED0) | 0x1 | RX or TX Activity | 0x2-0x4 | Reserved | 0x5 | 100BASE-TX mode | 0x6 | 10BASE-T mode | 0x7 | Full-Duplex | 0x8 | Link OK & Blink=RX or TX Activity | 0x9-0xF | Reserved |
| Value | Description | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | Link OK (Default LED0) | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | RX or TX Activity | | | | | | | | | | | | | | | | | | | | | |
| 0x2-0x4 | Reserved | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | 100BASE-TX mode | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | 10BASE-T mode | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Full-Duplex | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | Link OK & Blink=RX or TX Activity | | | | | | | | | | | | | | | | | | | | | |
| 0x9-0xF | Reserved | | | | | | | | | | | | | | | | | | | | | |

Register 16: Ethernet PHY MDIX (MDIX), offset 0x044

This register enables the transmit and receive lines to be reversed in order to implement the MDI/MDI-X functionality. Software can implement the MDI/MDI-X configuration by using any available timer resource such as SysTick (see “System Timer (SysTick)” on page 74 for more information) to implement this functionality. Once the Ethernet Controller has been configured and enabled, software should check to see if the `LINK` bit in the `MR1` register has been set within approximately 1 s; if not, set the `EN` bit of the `MDIX` register to switch the reverse the transmit and receive lines to the PHY layer. Software should check the `LINK` bit again after approximately another 1 s and if no link has been established, the `EN` bit should be cleared. Software must continue to change the termination back and forth by setting and clearing the `EN` bit every 1 s until a link is established.

Ethernet PHY MDIX (MDIX)

Base 0x4004.8000
Offset 0x044
Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | EN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | EN | R/W | 0 | MDI/MDI-X Enable |
| | | | | Value Description |
| | | | | 1 The transmit and receive signals are switched such that data is received on the transmit signals <code>TXOP</code> and <code>TXON</code> ; data is transmitted on the receive signals <code>RXIP</code> and <code>RXIN</code> |
| | | | | 0 No effect. |

20.7 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. The **Ethernet MAC Management Control (MACMCTL)** register is used to access the MII Management registers, see page 828. 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 812.

Register 17: 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

| | | | | | | | | | | | | | | | | |
|-------|-------|--------|---------|--------|-------|-----|-------|--------|------|----------|-----|-----|-----|-----|-----|-----|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | RESET | LOOPBK | SPEEDSL | ANEGEN | PWRDN | ISO | RANEG | DUPLEX | COLT | reserved | | | | | | |
| 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 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 15 | RESET | R/W | 0 | Reset Registers Value Description 1 The PHY layer registers reset to their default state and the internal state machines are reinitialized. 0 No effect. Once the reset operation has completed, this bit is automatically cleared by hardware. |
| 14 | LOOPBK | R/W | 0 | Loopback Mode Value Description 1 Enables the Loopback mode of operation. The receiver ignores external inputs and receives the data that is transmitted by the transmitter. 0 No effect. |
| 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 Value Description 1 Enables the auto-negotiation process. 0 No effect. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 11 | PWRDN | R/W | 0 | <p>Power Down</p> <p>Value Description</p> <p>1 The PHY layer is configured to be in a low-power consuming state. All data on the data inputs is ignored.</p> <p>0 No effect.</p> |
| 10 | ISO | R/W | 0 | <p>Isolate</p> <p>Value Description</p> <p>1 The transmit and receive data paths are isolated and all data being transmitted and received is ignored.</p> <p>0 No effect.</p> |
| 9 | RANEG | R/W | 0 | <p>Restart Auto-Negotiation</p> <p>Value Description</p> <p>1 Restarts the auto-negotiation process.</p> <p>0 No effect.</p> <p>Once the restart has initiated, this bit is automatically cleared by hardware.</p> |
| 8 | DUPLEX | R/W | 1 | <p>Set Duplex Mode</p> <p>Value Description</p> <p>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.</p> <p>0 Enables the Half-Duplex mode of operation.</p> <p>Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.</p> |
| 7 | COLT | R/W | 0 | <p>Collision Test</p> <p>Value Description</p> <p>1 Enables the Collision Test mode of operation.</p> <p>0 No effect.</p> <p>The COLT bit is set after the initiation of a transmission and is cleared once the transmission is halted.</p> |
| 6:0 | reserved | R/W | 0x00 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> <p>These bits should always be written as zero.</p> |

Register 18: 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

| | | | | | | | | | | | | | | | | |
|-------|----------|--------|--------|-------|-------|----------|----|----|----|-------|--------|-------|------|-----|------|----|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | 100X_F | 100X_H | 10T_F | 10T_H | reserved | | | | ANEGC | RFAULT | ANEGA | LINK | JAB | EXTD | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RC | RO | RO | RC | RO |
| Reset | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 15 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 14 | 100X_F | RO | 1 | 100BASE-TX Full-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 100BASE-TX Full-Duplex mode. 0 The Ethernet Controller is not capable of supporting 100BASE-TX Full-Duplex mode. |
| 13 | 100X_H | RO | 1 | 100BASE-TX Half-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 100BASE-TX Half-Duplex mode. 0 The Ethernet Controller is not capable of supporting 100BASE-TX Half-Duplex mode. |
| 12 | 10T_F | RO | 1 | 10BASE-T Full-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 10BASE-T Full-Duplex mode. 0 The Ethernet Controller is not capable of supporting 10BASE-T Full-Duplex mode. |
| 11 | 10T_H | RO | 1 | 10BASE-T Half-Duplex Mode Value Description 1 The Ethernet Controller is capable of supporting 10BASE-T Half-Duplex mode. 0 The Ethernet Controller is not capable of supporting 10BASE-T Half-Duplex mode. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 10:6 | 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 | ANEGC | RO | 0 | <p>Auto-Negotiation Complete</p> <p>Value Description</p> <p>1 The auto-negotiation process has been completed and that the extended registers defined by the auto-negotiation protocol are valid.</p> <p>0 The auto-negotiation process is not complete.</p> |
| 4 | RFAULT | RC | 0 | <p>Remote Fault</p> <p>Value Description</p> <p>1 A remote fault condition has been detected.</p> <p>0 A remote fault condition has not been detected.</p> <p>This bit remains set until it is read, even if the condition no longer exists.</p> |
| 3 | ANEGA | RO | 1 | <p>Auto-Negotiation</p> <p>Value Description</p> <p>1 The Ethernet Controller has the ability to perform auto-negotiation.</p> <p>0 The Ethernet Controller does not have the ability to perform auto-negotiation.</p> |
| 2 | LINK | RO | 0 | <p>Link Made</p> <p>Value Description</p> <p>1 A valid link has been established by the Ethernet Controller.</p> <p>0 A valid link has not been established by the Ethernet Controller.</p> |
| 1 | JAB | RC | 0 | <p>Jabber Condition</p> <p>Value Description</p> <p>1 A jabber condition has been detected by the Ethernet Controller.</p> <p>0 A jabber condition has not been detected by the Ethernet Controller.</p> <p>This bit remains set until it is read, even if the jabber condition no longer exists.</p> |
| 0 | EXTD | RO | 1 | <p>Extended Capabilities</p> <p>Value Description</p> <p>1 The Ethernet Controller provides an extended set of capabilities that can be accessed through the extended register set.</p> <p>0 The Ethernet Controller does not provide extended capabilities.</p> |

Register 19: Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02

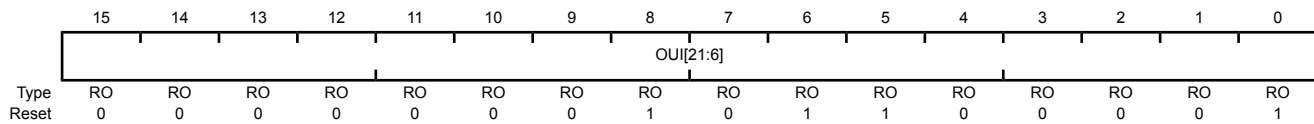
This register, along with **MR3**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2)

Base 0x4004.8000

Address 0x02

Type RO, reset 0x0161



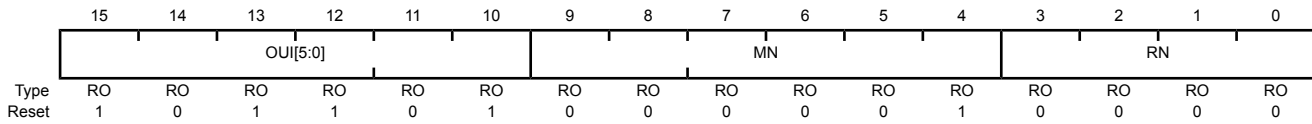
| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------|------|--------|--|
| 15:0 | OUI[21:6] | RO | 0x0161 | Organizationally Unique Identifier[21:6] This field, along with the OUI[5:0] field in MR3 , makes up the Organizationally Unique Identifier indicating the PHY manufacturer. |

Register 20: 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 0xB410



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 15:10 | OUI[5:0] | RO | 0x2D | 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 | 0x01 | Model Number The MN field represents the Model Number of the PHY. |
| 3:0 | RN | RO | 0x0 | Revision Number The RN field represents the Revision Number of the PHY implementation. |

Register 21: 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

| | | | | | | | | | | | | | | | | |
|-------|----|----------|-----|----------|----|----|----|-----|-----|-----|-----|----|----|----|----|----|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | NP | reserved | RF | reserved | | | A3 | A2 | A1 | A0 | S | | | | | |
| Type | RO | RO | R/W | RO | RO | RO | RO | R/W | R/W | R/W | R/W | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 15 | NP | RO | 0 | Next Page Value Description 1 The Ethernet Controller is capable of Next Page exchanges to provide more detailed information on the PHY layer's capabilities. 0 The Ethernet Controller is not capable of Next Page exchanges. |
| 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 Value Description 1 Indicates to the link partner that a Remote Fault condition has been encountered. 0 No 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] Value Description 1 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 <code>RANEG</code> bit in the MR0 register. 0 The Ethernet Controller does not support the 100Base-TX full-duplex signaling protocol. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 7 | A2 | R/W | 1 | <p>Technology Ability Field [2]</p> <p>Value Description</p> <p>1 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 <code>RANEG</code> bit in the MRO register.</p> <p>0 The Ethernet Controller does not support the 100Base-TX half-duplex signaling protocol.</p> |
| 6 | A1 | R/W | 1 | <p>Technology Ability Field [1]</p> <p>Value Description</p> <p>1 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 <code>RANEG</code> bit in the MRO register.</p> <p>0 The Ethernet Controller does not support the 10BASE-T full-duplex signaling protocol.</p> |
| 5 | A0 | R/W | 1 | <p>Technology Ability Field [0]</p> <p>Value Description</p> <p>1 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 <code>RANEG</code> bit in the MRO register.</p> <p>0 The Ethernet Controller does not support the 10BASE-T half-duplex signaling protocol.</p> |
| 4:0 | S | RO | 0x1 | <p>Selector Field</p> <p>This field encodes 32 possible messages for communicating between Ethernet Controllers. This field is hard-coded to 0x01, indicating that the Stellaris® Ethernet Controller is <i>IEEE 802.3</i> compliant.</p> |

Register 22: Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05

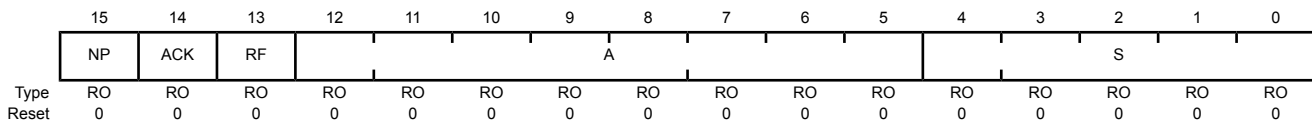
This register provides the advertised abilities of the link partner's Ethernet Controller that are received and stored during auto-negotiation.

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 |
| | | | | Value Description |
| | | | | 1 The link partner's Ethernet Controller is capable of Next page exchanges to provide more detailed information on the Ethernet Controller's capabilities. |
| | | | | 0 The link partner's Ethernet Controller is not capable of Next page exchanges |
| 14 | ACK | RO | 0 | Acknowledge |
| | | | | Value Description |
| | | | | 1 The Ethernet Controller has successfully received the link partner's advertised abilities during auto-negotiation. |
| | | | | 0 The Ethernet Controller has not received the link partner's advertised abilities during auto-negotiation. |
| 13 | RF | RO | 0 | Remote Fault |
| | | | | Value Description |
| | | | | 1 The link partner is indicating that a Remote Fault condition has been encountered. |
| | | | | 0 The link partner is not indicating that a Remote Fault condition has been encountered. |
| 12:5 | A | RO | 0x00 | Technology Ability Field |
| | | | | This 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. |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | |
|-----------|--------------------------|------|-------|--|-------|-------------|------|----------|------|----------------|------|--------------------------|------|----------------|------|---------------|-----------|----------|
| 4:0 | S | RO | 0x00 | Selector Field This field encodes possible messages for communicating between Ethernet Controllers. | | | | | | | | | | | | | | |
| | | | | <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x00</td><td>Reserved</td></tr><tr><td>0x01</td><td>IEEE Std 802.3</td></tr><tr><td>0x02</td><td>IEEE Std 802.9 ISLAN-16T</td></tr><tr><td>0x03</td><td>IEEE Std 802.5</td></tr><tr><td>0x04</td><td>IEEE Std 1394</td></tr><tr><td>0x05–0x1F</td><td>Reserved</td></tr></tbody></table> | 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 |
| 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 23: Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06

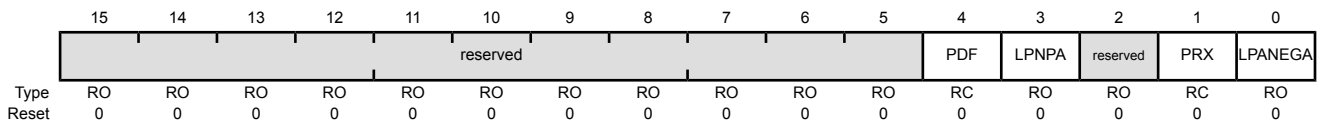
This register enables software to determine the auto-negotiation and next page capabilities of the Ethernet Controller and the link partner after auto-negotiation.

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 Value Description 1 More than one technology was detected at link up. 0 Only one technology was detected at link up. This bit is automatically cleared when read. |
| 3 | LPNPA | RO | 0 | Link Partner is Next Page Able Value Description 1 The link partner is enabled to support next page. 0 The link partner is not 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 Value Description 1 A new page has been received from the link partner and stored. 0 A new page has not been received. This bit is automatically cleared when read. |
| 0 | LPANEGA | RO | 0 | Link Partner is Auto-Negotiation Able Value Description 1 The link partner is enabled to support auto-negotiation. 0 The link partner is not enabled to support auto-negotiation. |

Register 24: Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10

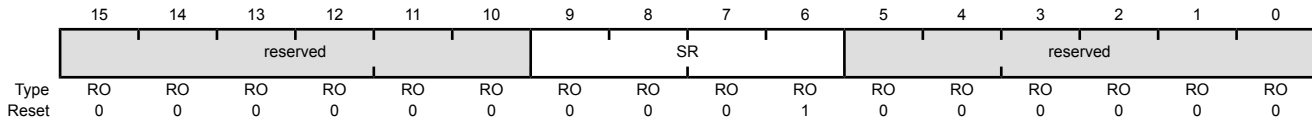
This register contains a silicon revision identifier.

Ethernet PHY Management Register 16 – Vendor-Specific (MR16)

Base 0x4004.8000

Address 0x10

Type RO, reset 0x0040



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 15:10 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9:6 | SR | RO | 0x1 | Silicon Revision Identifier This field contains the four-bit identifier for the silicon revision. |
| 5:0 | 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 25: Ethernet PHY Management Register 17 – Mode Control/Status (MR17), address 0x11

This register provides the means for controlling and observing various PHY layer modes.

Ethernet PHY Management Register 17 – Mode Control/Status (MR17)

Base 0x4004.8000

Address 0x11

Type R/W, reset 0x0002

| | | | | | | | | | | | | | | | | |
|-------|----------|---------|------|----------|------|----------|---------|----------|----------|----------|----------|----------|----------|------|------|----------|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | FASTRIP | EDPD | reserved | LSQE | reserved | FASTEST | reserved | reserved | reserved | reserved | reserved | reserved | FGLS | ENON | reserved |
| Type | R/W | R/W | R/W | R/W | R/W | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 15 | reserved | R/W | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. Important: This bit must always be written with a 0 to ensure proper operation. |
| 14 | FASTRIP | R/W | 0 | 10-BASE-T Fast Mode Enable Value Description 1 Enables PHYT_10 test mode. 0 No effect. |
| 13 | EDPD | R/W | 0 | Enable Energy Detect Power Down Value Description 1 Enables the Energy Detect Power Down mode. 0 No effect. |
| 12 | reserved | R/W | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. Important: This bit must always be written with a 0 to ensure proper operation. |
| 11 | LSQE | R/W | 0 | Low Squelch Enable Value Description 1 Enables a lower threshold meaning more sensitivity to the signal levels. 0 No effect. |
| 10:9 | 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. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 8 | FASTEST | R/W | 0 | <p>Auto-Negotiation Test Mode</p> <p>Value Description</p> <p>1 Enables the Auto-Negotiation Test mode.</p> <p>0 No effect.</p> |
| 7:3 | reserved | R/W | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> <hr/> <p>Important: This bit must always be written with a 0 to ensure proper operation.</p> <hr/> |
| 2 | FGLS | R/W | 0 | <p>Force Good Link Status</p> <p>Value Description</p> <p>1 Forces the 100BASE-T link to be active.</p> <p>0 No effect.</p> <p>Note: This bit should only be set when testing.</p> |
| 1 | ENON | RO | 1 | <p>Energy On</p> <p>Value Description</p> <p>1 Energy is detected on the line.</p> <p>0 Valid energy has not been detected on the line within 256 ms.</p> <p>This bit is set by a hardware reset, but is unaffected by a software reset.</p> |
| 0 | reserved | R/W | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> <hr/> <p>Important: This bit must always be written with a 0 to ensure proper operation.</p> <hr/> |

Register 26: Ethernet PHY Management Register 27 – Special Control/Status (MR27), address 0x1B

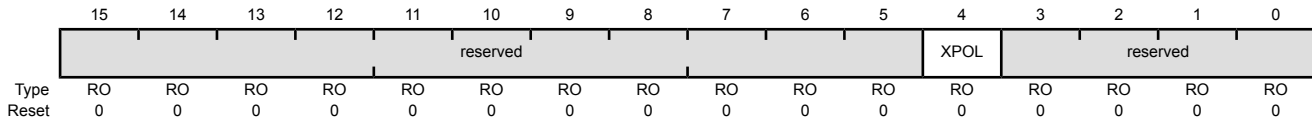
This register shows the status of the 10BASE-T polarity.

Ethernet PHY Management Register 27 – Special Control/Status (MR27)

Base 0x4004.8000

Address 0x1B

Type RO, reset -



| 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 | XPOL | RO | 0 | Polarity State of 10 BASE-T Value Description 1 The 10BASE-T is reversed polarity. 0 The 10BASE-T is normal polarity. |
| 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 27: Ethernet PHY Management Register 29 – Interrupt Status (MR29), address 0x1D

This register contains information about the source of PHY layer interrupts. Reading this register clears any bits that are set. The `PHYINT` bit is set in the `MACRIS/MACIACK` register whenever any of the bits in this register are set.

Ethernet PHY Management Register 29 – Interrupt Status (MR29)

Base 0x4004.8000
Address 0x1D
Type RO, reset 0x0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-------|----------|--------|------|---------|-------|-------|----------|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | EONIS | ANCOMPIS | RFLTIS | LDIS | LPACKIS | PDFIS | PRXIS | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 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 | EONIS | RO | 0 | ENERGYON Interrupt Value Description 1 An interrupt has been generated due to the <code>ENON</code> bit being set in the <code>MR17</code> register. 0 No interrupt. This bit is cleared by reading the value. |
| 6 | ANCOMPIS | RO | 0 | Auto-Negotiation Complete Interrupt Value Description 1 An interrupt has been generated due to the completion of auto negotiation. 0 No interrupt. This bit is cleared by reading the value. |
| 5 | RFLTIS | RO | 0 | Remote Fault Interrupt Value Description 1 An interrupt has been generated due to the detection of a Remote Fault. 0 No interrupt. This bit is cleared by reading the value. |
| 4 | LDIS | RO | 0 | Link Down Interrupt Value Description 1 An interrupt has been generated due to the negation of Link Status. 0 No interrupt. This bit is cleared by reading the value. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 3 | LPACKIS | RO | 0 | <p>Auto-Negotiation LP Acknowledge</p> <p>Value Description</p> <p>1 An interrupt has been generated due to the reception of an acknowledge message from the link partner during auto-negotiation.</p> <p>0 No interrupt.</p> <p>This bit is cleared by reading the value.</p> |
| 2 | PDFIS | RO | 0 | <p>Parallel Detection Fault</p> <p>Value Description</p> <p>1 An interrupt has been generated due to the detection of a parallel detection fault during auto negotiation.</p> <p>0 No interrupt.</p> <p>This bit is cleared by reading the value.</p> |
| 1 | PRXIS | RO | 0 | <p>Auto Negotiation Page Received</p> <p>Value Description</p> <p>1 An interrupt has been generated due to the reception of an auto negotiation page from the link partner.</p> <p>0 No interrupt.</p> <p>This bit is cleared by reading the value.</p> |
| 0 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |

Register 28: Ethernet PHY Management Register 30 – Interrupt Mask (MR30), address 0x1E

This register enables interrupts to be generated by the various sources of PHY layer interrupts.

Ethernet PHY Management Register 30 – Interrupt Mask (MR30)

Base 0x4004.8000

Address 0x1E

Type R/W, reset 0x0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|-------|----------|--------|------|---------|-------|-------|----------|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | EONIM | ANCOMPIM | RFLTIM | LDIM | LPACKIM | PDFIM | PRXIM | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 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 | EONIM | R/W | 0 | ENERGYON Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the EONIS bit in the MR29 register is set. 0 The EONIS interrupt is suppressed and not sent to the interrupt controller. |
| 6 | ANCOMPIM | R/W | 0 | Auto-Negotiation Complete Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the ANCOMPIS bit in the MR29 register is set. 0 The ANCOMPIS interrupt is suppressed and not sent to the interrupt controller. |
| 5 | RFLTIM | R/W | 0 | Remote Fault Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the RFLTIS bit in the MR29 register is set. 0 The RFLTIS interrupt is suppressed and not sent to the interrupt controller. |
| 4 | LDIM | R/W | 0 | Link Down Interrupt Enabled Value Description 1 An interrupt is sent to the interrupt controller when the LDIS bit in the MR29 register is set. 0 The LDIS interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 3 | LPACKIM | R/W | 0 | Auto-Negotiation LP Acknowledge Enabled Value Description 1 An interrupt is sent to the interrupt controller when the LPACKIS bit in the MR29 register is set. 0 The LPACKIS interrupt is suppressed and not sent to the interrupt controller. |
| 2 | PDFIM | R/W | 0 | Parallel Detection Fault Enabled Value Description 1 An interrupt is sent to the interrupt controller when the PDFIS bit in the MR29 register is set. 0 The PDFIS interrupt is suppressed and not sent to the interrupt controller. |
| 1 | PRXIM | R/W | 0 | Auto Negotiation Page Received Enabled Value Description 1 An interrupt is sent to the interrupt controller when the PRXIS bit in the MR29 register is set. 0 The PRXIS interrupt is suppressed and not sent to the interrupt controller. |
| 0 | reserved | R/W | 0 | Software should not rely on the value of 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: Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31), address 0x1F

This register provides special control and status for the PHY layer.

Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31)

Base 0x4004.8000

Address 0x1F

Type R/W, reset 0x00040

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

| Bit/Field | Name | Type | Reset | Description |
|--|----------|------|-------|---|
| 15:13 | reserved | R/W | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| Important: This bit field must always be written with a 0 to ensure proper operation. | | | | |
| 12 | AUTODONE | RO | 0 | Auto Negotiation Done |
| | | | | Value Description |
| | | | | 1 Auto negotiation is complete. |
| | | | | 0 Auto negotiation is not complete. |
| 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. |
| 4:2 | SPEED | RO | 0x0 | HCD Speed Value |
| | | | | Value Description |
| | | | | 0x0 Reserved |
| | | | | 0x1 10BASE-T half duplex |
| | | | | 0x2 100BASE-T half duplex |
| | | | | 0x3-0x4 Reserved |
| | | | | 0x5 10BASE-T full duplex |
| | | | | 0x6 100BASE-T full duplex |
| | | | | 0x7 Reserved |
| 1 | reserved | R/W | 0 | Software should not rely on the value of 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 | SCRDIS | R/W | 0 | Scramble Disable |
| | | | | Value Description |
| | | | | 1 Disables data scrambling. |
| | | | | 0 Enables data scrambling. |

21 Universal Serial Bus (USB) Controller

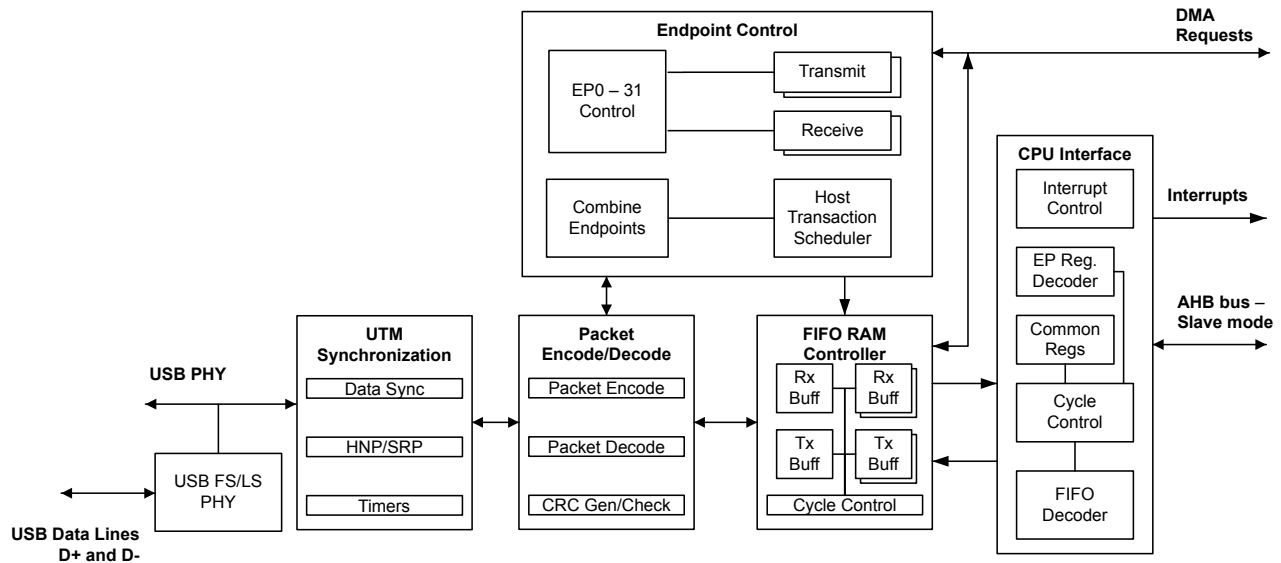
The Stellaris® USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB Host, Device, or OTG functions. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. 32 endpoints including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus 30 endpoints defined by firmware along with a dynamic sizable FIFO support multiple packet queueing. μ DMA access to the FIFO allows minimal interference from system software. Software-controlled connect and disconnect allows flexibility during USB device start-up. The controller complies with OTG standard's session request protocol (SRP) and host negotiation protocol (HNP).

The Stellaris® USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation
- Integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 32 endpoints
 - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
 - 15 configurable IN endpoints and 15 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop and valid ID detection and interrupt
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive for up to three IN endpoints and three OUT endpoints
 - Channel requests asserted when FIFO contains required amount of data

21.1 Block Diagram

Figure 21-1. USB Module Block Diagram



21.2 Signal Description

Table 21-1 on page 858 lists the external signals of the USB controller and describes the function of each. Some USB controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these USB signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the USB function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPTL)** register (page 358) to assign the USB signal to the specified GPIO port pin. The `USB0VBUS` and `USB0ID` signals are configured by clearing the appropriate `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 21-1. Signals for USB

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|----------------------------|---|----------|--------------------------|---|
| USB0DM | 70 | fixed | I/O | Analog | Bidirectional differential data pin (D- per USB specification). |
| USB0DP | 71 | fixed | I/O | Analog | Bidirectional differential data pin (D+ per USB specification). |
| USB0EPEN | 19 24 34 72 83 | PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4) | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |

Table 21-1. Signals for USB (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|-----------|--|---|----------|--------------------------|---|
| USB0ID | 66 | PB0 | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is an A device and pulled up is a B device). |
| USB0PFLT | 22 23 35 65 74 76 87 | PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9) | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| USB0RBIAS | 73 | fixed | O | Analog | 9.1-kΩ resistor (1% precision) used internally for USB analog circuitry. |
| USB0VBUS | 67 | PB1 | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |

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

21.3 Functional Description

Note: A 9.1-kΩ resistor should be connected between the `USB0RBIAS` and ground. The 9.1-kΩ resistor should have a 1% tolerance and should be located in close proximity to the `USB0RBIAS` pin. Power dissipation in the resistor is low, so a chip resistor of any geometry may be used.

The Stellaris® USB controller provides full OTG negotiation by supporting both the session request protocol (SRP) and the host negotiation protocol (HNP). The session request protocol allows devices on the B side of a cable to request the A side device turn on VBUS. The host negotiation protocol is used after the initial session request protocol has powered the bus and provides a method to determine which end of the cable will act as the Host controller. When the device is connected to non-OTG peripherals or devices, the controller can detect which cable end was used and provides a register to indicate if the controller should act as the Host or the Device controller. This indication and the mode of operation are handled automatically by the USB controller. This auto-detection allows the system to use a single A/B connector instead of having both A and B connectors in the system and supports full OTG negotiations with other OTG devices.

In addition, the USB controller provides support for connecting to non-OTG peripherals or Host controllers. The USB controller can be configured to act as either a dedicated Host or Device, in which case, the `USB0VBUS` and `USB0ID` signals can be used as GPIOs. However, when the USB controller is acting as a self-powered Device, a GPIO input or analog comparator input must be connected to VBUS and configured to generate an interrupt when the VBUS level drops. This interrupt is used to disable the pullup resistor on the `USB0DP` signal.

Note: When USB is used in the system, the minimum system frequency is 20 MHz.

21.3.1 Operation as a Device

This section describes the Stellaris® USB controller's actions when it is being used as a USB Device. Before the USB controller's operating mode is changed from Device to Host or Host to Device, software must reset the USB controller by setting the `USB0` bit in the **Software Reset Control 2**

(**SRCR2**) register (see page 198). IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and recognition of Start of Frame (SOF) are all described.

When in Device mode, IN transactions are controlled by an endpoint's transmit interface and use the transmit endpoint registers for the given endpoint. OUT transactions are handled with an endpoint's receive interface and use the receive endpoint registers for the given endpoint.

When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint for a USB Device. However, in most cases the USB Device should use the dedicated control endpoint on the USB controller's endpoint 0.

21.3.1.1 Endpoints

When operating as a Device, the USB controller provides two dedicated control endpoints (IN and OUT) and 30 configurable endpoints (15 IN and 15 OUT) that can be used for communications with a Host controller. The endpoint number and direction associated with an endpoint is directly related to its register designation. For example, when the Host is transmitting to endpoint 1, all configuration and data is in the endpoint 1 transmit register interface.

Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the USB controller's FIFO RAM as a shared memory for both IN and OUT transactions.

The remaining 30 endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as 15 configurable IN and 15 configurable OUT endpoints. The endpoint pairs are not required to have the same type for their IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair could be a bulk endpoint, while the IN portion of that endpoint pair could be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.

21.3.1.2 IN Transactions as a Device

When operating as a USB Device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the 15 configurable IN endpoints are determined by the **USB Transmit FIFO Start Address (USBTXFIFOADD)** register. The maximum size of a data packet that may be placed in a transmit endpoint's FIFO for transmission is programmable and is determined by the value written to the **USB Maximum Transmit Data Endpoint n (USBTXMAXPn)** register for that endpoint. The endpoint's FIFO can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size.

Note: The maximum packet size set for any endpoint must not exceed the FIFO size. The **USBTXMAXPn** register should not be written to while data is in the FIFO as unexpected results may occur.

Single-Packet Buffering

If the size of the transmit endpoint's FIFO is less than twice the maximum packet size for this endpoint (as set in the **USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ)** register), only one packet can be buffered in the FIFO and single-packet buffering is required. When each packet is completely loaded into the transmit FIFO, the **TXRDY** bit in the **USB Transmit Control and Status Endpoint n Low (USBTXCSRLn)** register must be set. If the **AUTOSET** bit in the **USB Transmit Control and Status Endpoint n High (USBTXCSRHn)** register is set, the **TXRDY** bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, the **TXRDY** bit must be set manually. When the **TXRDY** bit is set, either manually or automatically, the packet is ready to be sent. When the packet has been successfully sent, both **TXRDY** and **FIFONE** are cleared, and the appropriate transmit endpoint interrupt signaled. At this point, the next packet can be loaded into the FIFO.

Double-Packet Buffering

If the size of the transmit endpoint's FIFO is at least twice the maximum packet size for this endpoint, two packets can be buffered in the FIFO and double-packet buffering is allowed. As each packet is loaded into the transmit FIFO, the **TXRDY** bit in the **USBTXCSRLn** register must be set. If the **AUTOSET** bit in the **USBTXCSRHn** register is set, the **TXRDY** bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, **TXRDY** must be set manually. When the **TXRDY** bit is set, either manually or automatically, the packet is ready to be sent. After the first packet is loaded, **TXRDY** is immediately cleared and an interrupt is generated. A second packet can now be loaded into the transmit FIFO and **TXRDY** set again (either manually or automatically if the packet is the maximum size). At this point, both packets are ready to be sent. After each packet has been successfully sent, **TXRDY** is automatically cleared and the appropriate transmit endpoint interrupt signaled to indicate that another packet can now be loaded into the transmit FIFO. The state of the **FIFONE** bit in the **USBTXCSRLn** register at this point indicates how many packets may be loaded. If the **FIFONE** bit is set, then another packet is in the FIFO and only one more packet can be loaded. If the **FIFONE** bit is clear, then no packets are in the FIFO and two more packets can be loaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding **EPn** bit is set in the **USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

21.3.1.3 OUT Transactions as a Device

When in Device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the 15 configurable OUT endpoints are determined by the **USB Receive FIFO Start Address (USBRXFIFOADD)** register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the **USB Maximum Receive Data Endpoint n (USBRXMAXPn)** register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size.

Note: In all cases, the maximum packet size must not exceed the FIFO size.

Single-Packet Buffering

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the **RXRDY** and **FULL** bits in the **USB Receive Control and Status Endpoint n Low (USBRXCSRLn)** register are set and the appropriate receive endpoint is signaled, indicating that a packet can now be unloaded from the FIFO. After the packet

has been unloaded, the `RXRDY` bit must be cleared in order to allow further packets to be received. This action also generates the acknowledge signaling to the Host controller. If the `AUTOCL` bit in the **USB Receive Control and Status Endpoint n High (USBXCSRHn)** register is set and a maximum-sized packet is unloaded from the FIFO, the `RXRDY` and `FULL` bits are cleared automatically. For packet sizes less than the maximum, `RXRDY` must be cleared manually.

Double-Packet Buffering

If the size of the receive endpoint FIFO is at least twice the maximum packet size for the endpoint, two data packets can be buffered and double-packet buffering can be used. When the first packet is received and loaded into the receive FIFO, the `RXRDY` bit in the **USBXCSRLn** register is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

Note: The `FULL` bit in **USBXCSRLn** is not set when the first packet is received. It is only set if a second packet is received and loaded into the receive FIFO.

After each packet has been unloaded, the `RXRDY` bit must be cleared to allow further packets to be received. If the `AUTOCL` bit in the **USBXCSRHn** register is set and a maximum-sized packet is unloaded from the FIFO, the `RXRDY` bit is cleared automatically. For packet sizes less than the maximum, `RXRDY` must be cleared manually. If the `FULL` bit is set when `RXRDY` is cleared, the USB controller first clears the `FULL` bit, then sets `RXRDY` again to indicate that there is another packet waiting in the FIFO to be unloaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding `EPn` bit is set in the **USB Receive Double Packet Buffer Disable (USBXDPKTBUFDIS)** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

21.3.1.4 Scheduling

The Device has no control over the scheduling of transactions as scheduling is determined by the Host controller. The Stellaris® USB controller can set up a transaction at any time. The USB controller waits for the request from the Host controller and generates an interrupt when the transaction is complete or if it was terminated due to some error. If the Host controller makes a request and the Device controller is not ready, the USB controller sends a busy response (NAK) to all requests until it is ready.

21.3.1.5 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the Host controller: when the USB controller automatically stalls a control transfer and unexpected zero length OUT data packets.

Stalled Control Transfer

The USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The Host sends more data during an OUT data phase of a control transfer than was specified in the Device request during the SETUP phase. This condition is detected by the USB controller when the Host sends an OUT token (instead of an IN token) after the last OUT packet has been unloaded and the `DATAEND` bit in the **USB Control and Status Endpoint 0 Low (USBCSRL0)** register has been set.
2. The Host requests more data during an IN data phase of a control transfer than was specified in the Device request during the SETUP phase. This condition is detected by the USB controller

when the Host sends an IN token (instead of an OUT token) after the CPU has cleared `TXRDY` and set `DATAEND` in response to the ACK issued by the Host to what should have been the last packet.

3. The Host sends more than **USBRXMAXP_n** bytes of data with an OUT data token.
4. The Host sends more than a zero length data packet for the OUT STATUS phase.

Zero Length OUT Data Packets

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets should only be received after the entire length of the Device request has been transferred.

However, if the Host sends a zero-length OUT data packet before the entire length of Device request has been transferred, it is signaling the premature end of the transfer. In this case, the USB controller automatically flushes any IN token ready for the data phase from the FIFO and sets the `SETUP` bit in the **USBCSRL0** register.

Setting the Device Address

When a Host is attempting to enumerate the USB Device, it requests that the Device change its address from zero to some other value. The address is changed by writing the value that the Host requested to the **USB Device Functional Address (USBFADDR)** register. However, care should be taken when writing to **USBFADDR** to avoid changing the address before the transaction is complete. This register should only be set after the `SET_ADDRESS` command is complete. Like all control transactions, the transaction is only complete after the Device has left the STATUS phase. In the case of a `SET_ADDRESS` command, the transaction is completed by responding to the IN request from the Host with a zero-byte packet. Once the Device has responded to the IN request, the **USBFADDR** register should be programmed to the new value as soon as possible to avoid missing any new commands sent to the new address.

Note: If the **USBFADDR** register is set to the new value as soon as the Device receives the OUT transaction with the `SET_ADDRESS` command in the packet, it changes the address during the control transfer. In this case, the Device does not receive the IN request that allows the USB transaction to exit the STATUS phase of the control transfer because it is sent to the old address. As a result, the Host does not get a response to the IN request, and the Host fails to enumerate the Device.

21.3.1.6 Device Mode SUSPEND

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters SUSPEND mode. If the SUSPEND interrupt has been enabled in the **USB Interrupt Enable (USBIE)** register, an interrupt is generated at this time. When in SUSPEND mode, the PHY also goes into SUSPEND mode. When RESUME signaling is detected, the USB controller exits SUSPEND mode and takes the PHY out of SUSPEND. If the RESUME interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit SUSPEND mode by setting the `RESUME` bit in the **USB Power (USBPOWER)** register. When this bit is set, the USB controller exits SUSPEND mode and drives RESUME signaling onto the bus. The `RESUME` bit must be cleared after 10 ms (a maximum of 15 ms) to end RESUME signaling.

To meet USB power requirements, the controller can be put into Deep Sleep mode which keeps the controller in a static state. The USB controller is not able to Hibernate because all the internal states are lost as a result.

21.3.1.7 Start-of-Frame

When the USB controller is operating in Device mode, it receives a Start-Of-Frame (SOF) packet from the Host once every millisecond. When the SOF packet is received, the 11-bit frame number contained in the packet is written into the **USB Frame Value (USBFRAME)** register, and an SOF interrupt is also signaled and can be handled by the application. Once the USB controller has started to receive SOF packets, it expects one every millisecond. If no SOF packet is received after 1.00358 ms, the packet is assumed to have been lost, and the **USBFRAME** register is not updated. The USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.

21.3.1.8 USB RESET

When the USB controller is in Device mode and a RESET condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the **USBFADDR** register.
- Clears the **USB Endpoint Index (USBEPIDX)** register.
- Flushes all endpoint FIFOs.
- Clears all control/status registers.
- Enables all endpoint interrupts.
- Generates a RESET interrupt.

When the application software driving the USB controller receives a RESET interrupt, any open pipes are closed and the USB controller waits for bus enumeration to begin.

21.3.1.9 Connect/Disconnect

The USB controller connection to the USB bus is handled by software. The USB PHY can be switched between normal mode and non-driving mode by setting or clearing the **SOFTCONN** bit of the **USBPOWER** register. When the **SOFTCONN** bit is set, the PHY is placed in its normal mode, and the **USB0DP/USB0DM** lines of the USB bus are enabled. At the same time, the USB controller is placed into a state, in which it does not respond to any USB signaling except a USB RESET.

When the **SOFTCONN** bit is cleared, the PHY is put into non-driving mode, **USB0DP** and **USB0DM** are tristated, and the USB controller appears to other devices on the USB bus as if it has been disconnected. The non-driving mode is the default so the USB controller appears disconnected until the **SOFTCONN** bit has been set. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete, and the system is ready to perform enumeration before connecting to the USB bus. Once the **SOFTCONN** bit has been set, the USB controller can be disconnected by clearing this bit.

Note: The USB controller does not generate an interrupt when the Device is connected to the Host. However, an interrupt is generated when the Host terminates a session.

21.3.2 Operation as a Host

When the Stellaris[®] USB controller is operating in Host mode, it can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Before the USB controller's operating mode is changed from Host to Device or

Device to Host, software must reset the USB controller by setting the `USB0` bit in the **Software Reset Control 2 (SRCR2)** register (see page 198). Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, isochronous, and interrupt transactions are supported. This section describes the USB controller's actions when it is being used as a USB Host. Configuration of IN endpoints, OUT endpoints, entry into and exit from SUSPEND mode, and RESET are all described.

When in Host mode, IN transactions are controlled by an endpoint's receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint registers for a given endpoint. As in Device mode, the FIFOs for endpoints should take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint to communicate with a Device. However, in most cases the USB controller should use the dedicated control endpoint to communicate with a Device's endpoint 0.

21.3.2.1 Endpoints

The endpoint registers are used to control the USB endpoint interfaces which communicate with Device(s) that are connected. The endpoints consist of a dedicated control IN endpoint, a dedicated control OUT endpoint, 15 configurable OUT endpoints, and 15 configurable IN endpoints.

The dedicated control interface can only be used for control transactions to endpoint 0 of Devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of Devices. This control endpoint shares the first 64 bytes of the USB controller's FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, interrupt, or isochronous Device endpoints.

These USB interfaces can be used to simultaneously schedule as many as 15 independent OUT and 15 independent IN transactions to any endpoints on any Device. The IN and OUT controls are paired in three sets of registers. However, they can be configured to communicate with different types of endpoints and different endpoints on Devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a Device's bulk OUT endpoint 1, while the IN portion is communicating with a Device's interrupt IN endpoint 2.

Before accessing any Device, whether for point-to-point communications or for communications via a hub, the relevant **USB Receive Functional Address Endpoint n (USBRXFUNCADDRn)** or **USB Transmit Functional Address Endpoint n (USBTXFUNCADDRn)** registers must be set for each receive or transmit endpoint to record the address of the Device being accessed.

The USB controller also supports connections to Devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. Customization includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

21.3.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in Device mode except that the transaction first must be initiated by setting the `REQPKT` bit in the **USBCSRL0** register, indicating to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target Device. When the packet is received and placed in the receive FIFO, the `RXRDY` bit in the **USBCSRL0** register is set, and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, `RXRDY` must be cleared. The `AUTOCL` bit in the **USBXCSRHn** register can be used to have `RXRDY` automatically cleared when a maximum-sized packet has been unloaded from the FIFO. The `AUTORQ` bit in **USBXCSRHn** causes the `REQPKT` bit to be automatically set when the `RXRDY` bit is cleared. The `AUTOCL` and `AUTORQ` bits can be used with μ DMA accesses to perform complete bulk transfers without main processor intervention. When the `RXRDY` bit is cleared, the controller sends an acknowledge to the Device. When there is a known number of packets to be transferred, the **USB Request Packet Count in Block Transfer Endpoint n (USBRQPKTCOUNTn)** register associated with the endpoint should be configured to the number of packets to be transferred. The USB controller decrements the value in the **USBRQPKTCOUNTn** register following each request. When the **USBRQPKTCOUNTn** value decrements to 0, the `AUTORQ` bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, **USBRQPKTCOUNTn** should be cleared. `AUTORQ` then remains set until cleared by the reception of a short packet (that is, less than the `MAXLOAD` value in the **USBXMAXPn** register) such as may occur at the end of a bulk transfer.

If the Device responds to a bulk or interrupt IN token with a NAK, the USB Host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target Device responds with a STALL, however, the USB Host controller does not retry the transaction but sets the `STALLED` bit in the **USBCSRL0** register. If the target Device does not respond to the IN token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target Device has still not responded, the USB Host controller clears the `REQPKT` bit and sets the `ERROR` bit in the **USBCSRL0** register.

21.3.2.3 OUT Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in Device mode. The `TXRDY` bit in the **USBTXCSRLn** register must be set as each packet is loaded into the transmit FIFO. Again, setting the `AUTOSET` bit in the **USBTXCSRHn** register automatically sets `TXRDY` when a maximum-sized packet has been loaded into the FIFO. Furthermore, `AUTOSET` can be used with the μ DMA controller to perform complete bulk transfers without software intervention.

If the target Device responds to the OUT token with a NAK, the USB Host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target Device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the `STALLED` bit in the **USBTXCSRLn** register. If the target Device does not respond to the OUT token within the required time, or the packet contained a CRC or bit-stuff error, the USB Host controller retries the transaction. If after three attempts the target Device has still not responded, the USB controller flushes the FIFO and sets the `ERROR` bit in the **USBTXCSRLn** register.

21.3.2.4 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB Host controller. The Host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every

255 frames in 1 frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a Device is not responding. Isochronous endpoints can be scheduled from every frame to every 2^{16} frames, in powers of 2.

The USB controller maintains a frame counter. If the target Device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target Device is a low-speed device, a 'K' state is transmitted on the bus to act as a "keep-alive" to stop the low-speed device from going into SUSPEND mode.

After the SOF packet has been transmitted, the USB Host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for which the `REQPKT` bit is set or a transmit endpoint for which the `TXRDY` bit and/or the `FIFONE` bit is set.

An isochronous or interrupt transaction is started if the transaction is found on the first scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. As a result, only one interrupt or isochronous transaction occurs per endpoint every n frames, where n is the interval set via the **USB Host Transmit Interval Endpoint n (USBTXINTERVAL n)** or **USB Host Receive Interval Endpoint n (USBRXINTERVAL n)** register for that endpoint.

An active bulk transaction starts immediately, provided sufficient time is left in the frame to complete the transaction before the next SOF packet is due. If the transaction must be retried (for example, because a NAK was received or the target Device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This process ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The controller also allows the user to specify a limit to the length of time for NAKs to be received from a target Device before the endpoint times out.

21.3.2.5 USB Hubs

The following setup requirements apply to the USB Host controller only if it is used with a USB hub. When a full- or low-speed Device is connected to the USB controller via a USB 2.0 hub, details of the hub address and the hub port also must be recorded in the corresponding **USB Receive Hub Address Endpoint n (USBRXHUBADDR n)** and **USB Receive Hub Port Endpoint n (USBRXHUBPORT n)** or the **USB Transmit Hub Address Endpoint n (USBTXHUBADDR n)** and **USB Transmit Hub Port Endpoint n (USBTXHUBPORT n)** registers. In addition, the speed at which the Device operates (full or low) must be recorded in the **USB Type Endpoint 0 (USBTYPEn)** (endpoint 0), **USB Host Configure Transmit Type Endpoint n (USBTXTYPEn)**, or **USB Host Configure Receive Type Endpoint n (USBRXTYPEn)** registers for each endpoint that is accessed by the Device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB Devices. To maximize the number of Devices supported, the USB Host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to Device functions must be made following the completion of any on-going transactions on the endpoints affected.

21.3.2.6 Babble

The USB Host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. The controller also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB Host controller assumes that the target Device to which it is connected has malfunctioned, and the USB controller suspends all transactions and generates a babble interrupt.

21.3.2.7 Host SUSPEND

If the `SUSPEND` bit in the **USBPOWER** register is set, the USB Host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit SUSPEND mode, set the `RESUME` bit and clear the `SUSPEND` bit. While the `RESUME` bit is set, the USB Host controller generates RESUME signaling on the bus. After 20 ms, the `RESUME` bit must be cleared, at which point the frame counter and transaction scheduler start. The Host supports the detection of a remote wake-up.

21.3.2.8 USB RESET

If the `RESET` bit in the **USBPOWER** register is set, the USB Host controller generates USB RESET signaling on the bus. The `RESET` bit must be set for at least 20 ms to ensure correct resetting of the target Device. After the CPU has cleared the bit, the USB Host controller starts its frame counter and transaction scheduler.

21.3.2.9 Connect/Disconnect

A session is started by setting the `SESSION` bit in the **USB Device Control (USBDEVCTL)** register, enabling the USB controller to wait for a Device to be connected. When a Device is detected, a connect interrupt is generated. The speed of the Device that has been connected can be determined by reading the **USBDEVCTL** register where the `FSDEV` bit is set for a full-speed Device, and the `LSDEV` bit is set for a low-speed Device. The USB controller must generate a RESET to the Device, and then the USB Host controller can begin Device enumeration. If the Device is disconnected while a session is in progress, a disconnect interrupt is generated.

21.3.3 OTG Mode

To conserve power, the USB On-The-Go (OTG) supplement allows VBUS to only be powered up when required and to be turned off when the bus is not in use. VBUS is always supplied by the A device on the bus. The USB OTG controller determines whether it is the A device or the B device by sampling the ID input from the PHY. This signal is pulled Low when an A-type plug is sensed (signifying that the USB OTG controller should act as the A device) but taken High when a B-type plug is sensed (signifying that the USB controller is a B device). Note that when switching between OTG A and OTG B, the USB controller retains all register contents.

21.3.3.1 Starting a Session

When the USB OTG controller is ready to start a session, the `SESSION` bit must be set in the **USBDEVCTL** register. The USB OTG controller then enables ID pin sensing. The ID input is either taken Low if an A-type connection is detected or High if a B-type connection is detected. The `DEV` bit in the **USBDEVCTL** register is also set to indicate whether the USB OTG controller has adopted the role of the A device or the B device. The USB OTG controller also provides an interrupt to indicate that ID pin sensing has completed and the mode value in the **USBDEVCTL** register is valid. This interrupt is enabled in the **USBIDVIM** register, and the status is checked in the **USBIDVISC** register. As soon as the USB controller has detected that it is on the A side of the cable, it must enable VBUS power within 100ms or the USB controller reverts to device mode.

If the USB OTG controller is the A device, then the USB OTG controller enters Host mode (the A device is always the default Host), turns on VBUS, and waits for VBUS to go above the VBUS Valid threshold, as indicated by the `VBUS` bit in the **USBDEVCTL** register going to 0x3. The USB OTG controller then waits for a peripheral to be connected. When a peripheral is detected, a Connect interrupt is signaled and either the `FSDEV` or `LSDEV` bit in the **USBDEVCTL** register is set, depending whether a full-speed or a low-speed peripheral is detected. The USB controller then issues a RESET

to the connected Device. The `SESSION` bit in the **USBDEVCTL** register can be cleared to end a session. The USB OTG controller also automatically ends the session if babble is detected or if VBUS drops below session valid.

If the USB OTG controller is the B device, then the USB OTG controller requests a session using the session request protocol defined in the USB On-The-Go supplement, that is, it first discharges VBUS. Then when VBUS has gone below the Session End threshold (`VBUS` bit in the **USBDEVCTL** register goes to 0x0) and the line state has been a single-ended zero for > 2 ms, the USB OTG controller pulses the data line, then pulses VBUS. At the end of the session, the `SESSION` bit is cleared either by the USB OTG controller or by the application software. The USB OTG controller then causes the PHY to switch out the pull-up resistor on D+, signaling the A device to end the session.

21.3.3.2 Detecting Activity

When the other device of the OTG set-up wishes to start a session, it either raises VBUS above the Session Valid threshold if it is the A device, or if it is the B device, it pulses the data line then pulses VBUS. Depending on which of these actions happens, the USB controller can determine whether it is the A device or the B device in the current set-up and act accordingly. If VBUS is raised above the Session Valid threshold, then the USB controller is the B device. The USB controller sets the `SESSION` bit in the **USBDEVCTL** register. When RESET signaling is detected on the bus, a RESET interrupt is signaled, which is interpreted as the start of a session.

The USB controller is in Device mode as the B device is the default mode. At the end of the session, the A device turns off the power to VBUS. When VBUS drops below the Session Valid threshold, the USB controller detects this drop and clears the `SESSION` bit to indicate that the session has ended, causing a disconnect interrupt to be signaled. If data line and VBUS pulsing is detected, then the USB controller is the A device. The controller generates a SESSION REQUEST interrupt to indicate that the B device is requesting a session. The `SESSION` bit in the **USBDEVCTL** register must be set to start a session.

21.3.3.3 Host Negotiation

When the USB controller is the A device, ID is Low, and the controller automatically enters Host mode when a session starts. When the USB controller is the B device, ID is High, and the controller automatically enters Device mode when a session starts. However, software can request that the USB controller become the Host by setting the `HOSTREQ` bit in the **USBDEVCTL** register. This bit can be set either at the same time as requesting a Session Start by setting the `SESSION` bit in the **USBDEVCTL** register or at any time after a session has started. When the USB controller next enters SUSPEND mode and if the `HOSTREQ` bit remains set, the controller enters Host mode and begins host negotiation (as specified in the USB On-The-Go supplement) by causing the PHY to disconnect the pull-up resistor on the D+ line, causing the A device to switch to Device mode and connect its own pull-up resistor. When the USB controller detects this, a Connect interrupt is generated and the `RESET` bit in the **USBPOWER** register is set to begin resetting the A device. The USB controller begins this reset sequence automatically to ensure that RESET is started as required within 1 ms of the A device connecting its pull-up resistor. The main processor should wait at least 20 ms, then clear the `RESET` bit and enumerate the A device.

When the USB OTG controller B device has finished using the bus, the USB controller goes into SUSPEND mode by setting the `SUSPEND` bit in the **USBPOWER** register. The A device detects this and either terminates the session or reverts to Host mode. If the A device is USB OTG controller, it generates a Disconnect interrupt.

21.3.4 DMA Operation

The USB peripheral provides an interface connected to the μ DMA controller with separate channels for 3 transmit endpoints and 3 receive endpoints. Software selects which endpoints to service with the μ DMA channels using the **USB DMA Select (USBDMASEL)** register. The μ DMA operation of the USB is enabled through the **USBTXCSRHn** and **USBRXCSRHn** registers, for the TX and RX channels respectively. When μ DMA operation is enabled, the USB asserts a μ DMA request on the enabled receive or transmit channel when the associated FIFO can transfer data. When either FIFO can transfer data, the burst request for that channel is asserted. The μ DMA channel must be configured to operate in Basic mode, and the size of the μ DMA transfer must be restricted to whole multiples of the size of the USB FIFO. Both read and write transfers of the USB FIFOs using μ DMA must be configured in this manner. For example, if the USB endpoint is configured with a FIFO size of 64 bytes, the μ DMA channel can be used to transfer 64 bytes to or from the endpoint FIFO. If the number of bytes to transfer is less than 64, then a programmed I/O method must be used to copy the data to or from the FIFO.

If the **DMAMOD** bit in the **USBTXCSRHn/USBRXCSRHn** register is clear, an interrupt is generated after every packet is transferred, but the μ DMA continues transferring data. If the **DMAMOD** bit is set, an interrupt is generated only when the entire μ DMA transfer is complete. The interrupt occurs on the USB interrupt vector. Therefore, if interrupts are used for USB operation and the μ DMA is enabled, the USB interrupt handler must be designed to handle the μ DMA completion interrupt.

Care must be taken when using the μ DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of value of the **MAXLOAD** field in the **USBRXCSRHn** register. The **RXRDY** bit is cleared as follows.

Table 21-2. Remainder (RxMaxP/4)

| Value | Description |
|-------|--------------------|
| 0 | MAXLOAD = 64 bytes |
| 1 | MAXLOAD = 61 bytes |
| 2 | MAXLOAD = 62 bytes |
| 3 | MAXLOAD = 63 bytes |

Table 21-3. Actual Bytes Read

| Value | Description |
|-------|-------------|
| 0 | MAXLOAD |
| 1 | MAXLOAD+3 |
| 2 | MAXLOAD+2 |
| 3 | MAXLOAD+1 |

Table 21-4. Packet Sizes That Clear RXRDY

| Value | Description |
|-------|--|
| 0 | MAXLOAD, MAXLOAD-1, MAXLOAD-2, MAXLOAD-3 |
| 1 | MAXLOAD |
| 2 | MAXLOAD, MAXLOAD-1 |
| 3 | MAXLOAD, MAXLOAD-1, MAXLOAD-2 |

To enable DMA operation for the endpoint receive channel, the `DMAEN` bit of the **USBRXCSRHn** register should be set. To enable DMA operation for the endpoint transmit channel, the `DMAEN` bit of the **USBTXCSRHn** register must be set.

See “Micro Direct Memory Access (μ DMA)” on page 258 for more details about programming the μ DMA controller.

21.4 Initialization and Configuration

To use the USB Controller, the peripheral clock must be enabled by via the **RCGC2** register (see page 184). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 184). To find out which GPIO port to enable, refer to Table 24-4 on page 1028. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the USB signals to the appropriate pins (see page 358 and Table 24-5 on page 1035).

The initial configuration in all cases requires that the processor enable the USB controller and USB controller’s physical layer interface (PHY) before setting any registers. The next step is to enable the USB PLL so that the correct clocking is provided to the PHY. To ensure that voltage is not supplied to the bus incorrectly, the external power control signal, `USB0EPEN`, should be negated on start up by setting the `USB0EPEN` and `USB0PFLT` pins to be controlled by the USB controller and not have their default GPIO behavior.

The VBUS sense and ID pins (`USB0VBUS` and `USB0ID`) do not require any configuration as they are dedicated pins for the USB controller. In OTG mode, these pins directly connect to the USB connector’s VBUS and ID signals.

21.4.1 Pin Configuration

When using the Device controller portion of the USB controller in a system that also provides Host functionality, the power to VBUS must be disabled to allow the external Host controller to supply power. Usually, the `USB0EPEN` signal is used to control the external regulator and should be negated to avoid having two devices driving the `USB0VBUS` power pin on the USB connector.

When the USB controller is acting as a Host, it is in control of two signals that are attached to an external voltage supply that provides power to VBUS. The Host controller uses the `USB0EPEN` signal to enable or disable power to the `USB0VBUS` pin on the USB connector. An input pin, `USB0PFLT`, provides feedback when there has been a power fault on VBUS. The `USB0PFLT` signal can be configured to either automatically negate the `USB0EPEN` signal to disable power, and/or it can generate an interrupt to the interrupt controller to allow software to handle the power fault condition. The polarity and actions related to both `USB0EPEN` and `USB0PFLT` are fully configurable in the USB controller. The controller also provides interrupts on Device insertion and removal to allow the Host controller code to respond to these external events.

21.4.2 Endpoint Configuration

To start communication in Host or Device mode, the endpoint registers must first be configured. In Host mode, this configuration establishes a connection between an endpoint register and an endpoint on a Device. In Device mode, an endpoint must be configured before enumerating to the Host controller.

In both cases, the endpoint 0 configuration is limited because it is a fixed-function, fixed-FIFO-size endpoint. In Device and Host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In Device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the Host controller. In Host mode, the endpoints must be configured to operate as control, bulk, interrupt or

isochronous mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. Isochronous endpoints can have packets with up to 1023 bytes per packet. In either mode, the maximum packet size for the given endpoint must be set prior to sending or receiving data.

Configuring each endpoint's FIFO involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is 4 Kbytes with the first 64 bytes reserved for endpoint 0. The endpoint's FIFO must be at least as large as the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a Device, the USB Device controller's soft connect must be enabled when the Device is ready to start communications, indicating to the Host controller that the Device is ready to start the enumeration process. If operating as a Host controller, the Device soft connect must be disabled and power must be provided to VBUS via the `USB0EPEN` signal.

21.5 Register Map

Table 21-5 on page 872 lists the registers. All addresses given are relative to the USB base address of 0x4005.0000. Note that the USB controller clock must be enabled before the registers can be programmed (see page 184).

Table 21-5. Universal Serial Bus (USB) Controller Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------|------|-------------|-------------------------------|----------|
| 0x000 | USBFADDR | R/W | 0x00 | USB Device Functional Address | 884 |
| 0x001 | USBPOWER | R/W | 0x20 | USB Power | 885 |
| 0x002 | USBTXIS | RO | 0x0000 | USB Transmit Interrupt Status | 888 |
| 0x004 | USBRXIS | RO | 0x0000 | USB Receive Interrupt Status | 890 |
| 0x006 | USBTXIE | R/W | 0xFFFF | USB Transmit Interrupt Enable | 892 |
| 0x008 | USBRXIE | R/W | 0xFFFE | USB Receive Interrupt Enable | 894 |
| 0x00A | USBIS | RO | 0x00 | USB General Interrupt Status | 896 |
| 0x00B | USBIE | R/W | 0x06 | USB Interrupt Enable | 899 |
| 0x00C | USBFFRAME | RO | 0x0000 | USB Frame Value | 902 |
| 0x00E | USBEPIDX | R/W | 0x00 | USB Endpoint Index | 903 |
| 0x00F | USBTTEST | R/W | 0x00 | USB Test Mode | 904 |
| 0x020 | USBFIFO0 | R/W | 0x0000.0000 | USB FIFO Endpoint 0 | 906 |
| 0x024 | USBFIFO1 | R/W | 0x0000.0000 | USB FIFO Endpoint 1 | 906 |
| 0x028 | USBFIFO2 | R/W | 0x0000.0000 | USB FIFO Endpoint 2 | 906 |
| 0x02C | USBFIFO3 | R/W | 0x0000.0000 | USB FIFO Endpoint 3 | 906 |
| 0x030 | USBFIFO4 | R/W | 0x0000.0000 | USB FIFO Endpoint 4 | 906 |
| 0x034 | USBFIFO5 | R/W | 0x0000.0000 | USB FIFO Endpoint 5 | 906 |
| 0x038 | USBFIFO6 | R/W | 0x0000.0000 | USB FIFO Endpoint 6 | 906 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|----------------|------|-------------|--|----------|
| 0x03C | USBFIFO7 | R/W | 0x0000.0000 | USB FIFO Endpoint 7 | 906 |
| 0x040 | USBFIFO8 | R/W | 0x0000.0000 | USB FIFO Endpoint 8 | 906 |
| 0x044 | USBFIFO9 | R/W | 0x0000.0000 | USB FIFO Endpoint 9 | 906 |
| 0x048 | USBFIFO10 | R/W | 0x0000.0000 | USB FIFO Endpoint 10 | 906 |
| 0x04C | USBFIFO11 | R/W | 0x0000.0000 | USB FIFO Endpoint 11 | 906 |
| 0x050 | USBFIFO12 | R/W | 0x0000.0000 | USB FIFO Endpoint 12 | 906 |
| 0x054 | USBFIFO13 | R/W | 0x0000.0000 | USB FIFO Endpoint 13 | 906 |
| 0x058 | USBFIFO14 | R/W | 0x0000.0000 | USB FIFO Endpoint 14 | 906 |
| 0x05C | USBFIFO15 | R/W | 0x0000.0000 | USB FIFO Endpoint 15 | 906 |
| 0x060 | USBDEVCTL | R/W | 0x80 | USB Device Control | 908 |
| 0x062 | USBTXFIFOSZ | R/W | 0x00 | USB Transmit Dynamic FIFO Sizing | 910 |
| 0x063 | USBRXFIFOSZ | R/W | 0x00 | USB Receive Dynamic FIFO Sizing | 910 |
| 0x064 | USBTXFIFOADD | R/W | 0x0000 | USB Transmit FIFO Start Address | 911 |
| 0x066 | USBRXFIFOADD | R/W | 0x0000 | USB Receive FIFO Start Address | 911 |
| 0x07A | USBCONTIM | R/W | 0x5C | USB Connect Timing | 912 |
| 0x07B | USBVPLEN | R/W | 0x3C | USB OTG VBUS Pulse Timing | 913 |
| 0x07D | USBFSEOF | R/W | 0x77 | USB Full-Speed Last Transaction to End of Frame Timing | 914 |
| 0x07E | USBLSEOF | R/W | 0x72 | USB Low-Speed Last Transaction to End of Frame Timing | 915 |
| 0x080 | USBTXFUNCADDR0 | R/W | 0x00 | USB Transmit Functional Address Endpoint 0 | 916 |
| 0x082 | USBTXHUBADDR0 | R/W | 0x00 | USB Transmit Hub Address Endpoint 0 | 918 |
| 0x083 | USBTXHUBPORT0 | R/W | 0x00 | USB Transmit Hub Port Endpoint 0 | 920 |
| 0x088 | USBTXFUNCADDR1 | R/W | 0x00 | USB Transmit Functional Address Endpoint 1 | 916 |
| 0x08A | USBTXHUBADDR1 | R/W | 0x00 | USB Transmit Hub Address Endpoint 1 | 918 |
| 0x08B | USBTXHUBPORT1 | R/W | 0x00 | USB Transmit Hub Port Endpoint 1 | 920 |
| 0x08C | USBRXFUNCADDR1 | R/W | 0x00 | USB Receive Functional Address Endpoint 1 | 922 |
| 0x08E | USBRXHUBADDR1 | R/W | 0x00 | USB Receive Hub Address Endpoint 1 | 924 |
| 0x08F | USBRXHUBPORT1 | R/W | 0x00 | USB Receive Hub Port Endpoint 1 | 926 |
| 0x090 | USBTXFUNCADDR2 | R/W | 0x00 | USB Transmit Functional Address Endpoint 2 | 916 |
| 0x092 | USBTXHUBADDR2 | R/W | 0x00 | USB Transmit Hub Address Endpoint 2 | 918 |
| 0x093 | USBTXHUBPORT2 | R/W | 0x00 | USB Transmit Hub Port Endpoint 2 | 920 |
| 0x094 | USBRXFUNCADDR2 | R/W | 0x00 | USB Receive Functional Address Endpoint 2 | 922 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|----------------|------|-------|--|----------|
| 0x096 | USBRXHUBADDR2 | R/W | 0x00 | USB Receive Hub Address Endpoint 2 | 924 |
| 0x097 | USBRXHUBPORT2 | R/W | 0x00 | USB Receive Hub Port Endpoint 2 | 926 |
| 0x098 | USBTXFUNCADDR3 | R/W | 0x00 | USB Transmit Functional Address Endpoint 3 | 916 |
| 0x09A | USBTXHUBADDR3 | R/W | 0x00 | USB Transmit Hub Address Endpoint 3 | 918 |
| 0x09B | USBTXHUBPORT3 | R/W | 0x00 | USB Transmit Hub Port Endpoint 3 | 920 |
| 0x09C | USBRXFUNCADDR3 | R/W | 0x00 | USB Receive Functional Address Endpoint 3 | 922 |
| 0x09E | USBRXHUBADDR3 | R/W | 0x00 | USB Receive Hub Address Endpoint 3 | 924 |
| 0x09F | USBRXHUBPORT3 | R/W | 0x00 | USB Receive Hub Port Endpoint 3 | 926 |
| 0x0A0 | USBTXFUNCADDR4 | R/W | 0x00 | USB Transmit Functional Address Endpoint 4 | 916 |
| 0x0A2 | USBTXHUBADDR4 | R/W | 0x00 | USB Transmit Hub Address Endpoint 4 | 918 |
| 0x0A3 | USBTXHUBPORT4 | R/W | 0x00 | USB Transmit Hub Port Endpoint 4 | 920 |
| 0x0A4 | USBRXFUNCADDR4 | R/W | 0x00 | USB Receive Functional Address Endpoint 4 | 922 |
| 0x0A6 | USBRXHUBADDR4 | R/W | 0x00 | USB Receive Hub Address Endpoint 4 | 924 |
| 0x0A7 | USBRXHUBPORT4 | R/W | 0x00 | USB Receive Hub Port Endpoint 4 | 926 |
| 0x0A8 | USBTXFUNCADDR5 | R/W | 0x00 | USB Transmit Functional Address Endpoint 5 | 916 |
| 0x0AA | USBTXHUBADDR5 | R/W | 0x00 | USB Transmit Hub Address Endpoint 5 | 918 |
| 0x0AB | USBTXHUBPORT5 | R/W | 0x00 | USB Transmit Hub Port Endpoint 5 | 920 |
| 0x0AC | USBRXFUNCADDR5 | R/W | 0x00 | USB Receive Functional Address Endpoint 5 | 922 |
| 0x0AE | USBRXHUBADDR5 | R/W | 0x00 | USB Receive Hub Address Endpoint 5 | 924 |
| 0x0AF | USBRXHUBPORT5 | R/W | 0x00 | USB Receive Hub Port Endpoint 5 | 926 |
| 0x0B0 | USBTXFUNCADDR6 | R/W | 0x00 | USB Transmit Functional Address Endpoint 6 | 916 |
| 0x0B2 | USBTXHUBADDR6 | R/W | 0x00 | USB Transmit Hub Address Endpoint 6 | 918 |
| 0x0B3 | USBTXHUBPORT6 | R/W | 0x00 | USB Transmit Hub Port Endpoint 6 | 920 |
| 0x0B4 | USBRXFUNCADDR6 | R/W | 0x00 | USB Receive Functional Address Endpoint 6 | 922 |
| 0x0B6 | USBRXHUBADDR6 | R/W | 0x00 | USB Receive Hub Address Endpoint 6 | 924 |
| 0x0B7 | USBRXHUBPORT6 | R/W | 0x00 | USB Receive Hub Port Endpoint 6 | 926 |
| 0x0B8 | USBTXFUNCADDR7 | R/W | 0x00 | USB Transmit Functional Address Endpoint 7 | 916 |
| 0x0BA | USBTXHUBADDR7 | R/W | 0x00 | USB Transmit Hub Address Endpoint 7 | 918 |
| 0x0BB | USBTXHUBPORT7 | R/W | 0x00 | USB Transmit Hub Port Endpoint 7 | 920 |
| 0x0BC | USBRXFUNCADDR7 | R/W | 0x00 | USB Receive Functional Address Endpoint 7 | 922 |
| 0x0BE | USBRXHUBADDR7 | R/W | 0x00 | USB Receive Hub Address Endpoint 7 | 924 |
| 0x0BF | USBRXHUBPORT7 | R/W | 0x00 | USB Receive Hub Port Endpoint 7 | 926 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|-------|---|----------|
| 0x0C0 | USBTXFUNCADDR8 | R/W | 0x00 | USB Transmit Functional Address Endpoint 8 | 916 |
| 0x0C2 | USBTXHUBADDR8 | R/W | 0x00 | USB Transmit Hub Address Endpoint 8 | 918 |
| 0x0C3 | USBTXHUBPORT8 | R/W | 0x00 | USB Transmit Hub Port Endpoint 8 | 920 |
| 0x0C4 | USBRXFUNCADDR8 | R/W | 0x00 | USB Receive Functional Address Endpoint 8 | 922 |
| 0x0C6 | USBRXHUBADDR8 | R/W | 0x00 | USB Receive Hub Address Endpoint 8 | 924 |
| 0x0C7 | USBRXHUBPORT8 | R/W | 0x00 | USB Receive Hub Port Endpoint 8 | 926 |
| 0x0C8 | USBTXFUNCADDR9 | R/W | 0x00 | USB Transmit Functional Address Endpoint 9 | 916 |
| 0x0CA | USBTXHUBADDR9 | R/W | 0x00 | USB Transmit Hub Address Endpoint 9 | 918 |
| 0x0CB | USBTXHUBPORT9 | R/W | 0x00 | USB Transmit Hub Port Endpoint 9 | 920 |
| 0x0CC | USBRXFUNCADDR9 | R/W | 0x00 | USB Receive Functional Address Endpoint 9 | 922 |
| 0x0CE | USBRXHUBADDR9 | R/W | 0x00 | USB Receive Hub Address Endpoint 9 | 924 |
| 0x0CF | USBRXHUBPORT9 | R/W | 0x00 | USB Receive Hub Port Endpoint 9 | 926 |
| 0x0D0 | USBTXFUNCADDR10 | R/W | 0x00 | USB Transmit Functional Address Endpoint 10 | 916 |
| 0x0D2 | USBTXHUBADDR10 | R/W | 0x00 | USB Transmit Hub Address Endpoint 10 | 918 |
| 0x0D3 | USBTXHUBPORT10 | R/W | 0x00 | USB Transmit Hub Port Endpoint 10 | 920 |
| 0x0D4 | USBRXFUNCADDR10 | R/W | 0x00 | USB Receive Functional Address Endpoint 10 | 922 |
| 0x0D6 | USBRXHUBADDR10 | R/W | 0x00 | USB Receive Hub Address Endpoint 10 | 924 |
| 0x0D7 | USBRXHUBPORT10 | R/W | 0x00 | USB Receive Hub Port Endpoint 10 | 926 |
| 0x0D8 | USBTXFUNCADDR11 | R/W | 0x00 | USB Transmit Functional Address Endpoint 11 | 916 |
| 0x0DA | USBTXHUBADDR11 | R/W | 0x00 | USB Transmit Hub Address Endpoint 11 | 918 |
| 0x0DB | USBTXHUBPORT11 | R/W | 0x00 | USB Transmit Hub Port Endpoint 11 | 920 |
| 0x0DC | USBRXFUNCADDR11 | R/W | 0x00 | USB Receive Functional Address Endpoint 11 | 922 |
| 0x0DE | USBRXHUBADDR11 | R/W | 0x00 | USB Receive Hub Address Endpoint 11 | 924 |
| 0x0DF | USBRXHUBPORT11 | R/W | 0x00 | USB Receive Hub Port Endpoint 11 | 926 |
| 0x0E0 | USBTXFUNCADDR12 | R/W | 0x00 | USB Transmit Functional Address Endpoint 12 | 916 |
| 0x0E2 | USBTXHUBADDR12 | R/W | 0x00 | USB Transmit Hub Address Endpoint 12 | 918 |
| 0x0E3 | USBTXHUBPORT12 | R/W | 0x00 | USB Transmit Hub Port Endpoint 12 | 920 |
| 0x0E4 | USBRXFUNCADDR12 | R/W | 0x00 | USB Receive Functional Address Endpoint 12 | 922 |
| 0x0E6 | USBRXHUBADDR12 | R/W | 0x00 | USB Receive Hub Address Endpoint 12 | 924 |
| 0x0E7 | USBRXHUBPORT12 | R/W | 0x00 | USB Receive Hub Port Endpoint 12 | 926 |
| 0x0E8 | USBTXFUNCADDR13 | R/W | 0x00 | USB Transmit Functional Address Endpoint 13 | 916 |
| 0x0EA | USBTXHUBADDR13 | R/W | 0x00 | USB Transmit Hub Address Endpoint 13 | 918 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|--------|---|----------|
| 0x0EB | USBTXHUBPORT13 | R/W | 0x00 | USB Transmit Hub Port Endpoint 13 | 920 |
| 0x0EC | USBRXFUNCADDR13 | R/W | 0x00 | USB Receive Functional Address Endpoint 13 | 922 |
| 0x0EE | USBRXHUBADDR13 | R/W | 0x00 | USB Receive Hub Address Endpoint 13 | 924 |
| 0x0EF | USBRXHUBPORT13 | R/W | 0x00 | USB Receive Hub Port Endpoint 13 | 926 |
| 0x0F0 | USBTXFUNCADDR14 | R/W | 0x00 | USB Transmit Functional Address Endpoint 14 | 916 |
| 0x0F2 | USBTXHUBADDR14 | R/W | 0x00 | USB Transmit Hub Address Endpoint 14 | 918 |
| 0x0F3 | USBTXHUBPORT14 | R/W | 0x00 | USB Transmit Hub Port Endpoint 14 | 920 |
| 0x0F4 | USBRXFUNCADDR14 | R/W | 0x00 | USB Receive Functional Address Endpoint 14 | 922 |
| 0x0F6 | USBRXHUBADDR14 | R/W | 0x00 | USB Receive Hub Address Endpoint 14 | 924 |
| 0x0F7 | USBRXHUBPORT14 | R/W | 0x00 | USB Receive Hub Port Endpoint 14 | 926 |
| 0x0F8 | USBTXFUNCADDR15 | R/W | 0x00 | USB Transmit Functional Address Endpoint 15 | 916 |
| 0x0FA | USBTXHUBADDR15 | R/W | 0x00 | USB Transmit Hub Address Endpoint 15 | 918 |
| 0x0FB | USBTXHUBPORT15 | R/W | 0x00 | USB Transmit Hub Port Endpoint 15 | 920 |
| 0x0FC | USBRXFUNCADDR15 | R/W | 0x00 | USB Receive Functional Address Endpoint 15 | 922 |
| 0x0FE | USBRXHUBADDR15 | R/W | 0x00 | USB Receive Hub Address Endpoint 15 | 924 |
| 0x0FF | USBRXHUBPORT15 | R/W | 0x00 | USB Receive Hub Port Endpoint 15 | 926 |
| 0x102 | USBCSRL0 | W1C | 0x00 | USB Control and Status Endpoint 0 Low | 930 |
| 0x103 | USBCSRH0 | W1C | 0x00 | USB Control and Status Endpoint 0 High | 934 |
| 0x108 | USBCOUNT0 | RO | 0x00 | USB Receive Byte Count Endpoint 0 | 936 |
| 0x10A | USBTTYPE0 | R/W | 0x00 | USB Type Endpoint 0 | 937 |
| 0x10B | USBNAKLMT | R/W | 0x00 | USB NAK Limit | 938 |
| 0x110 | USBTXMAXP1 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 1 | 928 |
| 0x112 | USBTXCSSL1 | R/W | 0x00 | USB Transmit Control and Status Endpoint 1 Low | 939 |
| 0x113 | USBTXCSSL1 | R/W | 0x00 | USB Transmit Control and Status Endpoint 1 High | 944 |
| 0x114 | USBRXMAXP1 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 1 | 948 |
| 0x116 | USBRXCSSL1 | R/W | 0x00 | USB Receive Control and Status Endpoint 1 Low | 950 |
| 0x117 | USBRXCSSL1 | R/W | 0x00 | USB Receive Control and Status Endpoint 1 High | 955 |
| 0x118 | USBRXCOUNT1 | RO | 0x0000 | USB Receive Byte Count Endpoint 1 | 960 |
| 0x11A | USBTXTYPE1 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 1 | 962 |
| 0x11B | USBTXINTERVAL1 | R/W | 0x00 | USB Host Transmit Interval Endpoint 1 | 964 |
| 0x11C | USBRXTYPE1 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 1 | 966 |
| 0x11D | USBRXINTERVAL1 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 1 | 968 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|----------------|------|--------|---|----------|
| 0x120 | USBTXMAXP2 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 2 | 928 |
| 0x122 | USBTXCURL2 | R/W | 0x00 | USB Transmit Control and Status Endpoint 2 Low | 939 |
| 0x123 | USBTXCSRH2 | R/W | 0x00 | USB Transmit Control and Status Endpoint 2 High | 944 |
| 0x124 | USBRXMAXP2 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 2 | 948 |
| 0x126 | USBRXCURL2 | R/W | 0x00 | USB Receive Control and Status Endpoint 2 Low | 950 |
| 0x127 | USBRXCSRH2 | R/W | 0x00 | USB Receive Control and Status Endpoint 2 High | 955 |
| 0x128 | USBRXCOUNT2 | RO | 0x0000 | USB Receive Byte Count Endpoint 2 | 960 |
| 0x12A | USBTXTYPE2 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 2 | 962 |
| 0x12B | USBTXINTERVAL2 | R/W | 0x00 | USB Host Transmit Interval Endpoint 2 | 964 |
| 0x12C | USBRXTYPE2 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 2 | 966 |
| 0x12D | USBRXINTERVAL2 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 2 | 968 |
| 0x130 | USBTXMAXP3 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 3 | 928 |
| 0x132 | USBTXCURL3 | R/W | 0x00 | USB Transmit Control and Status Endpoint 3 Low | 939 |
| 0x133 | USBTXCSRH3 | R/W | 0x00 | USB Transmit Control and Status Endpoint 3 High | 944 |
| 0x134 | USBRXMAXP3 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 3 | 948 |
| 0x136 | USBRXCURL3 | R/W | 0x00 | USB Receive Control and Status Endpoint 3 Low | 950 |
| 0x137 | USBRXCSRH3 | R/W | 0x00 | USB Receive Control and Status Endpoint 3 High | 955 |
| 0x138 | USBRXCOUNT3 | RO | 0x0000 | USB Receive Byte Count Endpoint 3 | 960 |
| 0x13A | USBTXTYPE3 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 3 | 962 |
| 0x13B | USBTXINTERVAL3 | R/W | 0x00 | USB Host Transmit Interval Endpoint 3 | 964 |
| 0x13C | USBRXTYPE3 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 3 | 966 |
| 0x13D | USBRXINTERVAL3 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 3 | 968 |
| 0x140 | USBTXMAXP4 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 4 | 928 |
| 0x142 | USBTXCURL4 | R/W | 0x00 | USB Transmit Control and Status Endpoint 4 Low | 939 |
| 0x143 | USBTXCSRH4 | R/W | 0x00 | USB Transmit Control and Status Endpoint 4 High | 944 |
| 0x144 | USBRXMAXP4 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 4 | 948 |
| 0x146 | USBRXCURL4 | R/W | 0x00 | USB Receive Control and Status Endpoint 4 Low | 950 |
| 0x147 | USBRXCSRH4 | R/W | 0x00 | USB Receive Control and Status Endpoint 4 High | 955 |
| 0x148 | USBRXCOUNT4 | RO | 0x0000 | USB Receive Byte Count Endpoint 4 | 960 |
| 0x14A | USBTXTYPE4 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 4 | 962 |
| 0x14B | USBTXINTERVAL4 | R/W | 0x00 | USB Host Transmit Interval Endpoint 4 | 964 |
| 0x14C | USBRXTYPE4 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 4 | 966 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|----------------|------|--------|---|----------|
| 0x14D | USBRXINTERVAL4 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 4 | 968 |
| 0x150 | USBTXMAXP5 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 5 | 928 |
| 0x152 | USBTXCSSL5 | R/W | 0x00 | USB Transmit Control and Status Endpoint 5 Low | 939 |
| 0x153 | USBTXCSSL5H | R/W | 0x00 | USB Transmit Control and Status Endpoint 5 High | 944 |
| 0x154 | USBRXMAXP5 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 5 | 948 |
| 0x156 | USBRXCSSL5 | R/W | 0x00 | USB Receive Control and Status Endpoint 5 Low | 950 |
| 0x157 | USBRXCSSL5H | R/W | 0x00 | USB Receive Control and Status Endpoint 5 High | 955 |
| 0x158 | USBRXCOUNT5 | RO | 0x0000 | USB Receive Byte Count Endpoint 5 | 960 |
| 0x15A | USBTXTYPE5 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 5 | 962 |
| 0x15B | USBTXINTERVAL5 | R/W | 0x00 | USB Host Transmit Interval Endpoint 5 | 964 |
| 0x15C | USBRXTYPE5 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 5 | 966 |
| 0x15D | USBRXINTERVAL5 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 5 | 968 |
| 0x160 | USBTXMAXP6 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 6 | 928 |
| 0x162 | USBTXCSSL6 | R/W | 0x00 | USB Transmit Control and Status Endpoint 6 Low | 939 |
| 0x163 | USBTXCSSL6H | R/W | 0x00 | USB Transmit Control and Status Endpoint 6 High | 944 |
| 0x164 | USBRXMAXP6 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 6 | 948 |
| 0x166 | USBRXCSSL6 | R/W | 0x00 | USB Receive Control and Status Endpoint 6 Low | 950 |
| 0x167 | USBRXCSSL6H | R/W | 0x00 | USB Receive Control and Status Endpoint 6 High | 955 |
| 0x168 | USBRXCOUNT6 | RO | 0x0000 | USB Receive Byte Count Endpoint 6 | 960 |
| 0x16A | USBTXTYPE6 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 6 | 962 |
| 0x16B | USBTXINTERVAL6 | R/W | 0x00 | USB Host Transmit Interval Endpoint 6 | 964 |
| 0x16C | USBRXTYPE6 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 6 | 966 |
| 0x16D | USBRXINTERVAL6 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 6 | 968 |
| 0x170 | USBTXMAXP7 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 7 | 928 |
| 0x172 | USBTXCSSL7 | R/W | 0x00 | USB Transmit Control and Status Endpoint 7 Low | 939 |
| 0x173 | USBTXCSSL7H | R/W | 0x00 | USB Transmit Control and Status Endpoint 7 High | 944 |
| 0x174 | USBRXMAXP7 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 7 | 948 |
| 0x176 | USBRXCSSL7 | R/W | 0x00 | USB Receive Control and Status Endpoint 7 Low | 950 |
| 0x177 | USBRXCSSL7H | R/W | 0x00 | USB Receive Control and Status Endpoint 7 High | 955 |
| 0x178 | USBRXCOUNT7 | RO | 0x0000 | USB Receive Byte Count Endpoint 7 | 960 |
| 0x17A | USBTXTYPE7 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 7 | 962 |
| 0x17B | USBTXINTERVAL7 | R/W | 0x00 | USB Host Transmit Interval Endpoint 7 | 964 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|----------------|------|--------|--|----------|
| 0x17C | USBRXTYPE7 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 7 | 966 |
| 0x17D | USBRXINTERVAL7 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 7 | 968 |
| 0x180 | USBTXMAXP8 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 8 | 928 |
| 0x182 | USBTXCSSL8 | R/W | 0x00 | USB Transmit Control and Status Endpoint 8 Low | 939 |
| 0x183 | USBTXCSSL8H | R/W | 0x00 | USB Transmit Control and Status Endpoint 8 High | 944 |
| 0x184 | USBRXMAXP8 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 8 | 948 |
| 0x186 | USBRXCSSL8 | R/W | 0x00 | USB Receive Control and Status Endpoint 8 Low | 950 |
| 0x187 | USBRXCSSL8H | R/W | 0x00 | USB Receive Control and Status Endpoint 8 High | 955 |
| 0x188 | USBRXCOUNT8 | RO | 0x0000 | USB Receive Byte Count Endpoint 8 | 960 |
| 0x18A | USBTXTYPE8 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 8 | 962 |
| 0x18B | USBTXINTERVAL8 | R/W | 0x00 | USB Host Transmit Interval Endpoint 8 | 964 |
| 0x18C | USBRXTYPE8 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 8 | 966 |
| 0x18D | USBRXINTERVAL8 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 8 | 968 |
| 0x190 | USBTXMAXP9 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 9 | 928 |
| 0x192 | USBTXCSSL9 | R/W | 0x00 | USB Transmit Control and Status Endpoint 9 Low | 939 |
| 0x193 | USBTXCSSL9H | R/W | 0x00 | USB Transmit Control and Status Endpoint 9 High | 944 |
| 0x194 | USBRXMAXP9 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 9 | 948 |
| 0x196 | USBRXCSSL9 | R/W | 0x00 | USB Receive Control and Status Endpoint 9 Low | 950 |
| 0x197 | USBRXCSSL9H | R/W | 0x00 | USB Receive Control and Status Endpoint 9 High | 955 |
| 0x198 | USBRXCOUNT9 | RO | 0x0000 | USB Receive Byte Count Endpoint 9 | 960 |
| 0x19A | USBTXTYPE9 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 9 | 962 |
| 0x19B | USBTXINTERVAL9 | R/W | 0x00 | USB Host Transmit Interval Endpoint 9 | 964 |
| 0x19C | USBRXTYPE9 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 9 | 966 |
| 0x19D | USBRXINTERVAL9 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 9 | 968 |
| 0x1A0 | USBTXMAXP10 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 10 | 928 |
| 0x1A2 | USBTXCSSL10 | R/W | 0x00 | USB Transmit Control and Status Endpoint 10 Low | 939 |
| 0x1A3 | USBTXCSSL10H | R/W | 0x00 | USB Transmit Control and Status Endpoint 10 High | 944 |
| 0x1A4 | USBRXMAXP10 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 10 | 948 |
| 0x1A6 | USBRXCSSL10 | R/W | 0x00 | USB Receive Control and Status Endpoint 10 Low | 950 |
| 0x1A7 | USBRXCSSL10H | R/W | 0x00 | USB Receive Control and Status Endpoint 10 High | 955 |
| 0x1A8 | USBRXCOUNT10 | RO | 0x0000 | USB Receive Byte Count Endpoint 10 | 960 |
| 0x1AA | USBTXTYPE10 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 10 | 962 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|--------|--|----------|
| 0x1AB | USBTXINTERVAL10 | R/W | 0x00 | USB Host Transmit Interval Endpoint 10 | 964 |
| 0x1AC | USBRXTYPE10 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 10 | 966 |
| 0x1AD | USBRXINTERVAL10 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 10 | 968 |
| 0x1B0 | USBTXMAXP11 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 11 | 928 |
| 0x1B2 | USBTXCSRL11 | R/W | 0x00 | USB Transmit Control and Status Endpoint 11 Low | 939 |
| 0x1B3 | USBTXCSRH11 | R/W | 0x00 | USB Transmit Control and Status Endpoint 11 High | 944 |
| 0x1B4 | USBRXMAXP11 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 11 | 948 |
| 0x1B6 | USBRXCSRL11 | R/W | 0x00 | USB Receive Control and Status Endpoint 11 Low | 950 |
| 0x1B7 | USBRXCSRH11 | R/W | 0x00 | USB Receive Control and Status Endpoint 11 High | 955 |
| 0x1B8 | USBRXCOUNT11 | RO | 0x0000 | USB Receive Byte Count Endpoint 11 | 960 |
| 0x1BA | USBTXTYPE11 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 11 | 962 |
| 0x1BB | USBTXINTERVAL11 | R/W | 0x00 | USB Host Transmit Interval Endpoint 11 | 964 |
| 0x1BC | USBRXTYPE11 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 11 | 966 |
| 0x1BD | USBRXINTERVAL11 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 11 | 968 |
| 0x1C0 | USBTXMAXP12 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 12 | 928 |
| 0x1C2 | USBTXCSRL12 | R/W | 0x00 | USB Transmit Control and Status Endpoint 12 Low | 939 |
| 0x1C3 | USBTXCSRH12 | R/W | 0x00 | USB Transmit Control and Status Endpoint 12 High | 944 |
| 0x1C4 | USBRXMAXP12 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 12 | 948 |
| 0x1C6 | USBRXCSRL12 | R/W | 0x00 | USB Receive Control and Status Endpoint 12 Low | 950 |
| 0x1C7 | USBRXCSRH12 | R/W | 0x00 | USB Receive Control and Status Endpoint 12 High | 955 |
| 0x1C8 | USBRXCOUNT12 | RO | 0x0000 | USB Receive Byte Count Endpoint 12 | 960 |
| 0x1CA | USBTXTYPE12 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 12 | 962 |
| 0x1CB | USBTXINTERVAL12 | R/W | 0x00 | USB Host Transmit Interval Endpoint 12 | 964 |
| 0x1CC | USBRXTYPE12 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 12 | 966 |
| 0x1CD | USBRXINTERVAL12 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 12 | 968 |
| 0x1D0 | USBTXMAXP13 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 13 | 928 |
| 0x1D2 | USBTXCSRL13 | R/W | 0x00 | USB Transmit Control and Status Endpoint 13 Low | 939 |
| 0x1D3 | USBTXCSRH13 | R/W | 0x00 | USB Transmit Control and Status Endpoint 13 High | 944 |
| 0x1D4 | USBRXMAXP13 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 13 | 948 |
| 0x1D6 | USBRXCSRL13 | R/W | 0x00 | USB Receive Control and Status Endpoint 13 Low | 950 |
| 0x1D7 | USBRXCSRH13 | R/W | 0x00 | USB Receive Control and Status Endpoint 13 High | 955 |
| 0x1D8 | USBRXCOUNT13 | RO | 0x0000 | USB Receive Byte Count Endpoint 13 | 960 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|--------|---|----------|
| 0x1DA | USBTXTYPE13 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 13 | 962 |
| 0x1DB | USBTXINTERVAL13 | R/W | 0x00 | USB Host Transmit Interval Endpoint 13 | 964 |
| 0x1DC | USBRXTYPE13 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 13 | 966 |
| 0x1DD | USBRXINTERVAL13 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 13 | 968 |
| 0x1E0 | USBTXMAXP14 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 14 | 928 |
| 0x1E2 | USBTXCSRL14 | R/W | 0x00 | USB Transmit Control and Status Endpoint 14 Low | 939 |
| 0x1E3 | USBTXCSRH14 | R/W | 0x00 | USB Transmit Control and Status Endpoint 14 High | 944 |
| 0x1E4 | USBRXMAXP14 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 14 | 948 |
| 0x1E6 | USBRXCSRL14 | R/W | 0x00 | USB Receive Control and Status Endpoint 14 Low | 950 |
| 0x1E7 | USBRXCSRH14 | R/W | 0x00 | USB Receive Control and Status Endpoint 14 High | 955 |
| 0x1E8 | USBRXCOUNT14 | RO | 0x0000 | USB Receive Byte Count Endpoint 14 | 960 |
| 0x1EA | USBTXTYPE14 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 14 | 962 |
| 0x1EB | USBTXINTERVAL14 | R/W | 0x00 | USB Host Transmit Interval Endpoint 14 | 964 |
| 0x1EC | USBRXTYPE14 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 14 | 966 |
| 0x1ED | USBRXINTERVAL14 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 14 | 968 |
| 0x1F0 | USBTXMAXP15 | R/W | 0x0000 | USB Maximum Transmit Data Endpoint 15 | 928 |
| 0x1F2 | USBTXCSRL15 | R/W | 0x00 | USB Transmit Control and Status Endpoint 15 Low | 939 |
| 0x1F3 | USBTXCSRH15 | R/W | 0x00 | USB Transmit Control and Status Endpoint 15 High | 944 |
| 0x1F4 | USBRXMAXP15 | R/W | 0x0000 | USB Maximum Receive Data Endpoint 15 | 948 |
| 0x1F6 | USBRXCSRL15 | R/W | 0x00 | USB Receive Control and Status Endpoint 15 Low | 950 |
| 0x1F7 | USBRXCSRH15 | R/W | 0x00 | USB Receive Control and Status Endpoint 15 High | 955 |
| 0x1F8 | USBRXCOUNT15 | RO | 0x0000 | USB Receive Byte Count Endpoint 15 | 960 |
| 0x1FA | USBTXTYPE15 | R/W | 0x00 | USB Host Transmit Configure Type Endpoint 15 | 962 |
| 0x1FB | USBTXINTERVAL15 | R/W | 0x00 | USB Host Transmit Interval Endpoint 15 | 964 |
| 0x1FC | USBRXTYPE15 | R/W | 0x00 | USB Host Configure Receive Type Endpoint 15 | 966 |
| 0x1FD | USBRXINTERVAL15 | R/W | 0x00 | USB Host Receive Polling Interval Endpoint 15 | 968 |
| 0x304 | USBRQPKTCOUNT1 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 1 | 970 |
| 0x308 | USBRQPKTCOUNT2 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 2 | 970 |
| 0x30C | USBRQPKTCOUNT3 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 3 | 970 |
| 0x310 | USBRQPKTCOUNT4 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 4 | 970 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------------|------|-------------|--|----------|
| 0x314 | USBRQPKTCOUNT5 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 5 | 970 |
| 0x318 | USBRQPKTCOUNT6 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 6 | 970 |
| 0x31C | USBRQPKTCOUNT7 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 7 | 970 |
| 0x320 | USBRQPKTCOUNT8 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 8 | 970 |
| 0x324 | USBRQPKTCOUNT9 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 9 | 970 |
| 0x328 | USBRQPKTCOUNT10 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 10 | 970 |
| 0x32C | USBRQPKTCOUNT11 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 11 | 970 |
| 0x330 | USBRQPKTCOUNT12 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 12 | 970 |
| 0x334 | USBRQPKTCOUNT13 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 13 | 970 |
| 0x338 | USBRQPKTCOUNT14 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 14 | 970 |
| 0x33C | USBRQPKTCOUNT15 | R/W | 0x0000 | USB Request Packet Count in Block Transfer Endpoint 15 | 970 |
| 0x340 | USBRXDPKTBUFDIS | R/W | 0x0000 | USB Receive Double Packet Buffer Disable | 972 |
| 0x342 | USBTXDPKTBUFDIS | R/W | 0x0000 | USB Transmit Double Packet Buffer Disable | 974 |
| 0x400 | USBEPCC | R/W | 0x0000.0000 | USB External Power Control | 976 |
| 0x404 | USBEPCCRIS | RO | 0x0000.0000 | USB External Power Control Raw Interrupt Status | 979 |
| 0x408 | USBEPCCIM | R/W | 0x0000.0000 | USB External Power Control Interrupt Mask | 980 |
| 0x40C | USBEPCCISC | R/W | 0x0000.0000 | USB External Power Control Interrupt Status and Clear | 981 |
| 0x410 | USBDRRIS | RO | 0x0000.0000 | USB Device RESUME Raw Interrupt Status | 982 |
| 0x414 | USBDRRIM | R/W | 0x0000.0000 | USB Device RESUME Interrupt Mask | 983 |
| 0x418 | USBDRRISC | W1C | 0x0000.0000 | USB Device RESUME Interrupt Status and Clear | 984 |
| 0x41C | USBGPCS | R/W | 0x0000.0000 | USB General-Purpose Control and Status | 985 |
| 0x430 | USBVDC | R/W | 0x0000.0000 | USB VBUS Droop Control | 986 |
| 0x434 | USBVDCRIS | RO | 0x0000.0000 | USB VBUS Droop Control Raw Interrupt Status | 987 |
| 0x438 | USBVDCIM | R/W | 0x0000.0000 | USB VBUS Droop Control Interrupt Mask | 988 |
| 0x43C | USBVDCISC | R/W | 0x0000.0000 | USB VBUS Droop Control Interrupt Status and Clear | 989 |
| 0x444 | USBIDVRIS | RO | 0x0000.0000 | USB ID Valid Detect Raw Interrupt Status | 990 |

Table 21-5. Universal Serial Bus (USB) Controller Register Map (continued)

| Offset | Name | Type | Reset | Description | See page |
|--------|-----------|-------|-------------|--|----------|
| 0x448 | USBIDVIM | R/W | 0x0000.0000 | USB ID Valid Detect Interrupt Mask | 991 |
| 0x44C | USBIDVISC | R/W1C | 0x0000.0000 | USB ID Valid Detect Interrupt Status and Clear | 992 |
| 0x450 | USBDMASEL | R/W | 0x0033.2211 | USB DMA Select | 993 |

21.6 Register Descriptions

The LM3S9B90 USB controller is configured to the On-The-Go (OTG) communication mode as specified in the `USB0` bit field in the **DC6** register (see page 155).

**OTG B /
Device**

This icon indicates that the register is used in OTG B or Device mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode.

**OTG A /
Host**

This icon indicates that the register is used in OTG A or Host mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode. The USB controller is in OTG B or Device mode upon reset, so the reset values shown for these registers apply to the Device mode definition.

OTG

This icon indicates that the register is used for OTG-specific functions such as ID detection and negotiation. Once OTG negotiation is complete, then the USB controller registers are used according to their Host or Device mode meanings depending on whether the OTG negotiations made the USB controller OTG A (Host) or OTG B (Device).

Register 1: USB Device Functional Address (USBFADDR), offset 0x000

OTG B /
Device

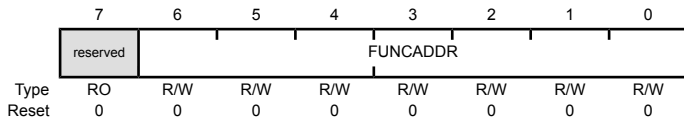
USBFADDR is an 8-bit register that contains the 7-bit address of the Device part of the transaction.

When the USB controller is being used in Device mode (the `HOST` bit in the **USBDEVCTL** register is clear), this register must be written with the address received through a `SET_ADDRESS` command, which is then used for decoding the function address in subsequent token packets.

Important: See the section called “Setting the Device Address” on page 863 for special considerations when writing this register.

USB Device Functional Address (USBFADDR)

Base 0x4005.0000
Offset 0x000
Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | FUNCADDR | R/W | 0x00 | Function Address Function Address of Device as received through <code>SET_ADDRESS</code> . |

Register 2: USB Power (USBPOWER), offset 0x001

OTG A /
Host

USBPOWER is an 8-bit register used for controlling SUSPEND and RESUME signaling and some basic operational aspects of the USB controller.

OTG B /
Device

OTG A / Host Mode

USB Power (USBPOWER)

Base 0x4005.0000
Offset 0x001
Type R/W, reset 0x20

| | | | | | | | | |
|-------|----------|----|----|----|-------|--------|---------|----------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | RESET | RESUME | SUSPEND | PWRDNPHY |
| Type | RO | RO | RO | RO | R/W | R/W | R/W1S | R/W |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

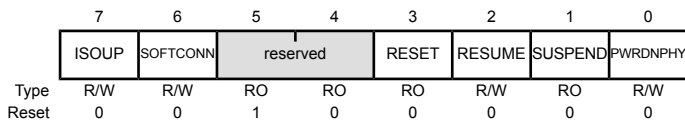
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|-------|---|
| 7:4 | reserved | RO | 0x2 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3 | RESET | R/W | 0 | RESET Signalling Value Description 1 Enables RESET signaling on the bus. 0 Ends RESET signaling on the bus. |
| 2 | RESUME | R/W | 0 | RESUME Signaling Value Description 1 Enables RESUME signaling when the Device is in SUSPEND mode. 0 Ends RESUME signaling on the bus. This bit must be cleared by software 20 ms after being set. |
| 1 | SUSPEND | R/W1S | 0 | SUSPEND Mode Value Description 1 Enables SUSPEND mode. 0 No effect. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|-------------------------------------|
| 0 | PWRDNPHY | R/W | 0 | Power Down PHY |
| | | | | Value Description |
| | | | | 1 Powers down the internal USB PHY. |
| | | | | 0 No effect. |

OTG B / Device Mode

USB Power (USBPOWER)

Base 0x4005.0000
 Offset 0x001
 Type R/W, reset 0x20



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | ISOUP | R/W | 0 | Isynchronous Update |
| | | | | Value Description |
| | | | | 1 The USB controller waits for an SOF token from the time the TXRDY bit is set in the USBTXCSRLn register before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent. |
| | | | | 0 No effect. |
| | | | | Note: This bit is only valid for isochronous transfers. |
| 6 | SOFTCONN | R/W | 0 | Soft Connect/Disconnect |
| | | | | Value Description |
| | | | | 1 The USB D+/D- lines are enabled. |
| | | | | 0 The USB D+/D- lines are tri-stated. |
| 5:4 | reserved | RO | 0x2 | Software should not rely on the value of 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 | RESET | RO | 0 | RESET Signalling |
| | | | | Value Description |
| | | | | 1 RESET signaling is present on the bus. |
| | | | | 0 RESET signaling is not present on the bus. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 2 | RESUME | R/W | 0 | <p>RESUME Signaling</p> <p>Value Description</p> <p>1 Enables RESUME signaling when the Device is in SUSPEND mode.</p> <p>0 Ends RESUME signaling on the bus.</p> <p>This bit must be cleared by software 10 ms (a maximum of 15 ms) after being set.</p> |
| 1 | SUSPEND | RO | 0 | <p>SUSPEND Mode</p> <p>Value Description</p> <p>1 The USB controller is in SUSPEND mode.</p> <p>0 This bit is cleared when software reads the interrupt register or sets the RESUME bit above.</p> |
| 0 | PWRDNPHY | R/W | 0 | <p>Power Down PHY</p> <p>Value Description</p> <p>1 Powers down the internal USB PHY.</p> <p>0 No effect.</p> |

Register 3: USB Transmit Interrupt Status (USBTXIS), offset 0x002

OTG A /
Host

USBTXIS is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–15. The meaning of the EP_n bits in this register is based on the mode of the device. The EP_1 through EP_{15} bits always indicate that the USB controller is sending data; however, in Host mode, the bits refer to OUT endpoints; while in Device mode, the bits refer to IN endpoints. The EP_0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

OTG B /
Device

Note: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

USB Transmit Interrupt Status (USBTXIS)

Base 0x4005.0000
Offset 0x002
Type RO, reset 0x0000

| | | | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| 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 |
|-----------|------|------|-------|---|
| 15 | EP15 | RO | 0 | TX Endpoint 15 Interrupt Value Description 0 No interrupt. 1 The Endpoint 15 transmit interrupt is asserted. |
| 14 | EP14 | RO | 0 | TX Endpoint 14 Interrupt Same description as EP15. |
| 13 | EP13 | RO | 0 | TX Endpoint 13 Interrupt Same description as EP15. |
| 12 | EP12 | RO | 0 | TX Endpoint 12 Interrupt Same description as EP15. |
| 11 | EP11 | RO | 0 | TX Endpoint 11 Interrupt Same description as EP15. |
| 10 | EP10 | RO | 0 | TX Endpoint 10 Interrupt Same description as EP15. |
| 9 | EP9 | RO | 0 | TX Endpoint 9 Interrupt Same description as EP15. |
| 8 | EP8 | RO | 0 | TX Endpoint 8 Interrupt Same description as EP15. |
| 7 | EP7 | RO | 0 | TX Endpoint 7 Interrupt Same description as EP15. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 6 | EP6 | RO | 0 | TX Endpoint 6 Interrupt Same description as EP15. |
| 5 | EP5 | RO | 0 | TX Endpoint 5 Interrupt Same description as EP15. |
| 4 | EP4 | RO | 0 | TX Endpoint 4 Interrupt Same description as EP15. |
| 3 | EP3 | RO | 0 | TX Endpoint 3 Interrupt Same description as EP15. |
| 2 | EP2 | RO | 0 | TX Endpoint 2 Interrupt Same description as EP15. |
| 1 | EP1 | RO | 0 | TX Endpoint 1 Interrupt Same description as EP15. |
| 0 | EP0 | RO | 0 | TX and RX Endpoint 0 Interrupt Same description as EP15. |

Register 4: USB Receive Interrupt Status (USBRIXIS), offset 0x004

OTG A /
Host

USBRIXIS is a 16-bit read-only register that indicates which of the interrupts for receive endpoints 1–15 are currently active.

Note: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

OTG B /
Device

USB Receive Interrupt Status (USBRIXIS)

Base 0x4005.0000
Offset 0x004
Type RO, reset 0x0000

| | | | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|----------|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 15 | EP15 | RO | 0 | RX Endpoint 15 Interrupt Value Description 0 No interrupt. 1 The Endpoint 15 receive interrupt is asserted. |
| 14 | EP14 | RO | 0 | RX Endpoint 14 Interrupt Same description as EP15. |
| 13 | EP13 | RO | 0 | RX Endpoint 13 Interrupt Same description as EP15. |
| 12 | EP12 | RO | 0 | RX Endpoint 12 Interrupt Same description as EP15. |
| 11 | EP11 | RO | 0 | RX Endpoint 11 Interrupt Same description as EP15. |
| 10 | EP10 | RO | 0 | RX Endpoint 10 Interrupt Same description as EP15. |
| 9 | EP9 | RO | 0 | RX Endpoint 9 Interrupt Same description as EP15. |
| 8 | EP8 | RO | 0 | RX Endpoint 8 Interrupt Same description as EP15. |
| 7 | EP7 | RO | 0 | RX Endpoint 7 Interrupt Same description as EP15. |
| 6 | EP6 | RO | 0 | RX Endpoint 6 Interrupt Same description as EP15. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 5 | EP5 | RO | 0 | RX Endpoint 5 Interrupt Same description as EP15. |
| 4 | EP4 | RO | 0 | RX Endpoint 4 Interrupt Same description as EP15. |
| 3 | EP3 | RO | 0 | RX Endpoint 3 Interrupt Same description as EP15. |
| 2 | EP2 | RO | 0 | RX Endpoint 2 Interrupt Same description as EP15. |
| 1 | EP1 | RO | 0 | RX Endpoint 1 Interrupt Same description as EP15. |
| 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: USB Transmit Interrupt Enable (USBTXIE), offset 0x006

OTG A /
Host

USBTXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the **USBTXIS** register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the **USBTXIS** register is set. When a bit is cleared, the interrupt in the **USBTXIS** register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

OTG B /
Device

USB Transmit Interrupt Enable (USBTXIE)

Base 0x4005.0000
Offset 0x006

Type R/W, reset 0xFFFF

| | | | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| 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 |
|-----------|------|------|-------|--|
| 15 | EP15 | R/W | 1 | TX Endpoint 15 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the EP15 bit in the USBTXIS register is set. 0 The EP15 transmit interrupt is suppressed and not sent to the interrupt controller. |
| 14 | EP14 | R/W | 1 | TX Endpoint 14 Interrupt Enable Same description as EP15. |
| 13 | EP13 | R/W | 1 | TX Endpoint 13 Interrupt Enable Same description as EP15. |
| 12 | EP12 | R/W | 1 | TX Endpoint 12 Interrupt Enable Same description as EP15. |
| 11 | EP11 | R/W | 1 | TX Endpoint 11 Interrupt Enable Same description as EP15. |
| 10 | EP10 | R/W | 1 | TX Endpoint 10 Interrupt Enable Same description as EP15. |
| 9 | EP9 | R/W | 1 | TX Endpoint 9 Interrupt Enable Same description as EP15. |
| 8 | EP8 | R/W | 1 | TX Endpoint 8 Interrupt Enable Same description as EP15. |
| 7 | EP7 | R/W | 1 | TX Endpoint 7 Interrupt Enable Same description as EP15. |
| 6 | EP6 | R/W | 1 | TX Endpoint 6 Interrupt Enable Same description as EP15. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 5 | EP5 | R/W | 1 | TX Endpoint 5 Interrupt Enable Same description as EP15. |
| 4 | EP4 | R/W | 1 | TX Endpoint 4 Interrupt Enable Same description as EP15. |
| 3 | EP3 | R/W | 1 | TX Endpoint 3 Interrupt Enable Same description as EP15. |
| 2 | EP2 | R/W | 1 | TX Endpoint 2 Interrupt Enable Same description as EP15. |
| 1 | EP1 | R/W | 1 | TX Endpoint 1 Interrupt Enable Same description as EP15. |
| 0 | EP0 | R/W | 1 | TX and RX Endpoint 0 Interrupt Enable Same description as EP15. |

Register 6: USB Receive Interrupt Enable (USBRXIE), offset 0x008

OTG A /
Host

USBRXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the **USBRXIS** register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the **USBRXIS** register is set. When a bit is cleared, the interrupt in the **USBRXIS** register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

OTG B /
Device

USB Receive Interrupt Enable (USBRXIE)

Base 0x4005.0000
Offset 0x008

Type R/W, reset 0xFFFFE

| | | | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|----------|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | reserved |
| 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 | RO |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 15 | EP15 | R/W | 1 | RX Endpoint 15 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the EP15 bit in the USBRXIS register is set. 0 The EP15 receive interrupt is suppressed and not sent to the interrupt controller. |
| 14 | EP14 | R/W | 1 | RX Endpoint 14 Interrupt Enable Same description as EP15. |
| 13 | EP13 | R/W | 1 | RX Endpoint 13 Interrupt Enable Same description as EP15. |
| 12 | EP12 | R/W | 1 | RX Endpoint 12 Interrupt Enable Same description as EP15. |
| 11 | EP11 | R/W | 1 | RX Endpoint 11 Interrupt Enable Same description as EP15. |
| 10 | EP10 | R/W | 1 | RX Endpoint 10 Interrupt Enable Same description as EP15. |
| 9 | EP9 | R/W | 1 | RX Endpoint 9 Interrupt Enable Same description as EP15. |
| 8 | EP8 | R/W | 1 | RX Endpoint 8 Interrupt Enable Same description as EP15. |
| 7 | EP7 | R/W | 1 | RX Endpoint 7 Interrupt Enable Same description as EP15. |
| 6 | EP6 | R/W | 1 | RX Endpoint 6 Interrupt Enable Same description as EP15. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 5 | EP5 | R/W | 1 | RX Endpoint 5 Interrupt Enable Same description as EP15. |
| 4 | EP4 | R/W | 1 | RX Endpoint 4 Interrupt Enable Same description as EP15. |
| 3 | EP3 | R/W | 1 | RX Endpoint 3 Interrupt Enable Same description as EP15. |
| 2 | EP2 | R/W | 1 | RX Endpoint 2 Interrupt Enable Same description as EP15. |
| 1 | EP1 | R/W | 1 | RX Endpoint 1 Interrupt Enable Same description as EP15. |
| 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: USB General Interrupt Status (USBIS), offset 0x00A

OTG A /
Host

USBIS is an 8-bit read-only register that indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

OTG B /
Device

OTG A / Host Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000

Offset 0x00A

Type RO, reset 0x00

| | | | | | | | | |
|-------|---------|--------|--------|------|-----|--------|--------|----------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | VBUSERR | SESREQ | DISCON | CONN | SOF | BABBLE | RESUME | reserved |
| Type | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

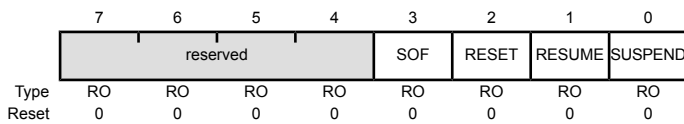
| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 7 | VBUSERR | RO | 0 | VBUS Error Value Description 1 VBUS has dropped below the VBUS Valid threshold during a session. 0 No interrupt. |
| 6 | SESREQ | RO | 0 | SESSION REQUEST Value Description 1 SESSION REQUEST signaling has been detected. 0 No interrupt. |
| 5 | DISCON | RO | 0 | Session Disconnect Value Description 1 A Device disconnect has been detected. 0 No interrupt. |
| 4 | CONN | RO | 0 | Session Connect Value Description 1 A Device connection has been detected. 0 No interrupt. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 3 | SOF | RO | 0 | Start of Frame Value Description 1 A new frame has started. 0 No interrupt. |
| 2 | BABBLE | RO | 0 | Babble Detected Value Description 1 Babble has been detected. This interrupt is active only after the first SOF has been sent. 0 No interrupt. |
| 1 | RESUME | RO | 0 | RESUME Signalling Detected Value Description 1 RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. 0 No interrupt. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS , USBDRIM , and USBDRISC registers should be used. |
| 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. |

OTG B / Device Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000
Offset 0x00A
Type RO, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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 | SOF | RO | 0 | Start of Frame Value Description 1 A new frame has started. 0 No interrupt. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 2 | RESET | RO | 0 | RESET Signalling Detected Value Description 1 RESET signaling has been detected on the bus. 0 No interrupt. |
| 1 | RESUME | RO | 0 | RESUME Signalling Detected Value Description 1 RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. 0 No interrupt. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS , USBDRIM , and USBDRISC registers should be used. |
| 0 | SUSPEND | RO | 0 | SUSPEND Signalling Detected Value Description 1 SUSPEND signaling has been detected on the bus. 0 No interrupt. |

Register 8: USB Interrupt Enable (USBIE), offset 0x00B

OTG A /
Host

USBIE is an 8-bit register that provides interrupt enable bits for each of the interrupts in **USBIS**. At reset interrupts 1 and 2 are enabled in Device mode.

OTG B /
Device

OTG A / Host Mode

USB Interrupt Enable (USBIE)

Base 0x4005.0000
Offset 0x00B
Type R/W, reset 0x06

| | | | | | | | | |
|-------|---------|--------|--------|------|-----|--------|--------|----------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | VBUSERR | SESREQ | DISCON | CONN | SOF | BABBLE | RESUME | reserved |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

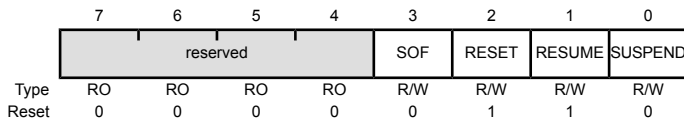
| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 7 | VBUSERR | R/W | 0 | Enable VBUS Error Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the VBUSERR bit in the USBIS register is set. 0 The VBUSERR interrupt is suppressed and not sent to the interrupt controller. |
| 6 | SESREQ | R/W | 0 | Enable Session Request Value Description 1 An interrupt is sent to the interrupt controller when the SESREQ bit in the USBIS register is set. 0 The SESREQ interrupt is suppressed and not sent to the interrupt controller. |
| 5 | DISCON | R/W | 0 | Enable Disconnect Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set. 0 The DISCON interrupt is suppressed and not sent to the interrupt controller. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 4 | CONN | R/W | 0 | <p>Enable Connect Interrupt</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the CONN bit in the USBIS register is set.</p> <p>0 The CONN interrupt is suppressed and not sent to the interrupt controller.</p> |
| 3 | SOF | R/W | 0 | <p>Enable Start-of-Frame Interrupt</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller SOF the CONN bit in the USBIS register is set.</p> <p>0 The SOF interrupt is suppressed and not sent to the interrupt controller.</p> |
| 2 | BABBLE | R/W | 1 | <p>Enable Babble Interrupt</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the BABBLE bit in the USBIS register is set.</p> <p>0 The BABBLE interrupt is suppressed and not sent to the interrupt controller.</p> |
| 1 | RESUME | R/W | 1 | <p>Enable RESUME Interrupt</p> <p>Value Description</p> <p>1 An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set.</p> <p>0 The RESUME interrupt is suppressed and not sent to the interrupt controller.</p> |
| 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. |

OTG B / Device Mode

USB Interrupt Enable (USBIE)

Base 0x4005.0000
 Offset 0x00B
 Type R/W, reset 0x06



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 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 | SOF | R/W | 0 | Enable Start-of-Frame Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set. 0 The SOF interrupt is suppressed and not sent to the interrupt controller. |
| 2 | RESET | R/W | 1 | Enable RESET Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESET bit in the USBIS register is set. 0 The RESET interrupt is suppressed and not sent to the interrupt controller. |
| 1 | RESUME | R/W | 1 | Enable RESUME Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set. 0 The RESUME interrupt is suppressed and not sent to the interrupt controller. |
| 0 | SUSPEND | R/W | 0 | Enable SUSPEND Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SUSPEND bit in the USBIS register is set. 0 The SUSPEND interrupt is suppressed and not sent to the interrupt controller. |

Register 9: USB Frame Value (USBFRAME), offset 0x00C

OTG A /
Host

USBFRAME is a 16-bit read-only register that holds the last received frame number.

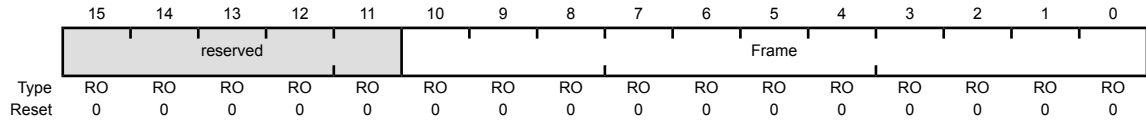
USB Frame Value (USBFRAME)

Base 0x4005.0000

Offset 0x00C

Type RO, reset 0x0000

OTG B /
Device



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 15:11 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10:0 | Frame | RO | 0x000 | Frame Number |

Register 10: USB Endpoint Index (USBEPIDX), offset 0x00E

**OTG A /
Host**

Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The **USBEPIDX** 16-bit register is used with the **USBTXFIFOSZ**, **USBRXFIFOSZ**, **USBTXFIFOADD**, and **USBRXFIFOADD** registers.

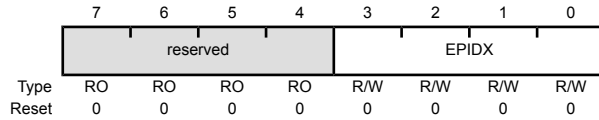
**OTG B /
Device**

USB Endpoint Index (USBEPIDX)

Base 0x4005.0000

Offset 0x00E

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 7:4 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3:0 | EPIDX | R/W | 0x0 | Endpoint Index This bit field configures which endpoint is accessed when reading or writing to one of the USB controller's indexed registers. A value of 0x0 corresponds to Endpoint 0 and a value of 0xF corresponds to Endpoint 15. |

Register 11: USB Test Mode (USBTEST), offset 0x00F

OTG A /
Host

USBTEST is an 8-bit register that is primarily used to put the USB controller into one of the four test modes for operation described in the *USB 2.0 Specification*, in response to a SET FEATURE: USBTESTMODE command. This register is not used in normal operation.

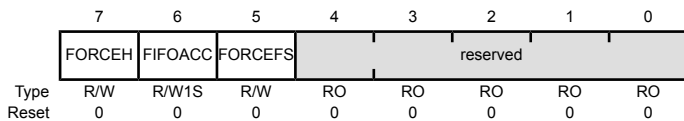
OTG B /
Device

Note: Only one of these bits should be set at any time.

OTG A / Host Mode

USB Test Mode (USBTEST)

Base 0x4005.0000
Offset 0x00F
Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-------------|
|-----------|------|------|-------|-------------|

| | | | | |
|---|--------|-----|---|-----------------|
| 7 | FORCEH | R/W | 0 | Force Host Mode |
|---|--------|-----|---|-----------------|

Value Description

1 Forces the USB controller to enter Host mode when the **SESSION** bit is set, regardless of whether the USB controller is connected to any peripheral. The state of the **USB0DP** and **USB0DM** signals is ignored. The USB controller then remains in Host mode until the **SESSION** bit is cleared, even if a Device is disconnected. If the **FORCEH** bit remains set, the USB controller re-enters Host mode the next time the **SESSION** bit is set.

0 No effect.

While in this mode, status of the bus connection may be read using the **DEV** bit of the **USBDEVCTL** register. The operating speed is determined from the **FORCEFS** bit.

| | | | | |
|---|---------|-------|---|-------------|
| 6 | FIFOACC | R/W1S | 0 | FIFO Access |
|---|---------|-------|---|-------------|

Value Description

1 Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.

0 No effect.

This bit is cleared automatically.

| | | | | |
|---|---------|-----|---|-----------------------|
| 5 | FORCEFS | R/W | 0 | Force Full-Speed Mode |
|---|---------|-----|---|-----------------------|

Value Description

1 Forces the USB controller into Full-Speed mode upon receiving a USB RESET.

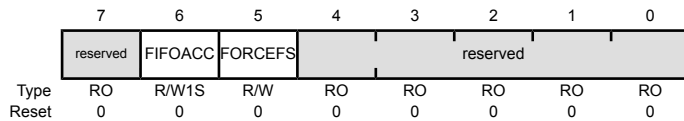
0 The USB controller operates at Low Speed.

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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. |

OTG B / Device Mode

USB Test Mode (USBTEST)

Base 0x4005.0000
 Offset 0x00F
 Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|-------|---|
| 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 | FIFOACC | R/W1S | 0 | FIFO Access Value Description 1 Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. 0 No effect. This bit is cleared automatically. |
| 5 | FORCEFS | R/W | 0 | Force Full-Speed Mode Value Description 1 Forces the USB controller into Full-Speed mode upon receiving a USB RESET. 0 The USB controller operates at Low Speed. |
| 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 12: USB FIFO Endpoint 0 (USBFIFO0), offset 0x020**
- Register 13: USB FIFO Endpoint 1 (USBFIFO1), offset 0x024**
- Register 14: USB FIFO Endpoint 2 (USBFIFO2), offset 0x028**
- Register 15: USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C**
- Register 16: USB FIFO Endpoint 4 (USBFIFO4), offset 0x030**
- Register 17: USB FIFO Endpoint 5 (USBFIFO5), offset 0x034**
- Register 18: USB FIFO Endpoint 6 (USBFIFO6), offset 0x038**
- Register 19: USB FIFO Endpoint 7 (USBFIFO7), offset 0x03C**
- Register 20: USB FIFO Endpoint 8 (USBFIFO8), offset 0x040**
- Register 21: USB FIFO Endpoint 9 (USBFIFO9), offset 0x044**
- Register 22: USB FIFO Endpoint 10 (USBFIFO10), offset 0x048**
- Register 23: USB FIFO Endpoint 11 (USBFIFO11), offset 0x04C**
- Register 24: USB FIFO Endpoint 12 (USBFIFO12), offset 0x050**
- Register 25: USB FIFO Endpoint 13 (USBFIFO13), offset 0x054**
- Register 26: USB FIFO Endpoint 14 (USBFIFO14), offset 0x058**
- Register 27: USB FIFO Endpoint 15 (USBFIFO15), offset 0x05C**

**OTG A /
Host**

These 32-bit registers provide an address for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

**OTG B /
Device**

Transfers to and from FIFOs may be 8-bit, 16-bit or 32-bit as required, and any combination of accesses is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, halfword- or word-aligned. However, the last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or double-packet buffering (see the section called “Single-Packet Buffering” on page 861). Burst writing of multiple packets is not supported as flags must be set after each packet is written.

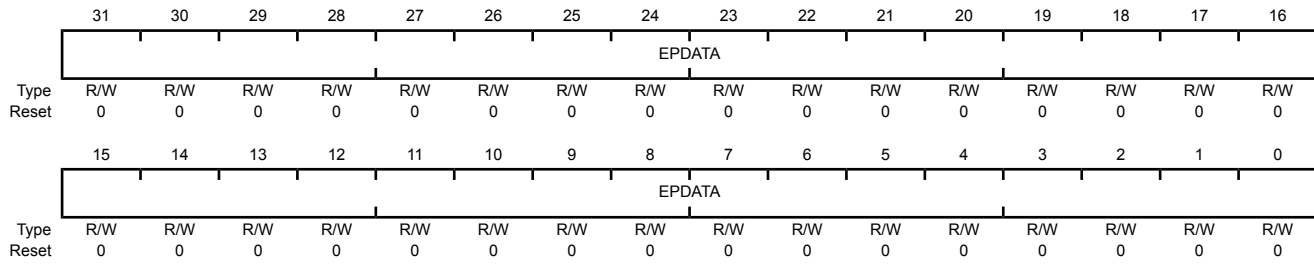
Following a STALL response or a transmit error on endpoint 1–15, the associated FIFO is completely flushed.

USB FIFO Endpoint 0 (USBFIFO0)

Base 0x4005.0000

Offset 0x020

Type R/W, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------------|---------------|
| 31:0 | EPDATA | R/W | 0x0000.0000 | Endpoint Data |

Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

Register 28: USB Device Control (USBDEVCTL), offset 0x060

**OTG A /
Host**

USBDEVCTL is an 8-bit register used for controlling and monitoring the USB VBUS line. If the PHY is suspended, no PHY clock is received and the VBUS is not sampled. In addition, in Host mode, **USBDEVCTL** provides the status information for the current operating mode (Host or Device) of the USB controller. If the USB controller is in Host mode, this register also indicates if a full- or low-speed Device has been connected.

USB Device Control (USBDEVCTL)

Base 0x4005.0000
Offset 0x060
Type R/W, reset 0x80

| | | | | | | | | |
|-------|-----|-------|-------|------|----|------|---------|---------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DEV | FSDEV | LSDEV | VBUS | | HOST | HOSTREQ | SESSION |
| Type | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-------------|
| 7 | DEV | RO | 1 | Device Mode |

| Value | Description |
|-------|--|
| 0 | The USB controller is operating as the OTG A device. In Host mode, the USB Controller is the Host. |
| 1 | The USB controller is operating as the OTG B device. In Host mode, the USB Controller is the Device. |

Note: This value is only valid while a session is in progress.

| | | | | |
|---|-------|----|---|----------------------------|
| 6 | FSDEV | RO | 0 | Full-Speed Device Detected |
|---|-------|----|---|----------------------------|

| Value | Description |
|-------|--|
| 0 | A full-speed Device has not been detected on the port. |
| 1 | A full-speed Device has been detected on the port. |

| | | | | |
|---|-------|----|---|---------------------------|
| 5 | LSDEV | RO | 0 | Low-Speed Device Detected |
|---|-------|----|---|---------------------------|

| Value | Description |
|-------|---|
| 0 | A low-speed Device has not been detected on the port. |
| 1 | A low-speed Device has been detected on the port. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 4:3 | VBUS | RO | 0x0 | <p>VBUS Level</p> <p>Value Description</p> <p>0x0 Below SessionEnd VBUS is detected as under 0.5 V.</p> <p>0x1 Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V.</p> <p>0x2 Above AValid, below VBUSValid VBUS is detected as above 1.5 V and below 4.5 V.</p> <p>0x3 Above VBUSValid VBUS is detected as above 4.5 V.</p> |
| 2 | HOST | RO | 0 | <p>Host Mode</p> <p>Value Description</p> <p>0 The USB controller is acting as a Device.</p> <p>1 The USB controller is acting as a Host.</p> <p>This bit is the logical NOT of the DEV bit.</p> <p>Note: This value is only valid while a session is in progress.</p> |
| 1 | HOSTREQ | R/W | 0 | <p>Host Request</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Initiates the Host Negotiation when SUSPEND mode is entered.</p> <p>This bit is cleared when Host Negotiation is completed.</p> |
| 0 | SESSION | R/W | 0 | <p>Session Start/End</p> <p><i>When operating as an OTG A device:</i></p> <p>Value Description</p> <p>0 When cleared by software, this bit ends a session.</p> <p>1 When set by software, this bit starts a session.</p> <p><i>When operating as an OTG B device:</i></p> <p>Value Description</p> <p>0 The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect.</p> <p>1 The USB controller has started a session. When set by software, the Session Request Protocol is initiated.</p> <p>Note: Clearing this bit when the USB controller is not suspended results in undefined behavior.</p> |

Register 29: USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062

Register 30: USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063

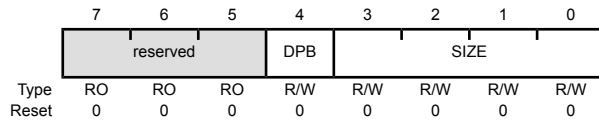
OTG A / Host

OTG B / Device

These 8-bit registers allow the selected TX/RX endpoint FIFOs to be dynamically sized. **USBEPIDX** is used to configure each transmit endpoint's FIFO size.

USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ)

Base 0x4005.0000
 Offset 0x062
 Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | | | | | | | |
|-----------|---------------------|------|-------|---|-------|---------------------|-----|---|-----|----|-----|----|-----|----|-----|-----|-----|-----|-----|-----|-----|------|-----|------|---------|----------|
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | | | | | | | | | | | | | |
| 4 | DPB | R/W | 0 | Double Packet Buffer Support Value Description 0 Only single-packet buffering is supported. 1 Double-packet buffering is supported. | | | | | | | | | | | | | | | | | | | | | | |
| 3:0 | SIZE | R/W | 0x0 | Max Packet Size Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size. <table border="1"> <thead> <tr> <th>Value</th> <th>Packet Size (Bytes)</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>8</td></tr> <tr><td>0x1</td><td>16</td></tr> <tr><td>0x2</td><td>32</td></tr> <tr><td>0x3</td><td>64</td></tr> <tr><td>0x4</td><td>128</td></tr> <tr><td>0x5</td><td>256</td></tr> <tr><td>0x6</td><td>512</td></tr> <tr><td>0x7</td><td>1024</td></tr> <tr><td>0x8</td><td>2048</td></tr> <tr><td>0x9-0xF</td><td>Reserved</td></tr> </tbody> </table> | Value | Packet Size (Bytes) | 0x0 | 8 | 0x1 | 16 | 0x2 | 32 | 0x3 | 64 | 0x4 | 128 | 0x5 | 256 | 0x6 | 512 | 0x7 | 1024 | 0x8 | 2048 | 0x9-0xF | Reserved |
| Value | Packet Size (Bytes) | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | 8 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | 16 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | 32 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | 64 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | 128 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | 256 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | 512 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | 1024 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | 2048 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9-0xF | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | |

Register 31: USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064

Register 32: USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066

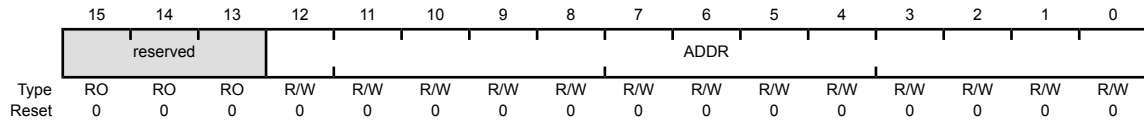
OTG A /
Host

USBTXFIFOADD and **USBRXFIFOADD** are 16-bit registers that controls the start address of the selected transmit and receive endpoint FIFOs.

USB Transmit FIFO Start Address (USBTXFIFOADD)

OTG B /
Device

Base 0x4005.0000
Offset 0x064
Type R/W, reset 0x0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|------------|---------------|-------|---|
| 15:13 | 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. |
| 12:0 | ADDR | R/W | 0x00 | Transmit/Receive Start Address Start address of the endpoint FIFO in units of 8 bytes. |
| | Value | Start Address | | |
| | 0x0 | 0 | | |
| | 0x1 | 8 | | |
| | 0x2 | 16 | | |
| | 0x3 | 32 | | |
| | 0x4 | 64 | | |
| | 0x5 | 128 | | |
| | 0x6 | 256 | | |
| | 0x7 | 512 | | |
| | 0x8 | 1024 | | |
| | 0x9 | 2048 | | |
| | 0xA-0x1FFF | Reserved | | |

Register 33: USB Connect Timing (USBCONTIM), offset 0x07A

OTG A /
Host

This 8-bit configuration register specifies connection and negotiation delays.

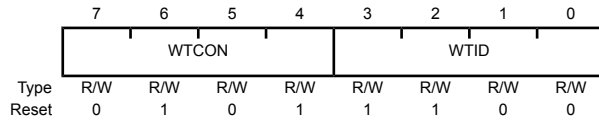
USB Connect Timing (USBCONTIM)

Base 0x4005.0000

Offset 0x07A

Type R/W, reset 0x5C

OTG B /
Device



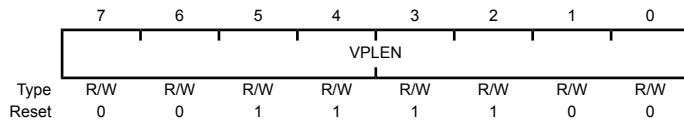
| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|---|
| 7:4 | WTCO | R/W | 0x5 | Connect Wait This field configures the wait required to allow for the user's connect/disconnect filter, in units of 533.3 ns. The default corresponds to 2.667 μ s. |
| 3:0 | WTID | R/W | 0xC | Wait ID This field configures the delay required from the enable of the ID detection to when the ID value is valid, in units of 4.369 ms. The default corresponds to 52.43 ms. |

Register 34: USB OTG VBUS Pulse Timing (USBVPLEN), offset 0x07B**OTG**

This 8-bit configuration register specifies the duration of the VBUS pulsing charge.

USB OTG VBUS Pulse Timing (USBVPLEN)

Base 0x4005.0000
 Offset 0x07B
 Type R/W, reset 0x3C



| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|-------------------|
| 7:0 | VPLEN | R/W | 0x3C | VBUS Pulse Length |

This field configures the duration of the VBUS pulsing charge in units of 546.1 μ s. The default corresponds to 32.77 ms.

Register 35: USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D

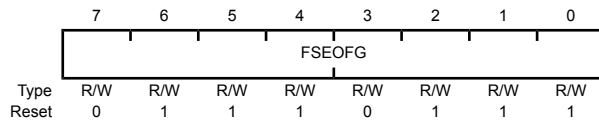
OTG A /
Host

This 8-bit configuration register specifies the minimum time gap allowed between the start of the last transaction and the EOF for full-speed transactions.

OTG B /
Device

USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF)

Base 0x4005.0000
Offset 0x07D
Type R/W, reset 0x77



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 7:0 | FSEOFG | R/W | 0x77 | Full-Speed End-of-Frame Gap This field is used during full-speed transactions to configure the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46 μ s. |

Register 36: USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E

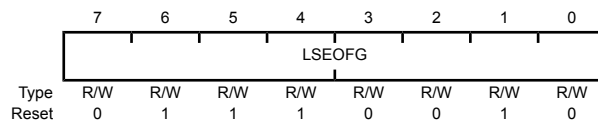
OTG A /
Host

This 8-bit configuration register specifies the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

OTG B /
Device

USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF)

Base 0x4005.0000
Offset 0x07E
Type R/W, reset 0x72



| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|----------------------------|
| 7:0 | LSEOFG | R/W | 0x72 | Low-Speed End-of-Frame Gap |

This field is used during low-speed transactions to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067 μ s. The default corresponds to 121.6 μ s.

Register 37: USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080

Register 38: USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088

Register 39: USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090

Register 40: USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098

Register 41: USB Transmit Functional Address Endpoint 4 (USBTXFUNCADDR4), offset 0x0A0

Register 42: USB Transmit Functional Address Endpoint 5 (USBTXFUNCADDR5), offset 0x0A8

Register 43: USB Transmit Functional Address Endpoint 6 (USBTXFUNCADDR6), offset 0x0B0

Register 44: USB Transmit Functional Address Endpoint 7 (USBTXFUNCADDR7), offset 0x0B8

Register 45: USB Transmit Functional Address Endpoint 8 (USBTXFUNCADDR8), offset 0x0C0

Register 46: USB Transmit Functional Address Endpoint 9 (USBTXFUNCADDR9), offset 0x0C8

Register 47: USB Transmit Functional Address Endpoint 10 (USBTXFUNCADDR10), offset 0x0D0

Register 48: USB Transmit Functional Address Endpoint 11 (USBTXFUNCADDR11), offset 0x0D8

Register 49: USB Transmit Functional Address Endpoint 12 (USBTXFUNCADDR12), offset 0x0E0

Register 50: USB Transmit Functional Address Endpoint 13 (USBTXFUNCADDR13), offset 0x0E8

Register 51: USB Transmit Functional Address Endpoint 14 (USBTXFUNCADDR14), offset 0x0F0

Register 52: USB Transmit Functional Address Endpoint 15 (USBTXFUNCADDR15), offset 0x0F8

OTG A /
Host

USBTXFUNCADDR_n is an 8-bit read/write register that records the address of the target function to be accessed through the associated endpoint (EP_n). USBTXFUNCADDR_n must be defined for each transmit endpoint that is used.

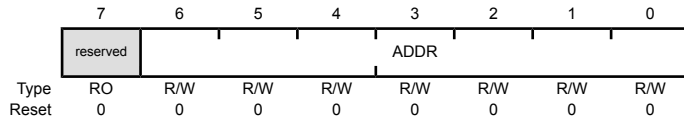
Note: USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0)

Base 0x4005.0000

Offset 0x080

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | ADDR | R/W | 0x00 | Device Address Specifies the USB bus address for the target Device. |

Register 53: USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082

Register 54: USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A

Register 55: USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092

Register 56: USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A

Register 57: USB Transmit Hub Address Endpoint 4 (USBTXHUBADDR4), offset 0x0A2

Register 58: USB Transmit Hub Address Endpoint 5 (USBTXHUBADDR5), offset 0x0AA

Register 59: USB Transmit Hub Address Endpoint 6 (USBTXHUBADDR6), offset 0x0B2

Register 60: USB Transmit Hub Address Endpoint 7 (USBTXHUBADDR7), offset 0x0BA

Register 61: USB Transmit Hub Address Endpoint 8 (USBTXHUBADDR8), offset 0x0C2

Register 62: USB Transmit Hub Address Endpoint 9 (USBTXHUBADDR9), offset 0x0CA

Register 63: USB Transmit Hub Address Endpoint 10 (USBTXHUBADDR10), offset 0x0D2

Register 64: USB Transmit Hub Address Endpoint 11 (USBTXHUBADDR11), offset 0x0DA

Register 65: USB Transmit Hub Address Endpoint 12 (USBTXHUBADDR12), offset 0x0E2

Register 66: USB Transmit Hub Address Endpoint 13 (USBTXHUBADDR13), offset 0x0EA

Register 67: USB Transmit Hub Address Endpoint 14 (USBTXHUBADDR14), offset 0x0F2

Register 68: USB Transmit Hub Address Endpoint 15 (USBTXHUBADDR15), offset 0x0FA

| |
|-----------------|
| OTG A / Host |
|-----------------|

USBTXHUBADDR_n is an 8-bit read/write register that, like USBTXHUBPORT_n, only must be written when a USB Device is connected to transmit endpoint EP_n via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

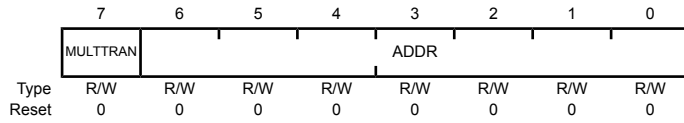
Note: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0)

Base 0x4005.0000

Offset 0x082

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 7 | MULTTRAN | R/W | 0 | Multiple Translators Value Description 0 Clear to indicate that the hub has a single transaction translator. 1 Set to indicate that the hub has multiple transaction translators. |
| 6:0 | ADDR | R/W | 0x00 | Hub Address This field specifies the USB bus address for the USB 2.0 hub. |

Register 69: USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083

Register 70: USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B

Register 71: USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093

Register 72: USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B

Register 73: USB Transmit Hub Port Endpoint 4 (USBTXHUBPORT4), offset 0x0A3

Register 74: USB Transmit Hub Port Endpoint 5 (USBTXHUBPORT5), offset 0x0AB

Register 75: USB Transmit Hub Port Endpoint 6 (USBTXHUBPORT6), offset 0x0B3

Register 76: USB Transmit Hub Port Endpoint 7 (USBTXHUBPORT7), offset 0x0BB

Register 77: USB Transmit Hub Port Endpoint 8 (USBTXHUBPORT8), offset 0x0C3

Register 78: USB Transmit Hub Port Endpoint 9 (USBTXHUBPORT9), offset 0x0CB

Register 79: USB Transmit Hub Port Endpoint 10 (USBTXHUBPORT10), offset 0x0D3

Register 80: USB Transmit Hub Port Endpoint 11 (USBTXHUBPORT11), offset 0x0DB

Register 81: USB Transmit Hub Port Endpoint 12 (USBTXHUBPORT12), offset 0x0E3

Register 82: USB Transmit Hub Port Endpoint 13 (USBTXHUBPORT13), offset 0x0EB

Register 83: USB Transmit Hub Port Endpoint 14 (USBTXHUBPORT14), offset 0x0F3

Register 84: USB Transmit Hub Port Endpoint 15 (USBTXHUBPORT15), offset 0x0FB

| |
|-----------------|
| OTG A / Host |
|-----------------|

USBTXHUBPORTn is an 8-bit read/write register that, like USBTXHUBADDRn, only must be written when a full- or low-speed Device is connected to transmit endpoint EPn via a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

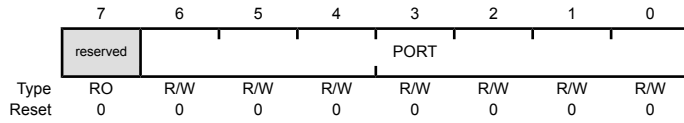
Note: USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0)

Base 0x4005.0000

Offset 0x083

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | PORT | R/W | 0x00 | Hub Port This field specifies the USB hub port number. |

Register 85: USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1), offset 0x08C

Register 86: USB Receive Functional Address Endpoint 2 (USBRXFUNCADDR2), offset 0x094

Register 87: USB Receive Functional Address Endpoint 3 (USBRXFUNCADDR3), offset 0x09C

Register 88: USB Receive Functional Address Endpoint 4 (USBRXFUNCADDR4), offset 0x0A4

Register 89: USB Receive Functional Address Endpoint 5 (USBRXFUNCADDR5), offset 0x0AC

Register 90: USB Receive Functional Address Endpoint 6 (USBRXFUNCADDR6), offset 0x0B4

Register 91: USB Receive Functional Address Endpoint 7 (USBRXFUNCADDR7), offset 0x0BC

Register 92: USB Receive Functional Address Endpoint 8 (USBRXFUNCADDR8), offset 0x0C4

Register 93: USB Receive Functional Address Endpoint 9 (USBRXFUNCADDR9), offset 0x0CC

Register 94: USB Receive Functional Address Endpoint 10 (USBRXFUNCADDR10), offset 0x0D4

Register 95: USB Receive Functional Address Endpoint 11 (USBRXFUNCADDR11), offset 0x0DC

Register 96: USB Receive Functional Address Endpoint 12 (USBRXFUNCADDR12), offset 0x0E4

Register 97: USB Receive Functional Address Endpoint 13 (USBRXFUNCADDR13), offset 0x0EC

Register 98: USB Receive Functional Address Endpoint 14 (USBRXFUNCADDR14), offset 0x0F4

Register 99: USB Receive Functional Address Endpoint 15 (USBRXFUNCADDR15), offset 0x0FC

**OTG A /
Host**

USBRXFUNCADDR_n is an 8-bit read/write register that records the address of the target function accessed through the associated endpoint (EP_n). **USBRXFUNCADDR_n** must be defined for each receive endpoint that is used.

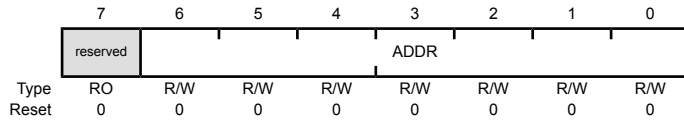
Note: **USBTXFUNCADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Functional Address Endpoint 1 (USBXFUNCADDR1)

Base 0x4005.0000

Offset 0x08C

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | ADDR | R/W | 0x00 | Device Address This field specifies the USB bus address for the target Device. |

Register 100: USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E

Register 101: USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096

Register 102: USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E

Register 103: USB Receive Hub Address Endpoint 4 (USBRXHUBADDR4), offset 0x0A6

Register 104: USB Receive Hub Address Endpoint 5 (USBRXHUBADDR5), offset 0x0AE

Register 105: USB Receive Hub Address Endpoint 6 (USBRXHUBADDR6), offset 0x0B6

Register 106: USB Receive Hub Address Endpoint 7 (USBRXHUBADDR7), offset 0x0BE

Register 107: USB Receive Hub Address Endpoint 8 (USBRXHUBADDR8), offset 0x0C6

Register 108: USB Receive Hub Address Endpoint 9 (USBRXHUBADDR9), offset 0x0CE

Register 109: USB Receive Hub Address Endpoint 10 (USBRXHUBADDR10), offset 0x0D6

Register 110: USB Receive Hub Address Endpoint 11 (USBRXHUBADDR11), offset 0x0DE

Register 111: USB Receive Hub Address Endpoint 12 (USBRXHUBADDR12), offset 0x0E6

Register 112: USB Receive Hub Address Endpoint 13 (USBRXHUBADDR13), offset 0x0EE

Register 113: USB Receive Hub Address Endpoint 14 (USBRXHUBADDR14), offset 0x0F6

Register 114: USB Receive Hub Address Endpoint 15 (USBRXHUBADDR15), offset 0x0FE

**OTG A /
Host**

USBRXHUBADDR_n is an 8-bit read/write register that, like **USBRXHUBPORT_n**, only must be written when a full- or low-speed Device is connected to receive endpoint EP_n via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

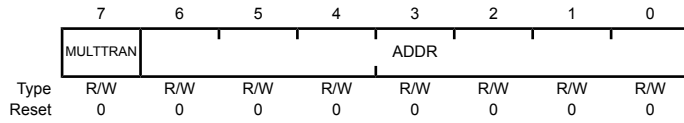
Note: **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1)

Base 0x4005.0000

Offset 0x08E

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 7 | MULTTRAN | R/W | 0 | Multiple Translators Value Description 0 Clear to indicate that the hub has a single transaction translator. 1 Set to indicate that the hub has multiple transaction translators. |
| 6:0 | ADDR | R/W | 0x00 | Hub Address This field specifies the USB bus address for the USB 2.0 hub. |

Register 115: USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F

Register 116: USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097

Register 117: USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F

Register 118: USB Receive Hub Port Endpoint 4 (USBRXHUBPORT4), offset 0x0A7

Register 119: USB Receive Hub Port Endpoint 5 (USBRXHUBPORT5), offset 0x0AF

Register 120: USB Receive Hub Port Endpoint 6 (USBRXHUBPORT6), offset 0x0B7

Register 121: USB Receive Hub Port Endpoint 7 (USBRXHUBPORT7), offset 0x0BF

Register 122: USB Receive Hub Port Endpoint 8 (USBRXHUBPORT8), offset 0x0C7

Register 123: USB Receive Hub Port Endpoint 9 (USBRXHUBPORT9), offset 0x0CF

Register 124: USB Receive Hub Port Endpoint 10 (USBRXHUBPORT10), offset 0x0D7

Register 125: USB Receive Hub Port Endpoint 11 (USBRXHUBPORT11), offset 0x0DF

Register 126: USB Receive Hub Port Endpoint 12 (USBRXHUBPORT12), offset 0x0E7

Register 127: USB Receive Hub Port Endpoint 13 (USBRXHUBPORT13), offset 0x0EF

Register 128: USB Receive Hub Port Endpoint 14 (USBRXHUBPORT14), offset 0x0F7

Register 129: USB Receive Hub Port Endpoint 15 (USBRXHUBPORT15), offset 0x0FF

**OTG A /
Host**

USBRXHUBPORT_n is an 8-bit read/write register that, like **USBRXHUBADDR_n**, only must be written when a full- or low-speed Device is connected to receive endpoint EP_n via a USB 2.0 hub. This register records the port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

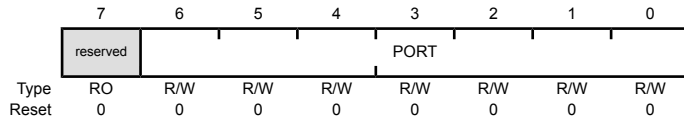
Note: **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1)

Base 0x4005.0000

Offset 0x08F

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | PORT | R/W | 0x00 | Hub Port This field specifies the USB hub port number. |

Register 130: USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110

Register 131: USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120

Register 132: USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130

Register 133: USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140

Register 134: USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150

Register 135: USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160

Register 136: USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170

Register 137: USB Maximum Transmit Data Endpoint 8 (USBTXMAXP8), offset 0x180

Register 138: USB Maximum Transmit Data Endpoint 9 (USBTXMAXP9), offset 0x190

Register 139: USB Maximum Transmit Data Endpoint 10 (USBTXMAXP10), offset 0x1A0

Register 140: USB Maximum Transmit Data Endpoint 11 (USBTXMAXP11), offset 0x1B0

Register 141: USB Maximum Transmit Data Endpoint 12 (USBTXMAXP12), offset 0x1C0

Register 142: USB Maximum Transmit Data Endpoint 13 (USBTXMAXP13), offset 0x1D0

Register 143: USB Maximum Transmit Data Endpoint 14 (USBTXMAXP14), offset 0x1E0

Register 144: USB Maximum Transmit Data Endpoint 15 (USBTXMAXP15), offset 0x1F0

**OTG A /
Host**

The **USBTXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the transmit endpoint in a single operation.

**OTG B /
Device**

Bits [10:0] define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operation.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO must be completely flushed (using the `FLUSH` bit in **USBTXCSRL1n**) after writing the new value to this register.

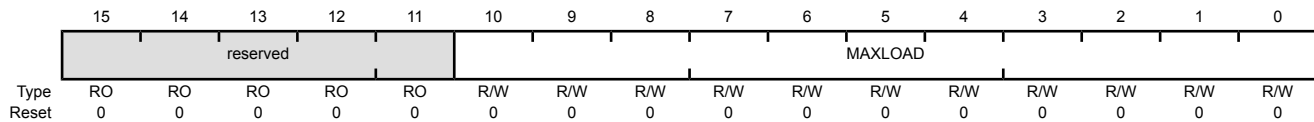
Note: **USBTXMAXPn** must be set to an even number of bytes for proper interrupt generation in μ DMA Basic Mode.

USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1)

Base 0x4005.0000

Offset 0x110

Type R/W, reset 0x0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 15:11 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10:0 | MAXLOAD | R/W | 0x000 | Maximum Payload This field specifies the maximum payload in bytes per transaction. |

Register 145: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102

OTG A /
Host

USBCSRL0 is an 8-bit register that provides control and status bits for endpoint 0.

OTG B /
Device

OTG A / Host Mode

USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000
Offset 0x102
Type W1C, reset 0x00

| | | | | | | | | |
|-------|-------|--------|--------|-------|-------|---------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | NAKTO | STATUS | REQPKT | ERROR | SETUP | STALLED | TXRDY | RXRDY |
| Type | 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 |
|-----------|--------|------|-------|--|
| 7 | NAKTO | R/W | 0 | <p>NAK Timeout</p> <p>Value Description</p> <p>0 No timeout.</p> <p>1 Indicates that endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the USBNAKLMT register.</p> <p>Software must clear this bit to allow the endpoint to continue.</p> |
| 6 | STATUS | R/W | 0 | <p>STATUS Packet</p> <p>Value Description</p> <p>0 No transaction.</p> <p>1 Initiates a STATUS stage transaction. This bit must be set at the same time as the TXRDY or REQPKT bit is set.</p> <p>Setting this bit ensures that the DT bit is set in the USBCSRH0 register so that a DATA1 packet is used for the STATUS stage transaction.</p> <p>This bit is automatically cleared when the STATUS stage is over.</p> |
| 5 | REQPKT | R/W | 0 | <p>Request Packet</p> <p>Value Description</p> <p>0 No request.</p> <p>1 Requests an IN transaction.</p> <p>This bit is cleared when the RXRDY bit is set.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 4 | ERROR | R/W | 0 | <p>Error</p> <p>Value Description</p> <p>0 No error.</p> <p>1 Three attempts have been made to perform a transaction with no response from the peripheral. The EP0 bit in the USBTXIS register is also set in this situation.</p> <p>Software must clear this bit.</p> |
| 3 | SETUP | R/W | 0 | <p>Setup Packet</p> <p>Value Description</p> <p>0 Sends an OUT token.</p> <p>1 Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.</p> <p>Setting this bit always clears the DT bit in the USBCSRH0 register to send a DATA0 packet.</p> |
| 2 | STALLED | R/W | 0 | <p>Endpoint Stalled</p> <p>Value Description</p> <p>0 No handshake has been received.</p> <p>1 A STALL handshake has been received.</p> <p>Software must clear this bit.</p> |
| 1 | TXRDY | R/W | 0 | <p>Transmit Packet Ready</p> <p>Value Description</p> <p>0 No transmit packet is ready.</p> <p>1 Software sets this bit after loading a data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.</p> <p>If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent.</p> <p>This bit is cleared automatically when the data packet has been transmitted.</p> |
| 0 | RXRDY | R/W | 0 | <p>Receive Packet Ready</p> <p>Value Description</p> <p>0 No received packet has been received.</p> <p>1 Indicates that a data packet has been received in the RX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.</p> <p>Software must clear this bit after the packet has been read from the FIFO to acknowledge that the data has been read from the FIFO.</p> |

OTG B / Device Mode

USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000
 Offset 0x102
 Type W1C, reset 0x00

| | | | | | | | | |
|-------|---------|--------|-------|--------|---------|---------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | SETENDC | RXRDYC | STALL | SETEND | DATAEND | STALLED | TXRDY | RXRDY |
| Type | W1C | W1C | R/W | RO | R/W | R/W | R/W | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 7 | SETENDC | W1C | 0 | Setup End Clear Writing a 1 to this bit clears the SETEND bit. |
| 6 | RXRDYC | W1C | 0 | RXRDY Clear Writing a 1 to this bit clears the RXRDY bit. |
| 5 | STALL | R/W | 0 | Send Stall Value Description 0 No effect. 1 Terminates the current transaction and transmits the STALL handshake. This bit is cleared automatically after the STALL handshake is transmitted. |
| 4 | SETEND | RO | 0 | Setup End Value Description 0 A control transaction has not ended or ended after the DATAEND bit was set. 1 A control transaction has ended before the DATAEND bit has been set. The EP0 bit in the USBTXIS register is also set in this situation. This bit is cleared by writing a 1 to the SETENDC bit. |
| 3 | DATAEND | R/W | 0 | Data End Value Description 0 No effect. 1 Set this bit in the following situations: <ul style="list-style-type: none"> ■ When setting TXRDY for the last data packet ■ When clearing RXRDY after unloading the last data packet ■ When setting TXRDY for a zero-length data packet This bit is cleared automatically. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 2 | STALLED | R/W | 0 | <p>Endpoint Stalled</p> <p>Value Description</p> <p>0 A STALL handshake has not been transmitted.</p> <p>1 A STALL handshake has been transmitted.</p> <p>Software must clear this bit.</p> |
| 1 | TXRDY | R/W | 0 | <p>Transmit Packet Ready</p> <p>Value Description</p> <p>0 No transmit packet is ready.</p> <p>1 Software sets this bit after loading an IN data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.</p> <p>This bit is cleared automatically when the data packet has been transmitted.</p> |
| 0 | RXRDY | RO | 0 | <p>Receive Packet Ready</p> <p>Value Description</p> <p>0 No data packet has been received.</p> <p>1 A data packet has been received. The EP0 bit in the USBTXIS register is also set in this situation.</p> <p>This bit is cleared by writing a 1 to the RXRDYC bit.</p> |

Register 146: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103

OTG A /
Host

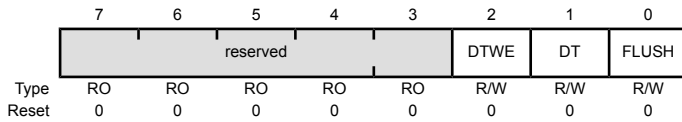
USBSR0H is an 8-bit register that provides control and status bits for endpoint 0.

OTG B /
Device

OTG A / Host Mode

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000
Offset 0x103
Type W1C, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 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 | DTWE | R/W | 0 | Data Toggle Write Enable Value Description 0 The DT bit cannot be written. 1 Enables the current state of the endpoint 0 data toggle to be written (see DT bit). This bit is automatically cleared once the new value is written. |
| 1 | DT | R/W | 0 | Data Toggle When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is set, this bit may be written with the required setting of the data toggle. If DTWE is Low, this bit cannot be written. Care should be taken when writing to this bit as it should only be changed to RESET USB endpoint 0. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|-------|------|-------|-------------|
| 0 | FLUSH | R/W | 0 | Flush FIFO |

Value Description

| | |
|---|--|
| 0 | No effect. |
| 1 | Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared. |

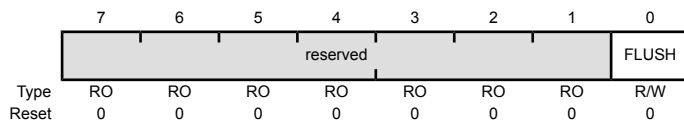
This bit is automatically cleared after the flush is performed.

Important: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

OTG B / Device Mode

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000
Offset 0x103
Type W1C, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 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 | FLUSH | R/W | 0 | Flush FIFO |

Value Description

| | |
|---|--|
| 0 | No effect. |
| 1 | Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared. |

This bit is automatically cleared after the flush is performed.

Important: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

Register 147: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108

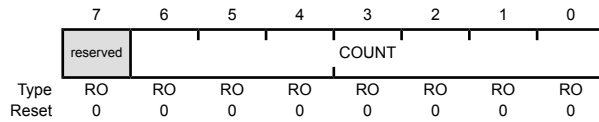
OTG A /
Host

USBCOUNT0 is an 8-bit read-only register that indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while the *RXRDY* bit is set.

OTG B /
Device

USB Receive Byte Count Endpoint 0 (USBCOUNT0)

Base 0x4005.0000
Offset 0x108
Type RO, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 6:0 | COUNT | RO | 0x00 | FIFO Count COUNT is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO. |

Register 148: USB Type Endpoint 0 (USBTYPE0), offset 0x10A

OTG A /
Host

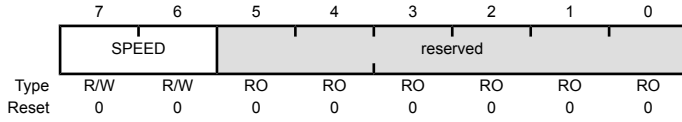
This is an 8-bit register that must be written with the operating speed of the targeted Device being communicated with using endpoint 0.

USB Type Endpoint 0 (USBTYPE0)

Base 0x4005.0000

Offset 0x10A

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7:6 | SPEED | R/W | 0x0 | Operating Speed |
| 5:0 | 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. |

| Value | Description |
|-----------|-------------|
| 0x0 - 0x1 | Reserved |
| 0x2 | Full |
| 0x3 | Low |

Register 149: USB NAK Limit (USBNAKLMT), offset 0x10B

OTG A /
Host

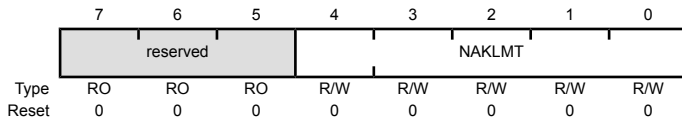
USBNAKLMT is an 8-bit register that sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their **USBTXINTERVALn** and **USBRXINTERVALn** registers.)

The number of frames selected is $2^{(m-1)}$ (where m is the value set in the register, with valid values of 2–16). If the Host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

Note: A value of 0 or 1 disables the NAK timeout function.

USB NAK Limit (USBNAKLMT)

Base 0x4005.0000
Offset 0x10B
Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 7:5 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 4:0 | NAKLMT | R/W | 0x0 | EP0 NAK Limit This field specifies the number of frames after receiving a stream of NAK responses. |

Register 150: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112

Register 151: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122

Register 152: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132

Register 153: USB Transmit Control and Status Endpoint 4 Low (USBTXCSRL4), offset 0x142

Register 154: USB Transmit Control and Status Endpoint 5 Low (USBTXCSRL5), offset 0x152

Register 155: USB Transmit Control and Status Endpoint 6 Low (USBTXCSRL6), offset 0x162

Register 156: USB Transmit Control and Status Endpoint 7 Low (USBTXCSRL7), offset 0x172

Register 157: USB Transmit Control and Status Endpoint 8 Low (USBTXCSRL8), offset 0x182

Register 158: USB Transmit Control and Status Endpoint 9 Low (USBTXCSRL9), offset 0x192

Register 159: USB Transmit Control and Status Endpoint 10 Low (USBTXCSRL10), offset 0x1A2

Register 160: USB Transmit Control and Status Endpoint 11 Low (USBTXCSRL11), offset 0x1B2

Register 161: USB Transmit Control and Status Endpoint 12 Low (USBTXCSRL12), offset 0x1C2

Register 162: USB Transmit Control and Status Endpoint 13 Low (USBTXCSRL13), offset 0x1D2

Register 163: USB Transmit Control and Status Endpoint 14 Low (USBTXCSRL14), offset 0x1E2

Register 164: USB Transmit Control and Status Endpoint 15 Low (USBTXCSRL15), offset 0x1F2

OTG A /
Host

USBTXCSRL_n is an 8-bit register that provides control and status bits for transfers through the currently selected transmit endpoint.

OTG B /
Device

OTG A / Host Mode

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000
 Offset 0x112
 Type R/W, reset 0x00

| | | | | | | | | |
|-------|-------|-------|---------|-------|-------|-------|--------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| Type | 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 |
|-----------|---------|------|-------|--|
| 7 | NAKTO | R/W | 0 | <p>NAK Timeout</p> <p>Value Description</p> <p>0 No timeout.</p> <p>1 <i>Bulk endpoints only:</i> Indicates that the transmit endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBXINTERVALn register. Software must clear this bit to allow the endpoint to continue.</p> |
| 6 | CLRDT | R/W | 0 | <p>Clear Data Toggle</p> <p>Writing a 1 to this bit clears the DT bit in the USBTXCSRHn register.</p> |
| 5 | STALLED | R/W | 0 | <p>Endpoint Stalled</p> <p>Value Description</p> <p>0 A STALL handshake has not been received.</p> <p>1 Indicates that a STALL handshake has been received. When this bit is set, any μDMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared.</p> <p>Software must clear this bit.</p> |
| 4 | SETUP | R/W | 0 | <p>Setup Packet</p> <p>Value Description</p> <p>0 No SETUP token is sent.</p> <p>1 Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.</p> <p>Note: Setting this bit also clears the DT bit in the USBTXCSRHn register.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 3 | FLUSH | R/W | 0 | <p>Flush FIFO</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation.</p> <p>This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.</p> <hr/> <p>Important: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.</p> <hr/> |
| 2 | ERROR | R/W | 0 | <p>Error</p> <p>Value Description</p> <p>0 No error.</p> <p>1 Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation.</p> <p>Software must clear this bit.</p> <p>Note: This is valid only when the endpoint is operating in Bulk or Interrupt mode.</p> |
| 1 | FIFONE | R/W | 0 | <p>FIFO Not Empty</p> <p>Value Description</p> <p>0 The FIFO is empty.</p> <p>1 At least one packet is in the transmit FIFO.</p> |
| 0 | TXRDY | R/W | 0 | <p>Transmit Packet Ready</p> <p>Value Description</p> <p>0 No transmit packet is ready.</p> <p>1 Software sets this bit after loading a data packet into the TX FIFO.</p> <p>This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the USBTXIS register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p> |

OTG B / Device Mode

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000
 Offset 0x112
 Type R/W, reset 0x00

| | | | | | | | | |
|-------|----------|-------|---------|-------|-------|-------|--------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| Type | RO | 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 |
|-----------|----------|------|-------|--|
| 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 | CLRDT | R/W | 0 | Clear Data Toggle Writing a 1 to this bit clears the DT bit in the USBTXCSRHn register. |
| 5 | STALLED | R/W | 0 | Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. The FIFO is flushed and the TXRDY bit is cleared. Software must clear this bit. |
| 4 | STALL | R/W | 0 | Send STALL Value Description 0 No effect. 1 Issues a STALL handshake to an IN token. Software clears this bit to terminate the STALL condition. Note: This bit has no effect in isochronous transfers. |
| 3 | FLUSH | R/W | 0 | Flush FIFO Value Description 0 No effect. 1 Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation. This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO. Important: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 2 | UNDRN | R/W | 0 | <p>Underrun</p> <p>Value Description</p> <p>0 No underrun.</p> <p>1 An IN token has been received when <code>TXRDY</code> is not set.</p> <p>Software must clear this bit.</p> |
| 1 | FIFONE | R/W | 0 | <p>FIFO Not Empty</p> <p>Value Description</p> <p>0 The FIFO is empty.</p> <p>1 At least one packet is in the transmit FIFO.</p> |
| 0 | TXRDY | R/W | 0 | <p>Transmit Packet Ready</p> <p>Value Description</p> <p>0 No transmit packet is ready.</p> <p>1 Software sets this bit after loading a data packet into the TX FIFO.</p> <p>This bit is cleared automatically when a data packet has been transmitted. The <code>EPn</code> bit in the USBTXIS register is also set at this point. <code>TXRDY</code> is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p> |

Register 165: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113

Register 166: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123

Register 167: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133

Register 168: USB Transmit Control and Status Endpoint 4 High (USBTXCSRH4), offset 0x143

Register 169: USB Transmit Control and Status Endpoint 5 High (USBTXCSRH5), offset 0x153

Register 170: USB Transmit Control and Status Endpoint 6 High (USBTXCSRH6), offset 0x163

Register 171: USB Transmit Control and Status Endpoint 7 High (USBTXCSRH7), offset 0x173

Register 172: USB Transmit Control and Status Endpoint 8 High (USBTXCSRH8), offset 0x183

Register 173: USB Transmit Control and Status Endpoint 9 High (USBTXCSRH9), offset 0x193

Register 174: USB Transmit Control and Status Endpoint 10 High (USBTXCSRH10), offset 0x1A3

Register 175: USB Transmit Control and Status Endpoint 11 High (USBTXCSRH11), offset 0x1B3

Register 176: USB Transmit Control and Status Endpoint 12 High (USBTXCSRH12), offset 0x1C3

Register 177: USB Transmit Control and Status Endpoint 13 High (USBTXCSRH13), offset 0x1D3

Register 178: USB Transmit Control and Status Endpoint 14 High (USBTXCSRH14), offset 0x1E3

Register 179: USB Transmit Control and Status Endpoint 15 High (USBTXCSRH15), offset 0x1F3

OTG A /
Host

USBTXCSRHn is an 8-bit register that provides additional control for transfers through the currently selected transmit endpoint.

OTG B /
Device

OTG A / Host Mode

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000
Offset 0x113
Type R/W, reset 0x00

| | | | | | | | | |
|-------|---------|----------|------|-------|-----|--------|------|-----|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | AUTOSET | reserved | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| Type | R/W | RO | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

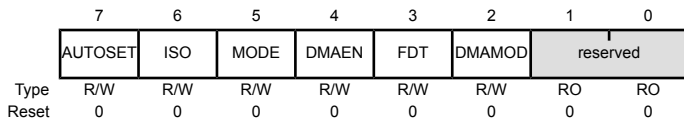
| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 7 | AUTOSET | R/W | 0 | Auto Set Value Description 0 The <code>TXRDY</code> bit must be set manually. 1 Enables the <code>TXRDY</code> bit to be automatically set when data of the maximum packet size (value in <code>USBTXMAXPn</code>) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the <code>TXRDY</code> bit must be set manually. |
| 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 | MODE | R/W | 0 | Mode Value Description 0 Enables the endpoint direction as RX. 1 Enables the endpoint direction as TX. Note: This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions. |
| 4 | DMAEN | R/W | 0 | DMA Request Enable Value Description 0 Disables the μ DMA request for the transmit endpoint. 1 Enables the μ DMA request for the transmit endpoint. Note: 3 TX and 3 /RX endpoints can be connected to the μ DMA module. If this bit is set for a particular endpoint, the <code>DMAATX</code> , <code>DMABTX</code> , or <code>DMACTX</code> field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly. |
| 3 | FDT | R/W | 0 | Force Data Toggle Value Description 0 No effect. 1 Forces the endpoint <code>DT</code> bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|--|
| 2 | DMAMOD | R/W | 0 | <p>DMA Request Mode</p> <p>Value Description</p> <p>0 An interrupt is generated after every μDMA packet transfer.</p> <p>1 An interrupt is generated only after the entire μDMA transfer is complete.</p> <p>Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.</p> |
| 1 | DTWE | R/W | 0 | <p>Data Toggle Write Enable</p> <p>Value Description</p> <p>0 The DT bit cannot be written.</p> <p>1 Enables the current state of the transmit endpoint data to be written (see DT bit).</p> <p>This bit is automatically cleared once the new value is written.</p> |
| 0 | DT | R/W | 0 | <p>Data Toggle</p> <p>When read, this bit indicates the current state of the transmit endpoint data toggle.</p> <p>If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the transmit endpoint.</p> |

OTG B / Device Mode

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000
 Offset 0x113
 Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 7 | AUTOSET | R/W | 0 | <p>Auto Set</p> <p>Value Description</p> <p>0 The TXRDY bit must be set manually.</p> <p>1 Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXPn) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 6 | ISO | R/W | 0 | <p>Isochronous Transfers</p> <p>Value Description</p> <p>0 Enables the transmit endpoint for bulk or interrupt transfers.</p> <p>1 Enables the transmit endpoint for isochronous transfers.</p> |
| 5 | MODE | R/W | 0 | <p>Mode</p> <p>Value Description</p> <p>0 Enables the endpoint direction as RX.</p> <p>1 Enables the endpoint direction as TX.</p> <p>Note: This bit only has an effect where the same endpoint FIFO is used for both transmit and receive transactions.</p> |
| 4 | DMAEN | R/W | 0 | <p>DMA Request Enable</p> <p>Value Description</p> <p>0 Disables the μDMA request for the transmit endpoint.</p> <p>1 Enables the μDMA request for the transmit endpoint.</p> <p>Note: 3 TX and 3 RX endpoints can be connected to the μDMA module. If this bit is set for a particular endpoint, the <i>DMAATX</i>, <i>DMABTX</i>, or <i>DMACTX</i> field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.</p> |
| 3 | FDT | R/W | 0 | <p>Force Data Toggle</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Forces the endpoint <i>DT</i> bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.</p> |
| 2 | DMAMOD | R/W | 0 | <p>DMA Request Mode</p> <p>Value Description</p> <p>0 An interrupt is generated after every μDMA packet transfer.</p> <p>1 An interrupt is generated only after the entire μDMA transfer is complete.</p> <p>Note: This bit must not be cleared either before or in the same cycle as the above <i>DMAEN</i> bit is cleared.</p> |
| 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 180: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114

Register 181: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124

Register 182: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134

Register 183: USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144

Register 184: USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154

Register 185: USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164

Register 186: USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174

Register 187: USB Maximum Receive Data Endpoint 8 (USBRXMAXP8), offset 0x184

Register 188: USB Maximum Receive Data Endpoint 9 (USBRXMAXP9), offset 0x194

Register 189: USB Maximum Receive Data Endpoint 10 (USBRXMAXP10), offset 0x1A4

Register 190: USB Maximum Receive Data Endpoint 11 (USBRXMAXP11), offset 0x1B4

Register 191: USB Maximum Receive Data Endpoint 12 (USBRXMAXP12), offset 0x1C4

Register 192: USB Maximum Receive Data Endpoint 13 (USBRXMAXP13), offset 0x1D4

Register 193: USB Maximum Receive Data Endpoint 14 (USBRXMAXP14), offset 0x1E4

Register 194: USB Maximum Receive Data Endpoint 15 (USBRXMAXP15), offset 0x1F4

**OTG A /
Host**

The **USBRXMAXPn** is a 16-bit register which defines the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

**OTG B /
Device**

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operations.

The total amount of data represented by the value written to this register must not exceed the FIFO size for the receive endpoint, and must not exceed half the FIFO size if double-buffering is required.

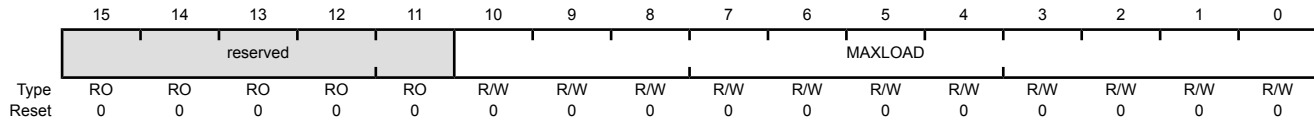
Note: **USBRXMAXPn** must be set to an even number of bytes for proper interrupt generation in μ DMA Basic mode.

USB Maximum Receive Data Endpoint 1 (USBRXMAXP1)

Base 0x4005.0000

Offset 0x114

Type R/W, reset 0x0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 15:11 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 10:0 | MAXLOAD | R/W | 0x000 | Maximum Payload The maximum payload in bytes per transaction. |

Register 195: USB Receive Control and Status Endpoint 1 Low (USBXCSRL1), offset 0x116

Register 196: USB Receive Control and Status Endpoint 2 Low (USBXCSRL2), offset 0x126

Register 197: USB Receive Control and Status Endpoint 3 Low (USBXCSRL3), offset 0x136

Register 198: USB Receive Control and Status Endpoint 4 Low (USBXCSRL4), offset 0x146

Register 199: USB Receive Control and Status Endpoint 5 Low (USBXCSRL5), offset 0x156

Register 200: USB Receive Control and Status Endpoint 6 Low (USBXCSRL6), offset 0x166

Register 201: USB Receive Control and Status Endpoint 7 Low (USBXCSRL7), offset 0x176

Register 202: USB Receive Control and Status Endpoint 8 Low (USBXCSRL8), offset 0x186

Register 203: USB Receive Control and Status Endpoint 9 Low (USBXCSRL9), offset 0x196

Register 204: USB Receive Control and Status Endpoint 10 Low (USBXCSRL10), offset 0x1A6

Register 205: USB Receive Control and Status Endpoint 11 Low (USBXCSRL11), offset 0x1B6

Register 206: USB Receive Control and Status Endpoint 12 Low (USBXCSRL12), offset 0x1C6

Register 207: USB Receive Control and Status Endpoint 13 Low (USBXCSRL13), offset 0x1D6

Register 208: USB Receive Control and Status Endpoint 14 Low (USBXCSRL14), offset 0x1E6

Register 209: USB Receive Control and Status Endpoint 15 Low (USBXCSRL15), offset 0x1F6

OTG A /
Host

USBXCSRL_n is an 8-bit register that provides control and status bits for transfers through the currently selected receive endpoint.

OTG B /
Device

OTG A / Host Mode

USB Receive Control and Status Endpoint 1 Low (USBXCSRL1)

Base 0x4005.0000

Offset 0x116

Type R/W, reset 0x00

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|-------|---------|--------|-------|--------------------|-------|------|-------|
| | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| Type | W1C | R/W | R/W | R/W | R/W | R/W | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|--|
| 7 | CLRDT | W1C | 0 | <p>Clear Data Toggle</p> <p>Writing a 1 to this bit clears the DT bit in the USBXCSRHn register.</p> |
| 6 | STALLED | R/W | 0 | <p>Endpoint Stalled</p> <p>Value Description</p> <p>0 A STALL handshake has not been received.</p> <p>1 A STALL handshake has been received. The EPn bit in the USBXIS register is also set.</p> <p>Software must clear this bit.</p> |
| 5 | REQPKT | R/W | 0 | <p>Request Packet</p> <p>Value Description</p> <p>0 No request.</p> <p>1 Requests an IN transaction.</p> <p>This bit is cleared when RXRDY is set.</p> |
| 4 | FLUSH | R/W | 0 | <p>Flush FIFO</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Flushes the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.</p> <p>Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.</p> |

Important: This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.

| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------------|------|-------|---|
| 3 | DATAERR / NAKTO | R/W | 0 | <p>Data Error / NAK Timeout</p> <p>Value Description</p> <p>0 Normal operation.</p> <p>1 <i>Isochronous endpoints only:</i> Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared.</p> <p><i>Bulk endpoints only:</i> Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBXINTERVALn register. Software must clear this bit to allow the endpoint to continue.</p> |
| 2 | ERROR | R/W | 0 | <p>Error</p> <p>Value Description</p> <p>0 No error.</p> <p>1 Three attempts have been made to receive a packet and no data packet has been received. The EPn bit in the USBXIS register is set in this situation.</p> <p>Software must clear this bit.</p> <p>Note: This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In Isochronous mode, it always returns zero.</p> |
| 1 | FULL | RO | 0 | <p>FIFO Full</p> <p>Value Description</p> <p>0 The receive FIFO is not full.</p> <p>1 No more packets can be loaded into the receive FIFO.</p> |
| 0 | RXRDY | R/W | 0 | <p>Receive Packet Ready</p> <p>Value Description</p> <p>0 No data packet has been received.</p> <p>1 A data packet has been received. The EPn bit in the USBXIS register is also set in this situation.</p> <p>If the AUTOCLR bit in the USBXCSRHn register is set, then this bit is automatically cleared when a packet of USBXMAXPn bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.</p> |

OTG B / Device Mode

USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1)

Base 0x4005.0000

Offset 0x116

Type R/W, reset 0x00

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|-------|---------|-------|-------|---------|------|------|-------|
| | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| Type | W1C | R/W | R/W | R/W | RO | R/W | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | |
|-----------|--|------|-------|---|---|---|---|--|
| 7 | CLRDT | W1C | 0 | <p>Clear Data Toggle</p> <p>Writing a 1 to this bit clears the <code>DT</code> bit in the <code>USBRXCSRHn</code> register.</p> | | | | |
| 6 | STALLED | R/W | 0 | <p>Endpoint Stalled</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>A STALL handshake has not been transmitted.</td> </tr> <tr> <td>1</td> <td>A STALL handshake has been transmitted.</td> </tr> </table> <p>Software must clear this bit.</p> | 0 | A STALL handshake has not been transmitted. | 1 | A STALL handshake has been transmitted. |
| 0 | A STALL handshake has not been transmitted. | | | | | | | |
| 1 | A STALL handshake has been transmitted. | | | | | | | |
| 5 | STALL | R/W | 0 | <p>Send STALL</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Issues a STALL handshake.</td> </tr> </table> <p>Software must clear this bit to terminate the STALL condition.</p> <p>Note: This bit has no effect where the endpoint is being used for isochronous transfers.</p> | 0 | No effect. | 1 | Issues a STALL handshake. |
| 0 | No effect. | | | | | | | |
| 1 | Issues a STALL handshake. | | | | | | | |
| 4 | FLUSH | R/W | 0 | <p>Flush FIFO</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the <code>RXRDY</code> bit is cleared.</td> </tr> </table> <p>The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the <code>RXRDY</code> bit is cleared. Note that if the FIFO is double-buffered, <code>FLUSH</code> may have to be set twice to completely clear the FIFO.</p> <p>Important: This bit should only be set when the <code>RXRDY</code> bit is set. At other times, it may cause data to be corrupted.</p> | 0 | No effect. | 1 | Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the <code>RXRDY</code> bit is cleared. |
| 0 | No effect. | | | | | | | |
| 1 | Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the <code>RXRDY</code> bit is cleared. | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|---------|------|-------|---|
| 3 | DATAERR | RO | 0 | <p>Data Error</p> <p>Value Description</p> <p>0 Normal operation.</p> <p>1 Indicates that <code>RXRDY</code> is set and the data packet has a CRC or bit-stuff error.</p> <p>This bit is cleared when <code>RXRDY</code> is cleared.</p> <p>Note: This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.</p> |
| 2 | OVER | R/W | 0 | <p>Overrun</p> <p>Value Description</p> <p>0 No overrun error.</p> <p>1 Indicates that an OUT packet cannot be loaded into the receive FIFO.</p> <p>Software must clear this bit.</p> <p>Note: This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.</p> |
| 1 | FULL | RO | 0 | <p>FIFO Full</p> <p>Value Description</p> <p>0 The receive FIFO is not full.</p> <p>1 No more packets can be loaded into the receive FIFO.</p> |
| 0 | RXRDY | R/W | 0 | <p>Receive Packet Ready</p> <p>Value Description</p> <p>0 No data packet has been received.</p> <p>1 A data packet has been received. The <code>EPn</code> bit in the <code>USBRXIS</code> register is also set in this situation.</p> <p>If the <code>AUTOCLR</code> bit in the <code>USBRXCSRHn</code> register is set, then the this bit is automatically cleared when a packet of <code>USBRXMAXPn</code> bytes has been unloaded from the receive FIFO. If the <code>AUTOCLR</code> bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.</p> |

Register 210: USB Receive Control and Status Endpoint 1 High (USBXCSRH1), offset 0x117

Register 211: USB Receive Control and Status Endpoint 2 High (USBXCSRH2), offset 0x127

Register 212: USB Receive Control and Status Endpoint 3 High (USBXCSRH3), offset 0x137

Register 213: USB Receive Control and Status Endpoint 4 High (USBXCSRH4), offset 0x147

Register 214: USB Receive Control and Status Endpoint 5 High (USBXCSRH5), offset 0x157

Register 215: USB Receive Control and Status Endpoint 6 High (USBXCSRH6), offset 0x167

Register 216: USB Receive Control and Status Endpoint 7 High (USBXCSRH7), offset 0x177

Register 217: USB Receive Control and Status Endpoint 8 High (USBXCSRH8), offset 0x187

Register 218: USB Receive Control and Status Endpoint 9 High (USBXCSRH9), offset 0x197

Register 219: USB Receive Control and Status Endpoint 10 High (USBXCSRH10), offset 0x1A7

Register 220: USB Receive Control and Status Endpoint 11 High (USBXCSRH11), offset 0x1B7

Register 221: USB Receive Control and Status Endpoint 12 High (USBXCSRH12), offset 0x1C7

Register 222: USB Receive Control and Status Endpoint 13 High (USBXCSRH13), offset 0x1D7

Register 223: USB Receive Control and Status Endpoint 14 High (USBXCSRH14), offset 0x1E7

Register 224: USB Receive Control and Status Endpoint 15 High (USBXCSRH15), offset 0x1F7

OTG A /
Host

USBXCSRHn is an 8-bit register that provides additional control and status bits for transfers through the currently selected receive endpoint.

OTG B /
Device

OTG A / Host Mode

USB Receive Control and Status Endpoint 1 High (USBXRCSRH1)

Base 0x4005.0000
 Offset 0x117
 Type R/W, reset 0x00

| | | | | | | | | |
|-------|--------|--------|-------|--------|--------|------|----|----------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | reserved |
| Type | R/W | R/W | R/W | RO | R/W | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 7 | AUTOCL | R/W | 0 | <p>Auto Clear</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Enables the <code>RXRDY</code> bit to be automatically cleared when a packet of <code>USBXRMAXPn</code> bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, <code>RXRDY</code> must be cleared manually. Care must be taken when using μDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the <code>MAXLOAD</code> field in the <code>USBXRMAXPn</code> register, see "DMA Operation" on page 870.</p> |
| 6 | AUTORQ | R/W | 0 | <p>Auto Request</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Enables the <code>REQPKT</code> bit to be automatically set when the <code>RXRDY</code> bit is cleared.</p> <p>Note: This bit is automatically cleared when a short packet is received.</p> |
| 5 | DMAEN | R/W | 0 | <p>DMA Request Enable</p> <p>Value Description</p> <p>0 Disables the μDMA request for the receive endpoint.</p> <p>1 Enables the μDMA request for the receive endpoint.</p> <p>Note: 3 TX and 3 RX endpoints can be connected to the μDMA module. If this bit is set for a particular endpoint, the <code>DMAARX</code>, <code>DMABRX</code>, or <code>DMACRX</code> field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.</p> |
| 4 | PIDERR | RO | 0 | <p>PID Error</p> <p>Value Description</p> <p>0 No error.</p> <p>1 Indicates a PID error in the received packet of an isochronous transaction.</p> <p>This bit is ignored in bulk or interrupt transactions.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 3 | DMAMOD | R/W | 0 | <p>DMA Request Mode</p> <p>Value Description</p> <p>0 An interrupt is generated after every μDMA packet transfer.</p> <p>1 An interrupt is generated only after the entire μDMA transfer is complete.</p> <p>Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.</p> |
| 2 | DTWE | RO | 0 | <p>Data Toggle Write Enable</p> <p>Value Description</p> <p>0 The DT bit cannot be written.</p> <p>1 Enables the current state of the receive endpoint data to be written (see DT bit).</p> <p>This bit is automatically cleared once the new value is written.</p> |
| 1 | DT | RO | 0 | <p>Data Toggle</p> <p>When read, this bit indicates the current state of the receive data toggle.</p> <p>If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored. Care should be taken when writing to this bit as it should only be changed to RESET the receive endpoint.</p> |
| 0 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |

OTG B / Device Mode

USB Receive Control and Status Endpoint 1 High (USBXCSRH1)

Base 0x4005.0000
 Offset 0x117
 Type R/W, reset 0x00

| | | | | | | | | |
|-------|--------|-----|-------|------------------|--------|----------|----|----|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | reserved | | |
| Type | R/W | R/W | R/W | R/W | R/W | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------------------|------|-------|---|
| 7 | AUTOCL | R/W | 0 | <p>Auto Clear</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Enables the <code>RXRDY</code> bit to be automatically cleared when a packet of <code>USBRXMAXPn</code> bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, <code>RXRDY</code> must be cleared manually. Care must be taken when using μDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the <code>MAXLOAD</code> field in the <code>USBRXMAXPn</code> register, see "DMA Operation" on page 870.</p> |
| 6 | ISO | R/W | 0 | <p>Isochronous Transfers</p> <p>Value Description</p> <p>0 Enables the receive endpoint for isochronous transfers.</p> <p>1 Enables the receive endpoint for bulk/interrupt transfers.</p> |
| 5 | DMAEN | R/W | 0 | <p>DMA Request Enable</p> <p>Value Description</p> <p>0 Disables the μDMA request for the receive endpoint.</p> <p>1 Enables the μDMA request for the receive endpoint.</p> <p>Note: 3 TX and 3 RX endpoints can be connected to the μDMA module. If this bit is set for a particular endpoint, the <code>DMAARX</code>, <code>DMABRX</code>, or <code>DMACRX</code> field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.</p> |
| 4 | DISNYET / PIDERR | R/W | 0 | <p>Disable NYET / PID Error</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 <i>For bulk or interrupt transactions:</i> Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full.</p> <p><i>For isochronous transactions:</i> Indicates a PID error in the received packet.</p> |
| 3 | DMAMOD | R/W | 0 | <p>DMA Request Mode</p> <p>Value Description</p> <p>0 An interrupt is generated after every μDMA packet transfer.</p> <p>1 An interrupt is generated only after the entire μDMA transfer is complete.</p> <p>Note: This bit must not be cleared either before or in the same cycle as the above <code>DMAEN</code> bit is cleared.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 2:0 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 225: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118

Register 226: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128

Register 227: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138

Register 228: USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148

Register 229: USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158

Register 230: USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168

Register 231: USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178

Register 232: USB Receive Byte Count Endpoint 8 (USBRXCOUNT8), offset 0x188

Register 233: USB Receive Byte Count Endpoint 9 (USBRXCOUNT9), offset 0x198

Register 234: USB Receive Byte Count Endpoint 10 (USBRXCOUNT10), offset 0x1A8

Register 235: USB Receive Byte Count Endpoint 11 (USBRXCOUNT11), offset 0x1B8

Register 236: USB Receive Byte Count Endpoint 12 (USBRXCOUNT12), offset 0x1C8

Register 237: USB Receive Byte Count Endpoint 13 (USBRXCOUNT13), offset 0x1D8

Register 238: USB Receive Byte Count Endpoint 14 (USBRXCOUNT14), offset 0x1E8

Register 239: USB Receive Byte Count Endpoint 15 (USBRXCOUNT15), offset 0x1F8

OTG A /
Host

Note: The value returned changes as the FIFO is unloaded and is only valid while the `RXRDY` bit in the `USBRXCSSLn` register is set.

OTG B /
Device

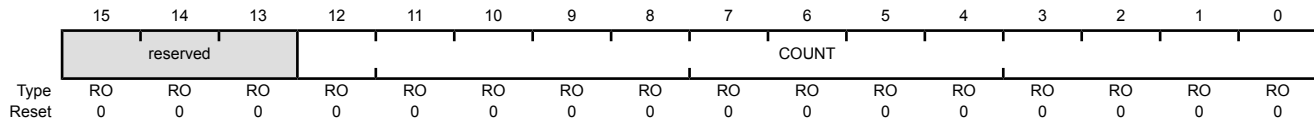
USBRXCOUNTn is a 16-bit read-only register that holds the number of data bytes in the packet currently in line to be read from the receive FIFO. If the packet is transmitted as multiple bulk packets, the number given is for the combined packet.

USB Receive Byte Count Endpoint 1 (USBRXCOUNT1)

Base 0x4005.0000

Offset 0x118

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. |
| 12:0 | COUNT | RO | 0x000 | Receive Packet Count Indicates the number of bytes in the receive packet. |

Register 240: USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A

Register 241: USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A

Register 242: USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A

Register 243: USB Host Transmit Configure Type Endpoint 4 (USBTXTYPE4), offset 0x14A

Register 244: USB Host Transmit Configure Type Endpoint 5 (USBTXTYPE5), offset 0x15A

Register 245: USB Host Transmit Configure Type Endpoint 6 (USBTXTYPE6), offset 0x16A

Register 246: USB Host Transmit Configure Type Endpoint 7 (USBTXTYPE7), offset 0x17A

Register 247: USB Host Transmit Configure Type Endpoint 8 (USBTXTYPE8), offset 0x18A

Register 248: USB Host Transmit Configure Type Endpoint 9 (USBTXTYPE9), offset 0x19A

Register 249: USB Host Transmit Configure Type Endpoint 10 (USBTXTYPE10), offset 0x1AA

Register 250: USB Host Transmit Configure Type Endpoint 11 (USBTXTYPE11), offset 0x1BA

Register 251: USB Host Transmit Configure Type Endpoint 12 (USBTXTYPE12), offset 0x1CA

Register 252: USB Host Transmit Configure Type Endpoint 13 (USBTXTYPE13), offset 0x1DA

Register 253: USB Host Transmit Configure Type Endpoint 14 (USBTXTYPE14), offset 0x1EA

Register 254: USB Host Transmit Configure Type Endpoint 15 (USBTXTYPE15), offset 0x1FA

**OTG A /
Host**

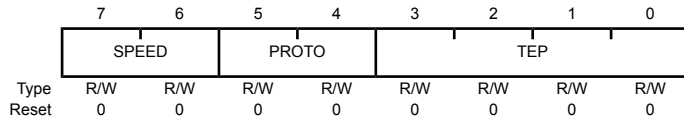
USBTXTYPE_n is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1)

Base 0x4005.0000

Offset 0x11A

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|-------------|------|-------|--|-------|-------------|-----|---------|-----|-------------|-----|------|-----|-----------|
| 7:6 | SPEED | R/W | 0x0 | <p>Operating Speed</p> <p>This bit field specifies the operating speed of the target Device:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Default</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Full</td> </tr> <tr> <td>0x3</td> <td>Low</td> </tr> </tbody> </table> | Value | Description | 0x0 | Default | 0x1 | Reserved | 0x2 | Full | 0x3 | Low |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Default | | | | | | | | | | | | | |
| 0x1 | Reserved | | | | | | | | | | | | | |
| 0x2 | Full | | | | | | | | | | | | | |
| 0x3 | Low | | | | | | | | | | | | | |
| 5:4 | PROTO | R/W | 0x0 | <p>Protocol</p> <p>Software must configure this bit field to select the required protocol for the transmit endpoint:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Control</td> </tr> <tr> <td>0x1</td> <td>Isochronous</td> </tr> <tr> <td>0x2</td> <td>Bulk</td> </tr> <tr> <td>0x3</td> <td>Interrupt</td> </tr> </tbody> </table> | Value | Description | 0x0 | Control | 0x1 | Isochronous | 0x2 | Bulk | 0x3 | Interrupt |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Control | | | | | | | | | | | | | |
| 0x1 | Isochronous | | | | | | | | | | | | | |
| 0x2 | Bulk | | | | | | | | | | | | | |
| 0x3 | Interrupt | | | | | | | | | | | | | |
| 3:0 | TEP | R/W | 0x0 | <p>Target Endpoint Number</p> <p>Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.</p> | | | | | | | | | | |

Register 255: USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B

Register 256: USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B

Register 257: USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B

Register 258: USB Host Transmit Interval Endpoint 4 (USBTXINTERVAL4), offset 0x14B

Register 259: USB Host Transmit Interval Endpoint 5 (USBTXINTERVAL5), offset 0x15B

Register 260: USB Host Transmit Interval Endpoint 6 (USBTXINTERVAL6), offset 0x16B

Register 261: USB Host Transmit Interval Endpoint 7 (USBTXINTERVAL7), offset 0x17B

Register 262: USB Host Transmit Interval Endpoint 8 (USBTXINTERVAL8), offset 0x18B

Register 263: USB Host Transmit Interval Endpoint 9 (USBTXINTERVAL9), offset 0x19B

Register 264: USB Host Transmit Interval Endpoint 10 (USBTXINTERVAL10), offset 0x1AB

Register 265: USB Host Transmit Interval Endpoint 11 (USBTXINTERVAL11), offset 0x1BB

Register 266: USB Host Transmit Interval Endpoint 12 (USBTXINTERVAL12), offset 0x1CB

Register 267: USB Host Transmit Interval Endpoint 13 (USBTXINTERVAL13), offset 0x1DB

Register 268: USB Host Transmit Interval Endpoint 14 (USBTXINTERVAL14), offset 0x1EB

Register 269: USB Host Transmit Interval Endpoint 15 (USBTXINTERVAL15), offset 0x1FB

| |
|-------------------------|
| <p>OTG A / Host</p> |
|-------------------------|

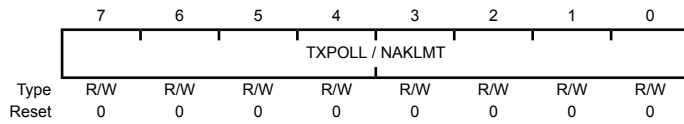
USBTXINTERVAL_n is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The **USBTXINTERVAL_n** register value defines a number of frames, as follows:

| Transfer Type | Speed | Valid values (m) | Interpretation |
|---------------|-------------------------|------------------|---|
| Interrupt | Low-Speed or Full-Speed | 0x01 – 0xFF | The polling interval is m frames. |
| Isochronous | Full-Speed | 0x01 – 0x10 | The polling interval is $2^{(m-1)}$ frames. |
| Bulk | Full-Speed | 0x02 – 0x10 | The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function. |

USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1)

Base 0x4005.0000
 Offset 0x11B
 Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------------|------|-------|------------------------|
| 7:0 | TXPOLL / NAKLMT | R/W | 0x00 | TX Polling / NAK Limit |

The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See table above for valid entries; other values are reserved.

Register 270: USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C

Register 271: USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C

Register 272: USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C

Register 273: USB Host Configure Receive Type Endpoint 4 (USBRXTYPE4), offset 0x14C

Register 274: USB Host Configure Receive Type Endpoint 5 (USBRXTYPE5), offset 0x15C

Register 275: USB Host Configure Receive Type Endpoint 6 (USBRXTYPE6), offset 0x16C

Register 276: USB Host Configure Receive Type Endpoint 7 (USBRXTYPE7), offset 0x17C

Register 277: USB Host Configure Receive Type Endpoint 8 (USBRXTYPE8), offset 0x18C

Register 278: USB Host Configure Receive Type Endpoint 9 (USBRXTYPE9), offset 0x19C

Register 279: USB Host Configure Receive Type Endpoint 10 (USBRXTYPE10), offset 0x1AC

Register 280: USB Host Configure Receive Type Endpoint 11 (USBRXTYPE11), offset 0x1BC

Register 281: USB Host Configure Receive Type Endpoint 12 (USBRXTYPE12), offset 0x1CC

Register 282: USB Host Configure Receive Type Endpoint 13 (USBRXTYPE13), offset 0x1DC

Register 283: USB Host Configure Receive Type Endpoint 14 (USBRXTYPE14), offset 0x1EC

Register 284: USB Host Configure Receive Type Endpoint 15 (USBRXTYPE15), offset 0x1FC

| |
|--------------------------------|
| <p>OTG A / Host</p> |
|--------------------------------|

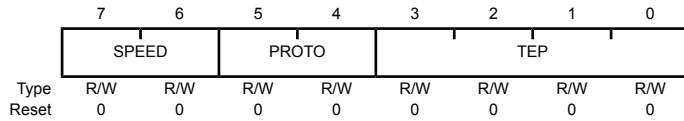
USBRXTYPE_n is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1)

Base 0x4005.0000

Offset 0x11C

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|-------------|------|-------|---|-------|-------------|-----|---------|-----|-------------|-----|------|-----|-----------|
| 7:6 | SPEED | R/W | 0x0 | <p>Operating Speed</p> <p>This bit field specifies the operating speed of the target Device:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Default</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Full</td> </tr> <tr> <td>0x3</td> <td>Low</td> </tr> </tbody> </table> | Value | Description | 0x0 | Default | 0x1 | Reserved | 0x2 | Full | 0x3 | Low |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Default | | | | | | | | | | | | | |
| 0x1 | Reserved | | | | | | | | | | | | | |
| 0x2 | Full | | | | | | | | | | | | | |
| 0x3 | Low | | | | | | | | | | | | | |
| 5:4 | PROTO | R/W | 0x0 | <p>Protocol</p> <p>Software must configure this bit field to select the required protocol for the receive endpoint:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Control</td> </tr> <tr> <td>0x1</td> <td>Isochronous</td> </tr> <tr> <td>0x2</td> <td>Bulk</td> </tr> <tr> <td>0x3</td> <td>Interrupt</td> </tr> </tbody> </table> | Value | Description | 0x0 | Control | 0x1 | Isochronous | 0x2 | Bulk | 0x3 | Interrupt |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Control | | | | | | | | | | | | | |
| 0x1 | Isochronous | | | | | | | | | | | | | |
| 0x2 | Bulk | | | | | | | | | | | | | |
| 0x3 | Interrupt | | | | | | | | | | | | | |
| 3:0 | TEP | R/W | 0x0 | <p>Target Endpoint Number</p> <p>Software must set this value to the endpoint number contained in the receive endpoint descriptor returned to the USB controller during Device enumeration.</p> | | | | | | | | | | |

Register 285: USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D

Register 286: USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D

Register 287: USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D

Register 288: USB Host Receive Polling Interval Endpoint 4 (USBRXINTERVAL4), offset 0x14D

Register 289: USB Host Receive Polling Interval Endpoint 5 (USBRXINTERVAL5), offset 0x15D

Register 290: USB Host Receive Polling Interval Endpoint 6 (USBRXINTERVAL6), offset 0x16D

Register 291: USB Host Receive Polling Interval Endpoint 7 (USBRXINTERVAL7), offset 0x17D

Register 292: USB Host Receive Polling Interval Endpoint 8 (USBRXINTERVAL8), offset 0x18D

Register 293: USB Host Receive Polling Interval Endpoint 9 (USBRXINTERVAL9), offset 0x19D

Register 294: USB Host Receive Polling Interval Endpoint 10 (USBRXINTERVAL10), offset 0x1AD

Register 295: USB Host Receive Polling Interval Endpoint 11 (USBRXINTERVAL11), offset 0x1BD

Register 296: USB Host Receive Polling Interval Endpoint 12 (USBRXINTERVAL12), offset 0x1CD

Register 297: USB Host Receive Polling Interval Endpoint 13 (USBRXINTERVAL13), offset 0x1DD

Register 298: USB Host Receive Polling Interval Endpoint 14 (USBRXINTERVAL14), offset 0x1ED

Register 299: USB Host Receive Polling Interval Endpoint 15 (USBRXINTERVAL15), offset 0x1FD

**OTG A /
Host**

USBRXINTERVAL_n is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected receive endpoint. For bulk endpoints, this register defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The **USBTXINTERVAL_n** register value defines a number of frames, as follows:

| Transfer Type | Speed | Valid values (m) | Interpretation |
|---------------|-------------------------|------------------|---|
| Interrupt | Low-Speed or Full-Speed | 0x01 – 0xFF | The polling interval is m frames. |
| Isochronous | Full-Speed | 0x01 – 0x10 | The polling interval is $2^{(m-1)}$ frames. |

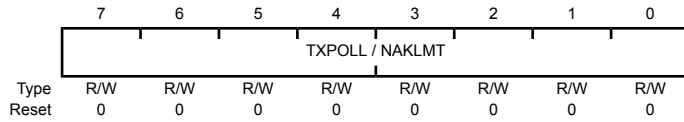
| Transfer Type | Speed | Valid values (m) | Interpretation |
|---------------|------------|------------------|---|
| Bulk | Full-Speed | 0x02 – 0x10 | The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function. |

USB Host Receive Polling Interval Endpoint 1 (USBXINTERVAL1)

Base 0x4005.0000

Offset 0x11D

Type R/W, reset 0x00



| Bit/Field | Name | Type | Reset | Description |
|-----------|-----------------|------|-------|------------------------|
| 7:0 | TXPOLL / NAKLMT | R/W | 0x00 | RX Polling / NAK Limit |

The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See table above for valid entries; other values are reserved.

Register 300: USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304

Register 301: USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308

Register 302: USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C

Register 303: USB Request Packet Count in Block Transfer Endpoint 4 (USBRQPKTCOUNT4), offset 0x310

Register 304: USB Request Packet Count in Block Transfer Endpoint 5 (USBRQPKTCOUNT5), offset 0x314

Register 305: USB Request Packet Count in Block Transfer Endpoint 6 (USBRQPKTCOUNT6), offset 0x318

Register 306: USB Request Packet Count in Block Transfer Endpoint 7 (USBRQPKTCOUNT7), offset 0x31C

Register 307: USB Request Packet Count in Block Transfer Endpoint 8 (USBRQPKTCOUNT8), offset 0x320

Register 308: USB Request Packet Count in Block Transfer Endpoint 9 (USBRQPKTCOUNT9), offset 0x324

Register 309: USB Request Packet Count in Block Transfer Endpoint 10 (USBRQPKTCOUNT10), offset 0x328

Register 310: USB Request Packet Count in Block Transfer Endpoint 11 (USBRQPKTCOUNT11), offset 0x32C

Register 311: USB Request Packet Count in Block Transfer Endpoint 12 (USBRQPKTCOUNT12), offset 0x330

Register 312: USB Request Packet Count in Block Transfer Endpoint 13 (USBRQPKTCOUNT13), offset 0x334

Register 313: USB Request Packet Count in Block Transfer Endpoint 14 (USBRQPKTCOUNT14), offset 0x338

Register 314: USB Request Packet Count in Block Transfer Endpoint 15 (USBRQPKTCOUNT15), offset 0x33C

| |
|--------------------------------|
| <p>OTG A / Host</p> |
|--------------------------------|

This 16-bit read/write register is used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint n. The USB controller uses the value recorded in this register to determine the number of requests to issue where the **AUTORQ** bit in the **USBRXCSRHn** register has been set. See "IN Transactions as a Host" on page 866.

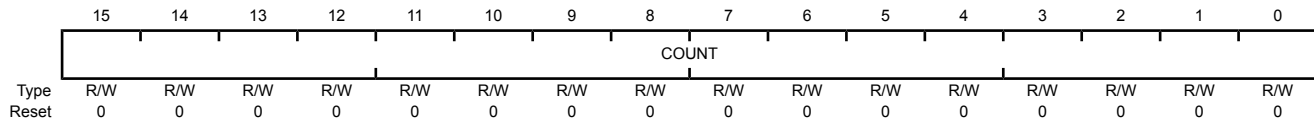
Note: Multiple packets combined into a single bulk packet within the FIFO count as one packet.

USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1)

Base 0x4005.0000

Offset 0x304

Type R/W, reset 0x0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|-------------|
|-----------|------|------|-------|-------------|

| | | | | |
|------|-------|-----|--------|-----------------------------|
| 15:0 | COUNT | R/W | 0x0000 | Block Transfer Packet Count |
|------|-------|-----|--------|-----------------------------|

Sets the number of packets of the size defined by the MAXLOAD bit field that are to be transferred in a block transfer.

Note: This is only used in Host mode when AUTORQ is set. The bit has no effect in Device mode or when AUTORQ is not set.

Register 315: USB Receive Double Packet Buffer Disable (USBXDPKTBUFDIS), offset 0x340

OTG A / Host

USBXDPKTBUFDIS is a 16-bit register that indicates which of the receive endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 862).

OTG B / Device

USB Receive Double Packet Buffer Disable (USBXDPKTBUFDIS)

Base 0x4005.0000
 Offset 0x340
 Type R/W, reset 0x0000

| | | | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|----------|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | reserved |
| 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 | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 15 | EP15 | R/W | 0 | EP15 RX Double-Packet Buffer Disable Value Description 0 Disables double-packet buffering. 1 Enables double-packet buffering. |
| 14 | EP14 | R/W | 0 | EP14 RX Double-Packet Buffer Disable Same description as EP15. |
| 13 | EP13 | R/W | 0 | EP13 RX Double-Packet Buffer Disable Same description as EP15. |
| 12 | EP12 | R/W | 0 | EP12 RX Double-Packet Buffer Disable Same description as EP15. |
| 11 | EP11 | R/W | 0 | EP11 RX Double-Packet Buffer Disable Same description as EP15. |
| 10 | EP10 | R/W | 0 | EP10 RX Double-Packet Buffer Disable Same description as EP15. |
| 9 | EP9 | R/W | 0 | EP9 RX Double-Packet Buffer Disable Same description as EP15. |
| 8 | EP8 | R/W | 0 | EP8 RX Double-Packet Buffer Disable Same description as EP15. |
| 7 | EP7 | R/W | 0 | EP7 RX Double-Packet Buffer Disable Same description as EP15. |
| 6 | EP6 | R/W | 0 | EP6 RX Double-Packet Buffer Disable Same description as EP15. |
| 5 | EP5 | R/W | 0 | EP5 RX Double-Packet Buffer Disable Same description as EP15. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 4 | EP4 | R/W | 0 | EP4 RX Double-Packet Buffer Disable Same description as EP15. |
| 3 | EP3 | R/W | 0 | EP3 RX Double-Packet Buffer Disable Same description as EP15. |
| 2 | EP2 | R/W | 0 | EP2 RX Double-Packet Buffer Disable Same description as EP15. |
| 1 | EP1 | R/W | 0 | EP1 RX Double-Packet Buffer Disable Same description as EP15. |
| 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 316: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFFDIS), offset 0x342

OTG A / Host

USBTXDPKTBUFFDIS is a 16-bit register that indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 861).

OTG B / Device

USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFFDIS)

Base 0x4005.0000
 Offset 0x342
 Type R/W, reset 0x0000

| | | | | | | | | | | | | | | | | |
|-------|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|----------|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | reserved |
| 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 | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|------|------|-------|--|
| 15 | EP15 | R/W | 0 | EP15 TX Double-Packet Buffer Disable Value Description 0 Disables double-packet buffering. 1 Enables double-packet buffering. |
| 14 | EP14 | R/W | 0 | EP14 TX Double-Packet Buffer Disable Same description as EP15. |
| 13 | EP13 | R/W | 0 | EP13 TX Double-Packet Buffer Disable Same description as EP15. |
| 12 | EP12 | R/W | 0 | EP12 TX Double-Packet Buffer Disable Same description as EP15. |
| 11 | EP11 | R/W | 0 | EP11 TX Double-Packet Buffer Disable Same description as EP15. |
| 10 | EP10 | R/W | 0 | EP10 TX Double-Packet Buffer Disable Same description as EP15. |
| 9 | EP9 | R/W | 0 | EP9 TX Double-Packet Buffer Disable Same description as EP15. |
| 8 | EP8 | R/W | 0 | EP8 TX Double-Packet Buffer Disable Same description as EP15. |
| 7 | EP7 | R/W | 0 | EP7 TX Double-Packet Buffer Disable Same description as EP15. |
| 6 | EP6 | R/W | 0 | EP6 TX Double-Packet Buffer Disable Same description as EP15. |
| 5 | EP5 | R/W | 0 | EP5 TX Double-Packet Buffer Disable Same description as EP15. |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 4 | EP4 | R/W | 0 | EP4 TX Double-Packet Buffer Disable Same description as EP15. |
| 3 | EP3 | R/W | 0 | EP3 TX Double-Packet Buffer Disable Same description as EP15. |
| 2 | EP2 | R/W | 0 | EP2 TX Double-Packet Buffer Disable Same description as EP15. |
| 1 | EP1 | R/W | 0 | EP1 TX Double-Packet Buffer Disable Same description as EP15. |
| 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 317: USB External Power Control (USBEPEN), offset 0x400

OTG A /
Host

This 32-bit register specifies the function of the two-pin external power interface (USB0EPEN and USB0PFLT). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary because the fault condition may require a response faster than one provided by firmware.

OTG B /
Device

USB External Power Control (USBEPEN)

Base 0x4005.0000

Offset 0x400

Type R/W, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|---------|----------|---------|---------|--------|----------|--------|------|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | PFLTACT | reserved | PFLTAEN | PFLTSEN | PFLTEN | reserved | EPENDE | EPEN | | |
| Type | RO | RO | RO | RO | RO | RO | 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 | | | | | | | | | | |
|-----------|--|------|----------|--|-------|-------------|-----|--|-----|---|-----|---|-----|---|
| 31:10 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |
| 9:8 | PFLTACT | R/W | 0x0 | <p>Power Fault Action</p> <p>This bit field specifies how the USB0EPEN signal is changed when detecting a USB power fault.</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0x0</td> <td> <p>Unchanged</p> <p>USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.</p> </td> </tr> <tr> <td>0x1</td> <td> <p>Tristate</p> <p>USB0EPEN is undriven (tristate).</p> </td> </tr> <tr> <td>0x2</td> <td> <p>Low</p> <p>USB0EPEN is driven Low.</p> </td> </tr> <tr> <td>0x3</td> <td> <p>High</p> <p>USB0EPEN is driven High.</p> </td> </tr> </table> | Value | Description | 0x0 | <p>Unchanged</p> <p>USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.</p> | 0x1 | <p>Tristate</p> <p>USB0EPEN is undriven (tristate).</p> | 0x2 | <p>Low</p> <p>USB0EPEN is driven Low.</p> | 0x3 | <p>High</p> <p>USB0EPEN is driven High.</p> |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | <p>Unchanged</p> <p>USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.</p> | | | | | | | | | | | | | |
| 0x1 | <p>Tristate</p> <p>USB0EPEN is undriven (tristate).</p> | | | | | | | | | | | | | |
| 0x2 | <p>Low</p> <p>USB0EPEN is driven Low.</p> | | | | | | | | | | | | | |
| 0x3 | <p>High</p> <p>USB0EPEN is driven High.</p> | | | | | | | | | | | | | |
| 7 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|--|
| 6 | PFLTAEN | R/W | 0 | <p>Power Fault Action Enable</p> <p>This bit specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the <code>USB0EPEN</code> signal.</p> <p>Value Description</p> <p>0 Disabled</p> <p><code>USB0EPEN</code> is controlled by the combination of the <code>EPEN</code> and <code>EPENDE</code> bits.</p> <p>1 Enabled</p> <p>The <code>USB0EPEN</code> output is automatically changed to the state specified by the <code>PFLTACT</code> field.</p> |
| 5 | PFLTSEN | R/W | 0 | <p>Power Fault Sense</p> <p>This bit specifies the logical sense of the <code>USB0PFLT</code> input signal that indicates an error condition.</p> <p>The complementary state is the inactive state.</p> <p>Value Description</p> <p>0 Low Fault</p> <p>If <code>USB0PFLT</code> is driven Low, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).</p> <p>1 High Fault</p> <p>If <code>USB0PFLT</code> is driven High, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).</p> |
| 4 | PFLTEN | R/W | 0 | <p>Power Fault Input Enable</p> <p>This bit specifies whether the <code>USB0PFLT</code> input signal is used in internal logic.</p> <p>Value Description</p> <p>0 Not Used</p> <p>The <code>USB0PFLT</code> signal is ignored.</p> <p>1 Used</p> <p>The <code>USB0PFLT</code> signal is used internally.</p> |
| 3 | reserved | RO | 0 | <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p> |

| Bit/Field | Name | Type | Reset | Description |
|-----------|--------|------|-------|---|
| 2 | EPENDE | R/W | 0 | <p>EPEN Drive Enable</p> <p>This bit specifies whether the <code>USB0EPEN</code> signal is driven or undriven (tristate). When driven, the signal value is specified by the <code>EPEN</code> field. When not driven, the <code>EPEN</code> field is ignored and the <code>USB0EPEN</code> signal is placed in a high-impedance state.</p> <p>Value Description</p> <p>0 Not Driven</p> <p>The <code>USB0EPEN</code> signal is high impedance.</p> <p>1 Driven</p> <p>The <code>USB0EPEN</code> signal is driven to the logical value specified by the value of the <code>EPEN</code> field.</p> <p>The <code>USB0EPEN</code> signal is undriven at reset because the sense of the external power supply enable is unknown. By adding the high-impedance state, system designers may bias the power supply enable to the disabled state using a large resistor (100 kΩ) and later configure and drive the output signal to enable the power supply.</p> |
| 1:0 | EPEN | R/W | 0x0 | <p>External Power Supply Enable Configuration</p> <p>This bit field specifies and controls the logical value driven on the <code>USB0EPEN</code> signal.</p> <p>Value Description</p> <p>0x0 Power Enable Active Low</p> <p>The <code>USB0EPEN</code> signal is driven Low if the <code>EPENDE</code> bit is set.</p> <p>0x1 Power Enable Active High</p> <p>The <code>USB0EPEN</code> signal is driven High if the <code>EPENDE</code> bit is set.</p> <p>0x2 Power Enable High if <code>VBUS</code> Low</p> <p>The <code>USB0EPEN</code> signal is driven High when the A device is not recognized.</p> <p>0x3 Power Enable High if <code>VBUS</code> High</p> <p>The <code>USB0EPEN</code> signal is driven High when the A device is recognized.</p> |

Register 318: USB External Power Control Raw Interrupt Status (USBEPCRIS), offset 0x404

OTG A /
Host

This 32-bit register specifies the unmasked interrupt status of the two-pin external power interface.

USB External Power Control Raw Interrupt Status (USBEPCRIS)

Base 0x4005.0000

Offset 0x404

Type RO, reset 0x0000.0000

OTG B /
Device

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 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 | | | | | | | | | | | | | | | PF |
| 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 |
|-----------|----------|------|------------|---|
| 31:1 | 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 | PF | RO | 0 | USB Power Fault Interrupt Status |
|---|----|----|---|----------------------------------|

Value Description

| | |
|---|---|
| 1 | A Power Fault status has been detected. |
| 0 | An interrupt has not occurred. |

This bit is cleared by writing a 1 to the PF bit in the **USBEPDISC** register.

Register 319: USB External Power Control Interrupt Mask (USBEPICIM), offset 0x408

OTG A /
Host

This 32-bit register specifies the interrupt mask of the two-pin external power interface.

USB External Power Control Interrupt Mask (USBEPICIM)

Base 0x4005.0000

Offset 0x408

Type R/W, reset 0x0000.0000

OTG B /
Device

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|
| | 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 | | | | | | | | | | | | | | | PF |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | PF | R/W | 0 | USB Power Fault Interrupt Mask |
| | | | | Value Description |
| | | | | 1 The raw interrupt signal from a detected power fault is sent to the interrupt controller. |
| | | | | 0 A detected power fault does not affect the interrupt status. |

Register 320: USB External Power Control Interrupt Status and Clear (USBEPICISC), offset 0x40C

OTG A /
Host

This 32-bit register specifies the masked interrupt status of the two-pin external power interface. It also provides a method to clear the interrupt state.

OTG B /
Device

USB External Power Control Interrupt Status and Clear (USBEPICISC)

Base 0x4005.0000
Offset 0x40C
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 | | | | | | | | | | | | | | | PF |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | PF | R/W1C | 0 | USB Power Fault Interrupt Status and Clear |

Value Description

- 1 The **PF** bits in the **USBEPICRIS** and **USBEPICIM** registers are set, providing an interrupt to the interrupt controller.
- 0 No interrupt has occurred or the interrupt is masked.

This bit is cleared by writing a 1. Clearing this bit also clears the **PF** bit in the **USBEPICRIS** register.

Register 321: USB Device RESUME Raw Interrupt Status (USBDRRIS), offset 0x410

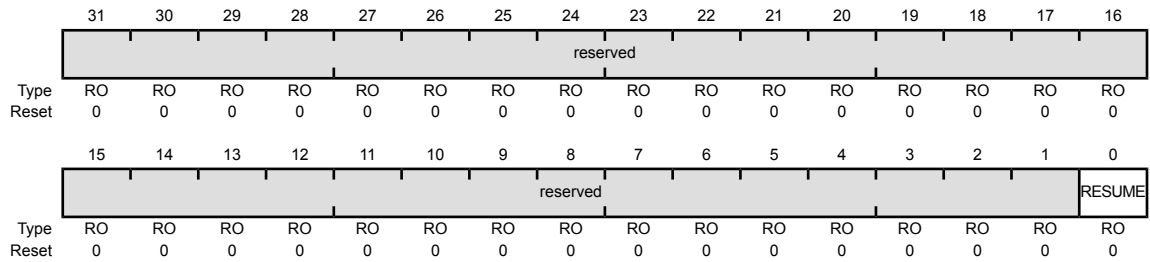
OTG A / Host

The **USBDRRIS** 32-bit 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.

OTG B / Device

USB Device RESUME Raw Interrupt Status (USBDRRIS)

Base 0x4005.0000
 Offset 0x410
 Type RO, reset 0x0000.0000



| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | RESUME | RO | 0 | RESUME Interrupt Status |
| | | | | Value Description |
| | | | | 1 A RESUME status has been detected. |
| | | | | 0 An interrupt has not occurred. |
| | | | | This bit is cleared by writing a 1 to the RESUME bit in the USBDRISC register. |

Register 322: USB Device RESUME Interrupt Mask (USBDRIM), offset 0x414OTG A /
Host

The **USBDRIM** 32-bit 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.

OTG B /
Device

USB Device RESUME Interrupt Mask (USBDRIM)

Base 0x4005.0000

Offset 0x414

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 | | | | | | | | | | | | | | | RESUME |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|-------|---|
| 31:1 | reserved | RO | 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 | RESUME | R/W | 0 | RESUME Interrupt Mask |

Value Description

- | | |
|---|---|
| 1 | The raw interrupt signal from a detected RESUME is sent to the interrupt controller. This bit should only be set when a SUSPEND has been detected (the <code>SUSPEND</code> bit in the USBIS register is set). |
| 0 | A detected RESUME does not affect the interrupt status. |

Register 323: USB Device RESUME Interrupt Status and Clear (USBDRISC), offset 0x418

OTG A / Host

The **USBDRISC** 32-bit register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

OTG B / Device

USB Device RESUME Interrupt Status and Clear (USBDRISC)

Base 0x4005.0000
Offset 0x418
Type 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 | | | | | | | | | | | | | | | RESUME |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | RESUME | R/W1C | 0 | RESUME Interrupt Status and Clear |

| Value | Description |
|-------|--|
| 1 | The RESUME bits in the USBDRRIS and USBDRCIM registers are set, providing an interrupt to the interrupt controller. |
| 0 | No interrupt has occurred or the interrupt is masked. |

This bit is cleared by writing a 1. Clearing this bit also clears the **RESUME** bit in the **USBDRCRIS** register.

Register 324: USB General-Purpose Control and Status (USBGPCS), offset 0x41C

OTG A /
Host

USBGPCS provides the state of the internal ID signal.

USB General-Purpose Control and Status (USBGPCS)

Base 0x4005.0000

Offset 0x41C

Type R/W, reset 0x0000.0000

OTG B /
Device

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|-----------|--------|-----|
| | 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 | | | | | | | | | | | | | | DEVMODOTG | DEVMOD | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description | | | | | | |
|-----------|---|------|------------|--|-------|-------------|---|---|---|---|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. | | | | | | |
| 1 | DEVMODOTG | R/W | 0 | <p>Enable Device Mode</p> <p>This bit enables the <code>DEVMOD</code> bit to control the state of the internal ID signal in OTG mode.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The mode is specified by the state of the internal ID signal.</td> </tr> <tr> <td>1</td> <td>This bit enables the <code>DEVMOD</code> bit to control the internal ID signal.</td> </tr> </tbody> </table> | Value | Description | 0 | The mode is specified by the state of the internal ID signal. | 1 | This bit enables the <code>DEVMOD</code> bit to control the internal ID signal. |
| Value | Description | | | | | | | | | |
| 0 | The mode is specified by the state of the internal ID signal. | | | | | | | | | |
| 1 | This bit enables the <code>DEVMOD</code> bit to control the internal ID signal. | | | | | | | | | |
| 0 | DEVMOD | R/W | 0 | <p>Device Mode</p> <p>This bit specifies the state of the internal ID signal in Host mode and in OTG mode when the <code>DEVMODOTG</code> bit is set.</p> <p>In Device mode this bit is ignored (assumed set).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Host mode</td> </tr> <tr> <td>1</td> <td>Device mode</td> </tr> </tbody> </table> | Value | Description | 0 | Host mode | 1 | Device mode |
| Value | Description | | | | | | | | | |
| 0 | Host mode | | | | | | | | | |
| 1 | Device mode | | | | | | | | | |

Register 325: USB VBUS Droop Control (USBVDC), offset 0x430

**OTG A /
Host**

This 32-bit register enables a controlled masking of VBUS to compensate for any in-rush current by a Device that is connected to the Host controller. The in-rush current can cause VBUS to droop, causing the USB controller's behavior to be unexpected. The USB Host controller allows VBUS to fall lower than the VBUS Valid level (4.5 V) but not below AValid (2.0 V) for 65 microseconds without signaling a `VBUSERR` interrupt in the controller. Without this, any glitch on VBUS would force the USB Host controller to remove power from VBUS and then re-enumerate the Device.

USB VBUS Droop Control (USBVDC)

Base 0x4005.0000
Offset 0x430
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 | | | | | | | | | | | | | | | VBDEN |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | VBDEN | R/W | 0 | VBUS Droop Enable |
| | | | | Value Description |
| | | | 0 | No effect. |
| | | | 1 | Any changes from VBUSVALID are masked when VBUS goes below 4.5 V but not lower than 2.0 V for 65 microseconds. During this time, the VBUS state indicates VBUSVALID. |

Register 326: USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS), offset 0x434

OTG A /
Host

This 32-bit register specifies the unmasked interrupt status of the VBUS droop limit of 65 microseconds.

USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS)

Base 0x4005.0000

Offset 0x434

Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | VD |
| 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 |
|-----------|----------|------|------------|---|
| 31:1 | 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 | VD | RO | 0 | VBUS Droop Raw Interrupt Status |

Value Description

| | |
|---|---|
| 1 | A VBUS droop lasting for 65 microseconds has been detected. |
| 0 | An interrupt has not occurred. |

This bit is cleared by writing a 1 to the VD bit in the **USBVDCISC** register.

Register 327: USB VBUS Droop Control Interrupt Mask (USBVDCIM), offset 0x438

OTG A / Host

This 32-bit register specifies the interrupt mask of the VBUS droop.

USB VBUS Droop Control Interrupt Mask (USBVDCIM)

Base 0x4005.0000

Offset 0x438

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 | | | | | | | | | | | | | | | VD |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | VD | R/W | 0 | VBUS Droop Interrupt Mask |
| | | | | Value Description |
| | | | 1 | The raw interrupt signal from a detected VBUS droop is sent to the interrupt controller. |
| | | | 0 | A detected VBUS droop does not affect the interrupt status. |

Register 328: USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC), offset 0x43C

OTG A /
Host

This 32-bit register specifies the masked interrupt status of the VBUS droop and provides a method to clear the interrupt state.

USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC)

Base 0x4005.0000

Offset 0x43C

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 | | | | | | | | | | | | | | | VD |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | VD | R/W1C | 0 | VBUS Droop Interrupt Status and Clear |

Value Description

- | | |
|---|---|
| 1 | The \overline{VD} bits in the USBVDCRIS and USBVDCIM registers are set, providing an interrupt to the interrupt controller. |
| 0 | No interrupt has occurred or the interrupt is masked. |

This bit is cleared by writing a 1. Clearing this bit also clears the \overline{VD} bit in the **USBVDCRIS** register.

Register 329: USB ID Valid Detect Raw Interrupt Status (USBIDVRIS), offset 0x444

OTG

This 32-bit register specifies whether the unmasked interrupt status of the ID value is valid.

USB ID Valid Detect Raw Interrupt Status (USBIDVRIS)

Base 0x4005.0000
 Offset 0x444
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | reserved | | | | | | | | | | | | | | | ID |
| 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 |
|-----------|----------|------|------------|---|
| 31:1 | 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 | ID | RO | 0 | ID Valid Detect Raw Interrupt Status |
|---|----|----|---|--------------------------------------|

| Value | Description |
|-------|--------------------------------|
| 1 | A valid ID has been detected. |
| 0 | An interrupt has not occurred. |

This bit is cleared by writing a 1 to the ID bit in the **USBIDVISC** register.

Register 330: USB ID Valid Detect Interrupt Mask (USBIDVIM), offset 0x448**OTG**

This 32-bit register specifies the interrupt mask of the ID valid detection.

USB ID Valid Detect Interrupt Mask (USBIDVIM)

Base 0x4005.0000
 Offset 0x448
 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 | | | | | | | | | | | | | | | ID |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

Value Description

- | | |
|---|--|
| 1 | The raw interrupt signal from a detected ID valid is sent to the interrupt controller. |
| 0 | A detected ID valid does not affect the interrupt status. |

Register 331: USB ID Valid Detect Interrupt Status and Clear (USBIDVISC), offset 0x44C

OTG

This 32-bit register specifies the masked interrupt status of the ID valid detect. It also provides a method to clear the interrupt state.

USB ID Valid Detect Interrupt Status and Clear (USBIDVISC)

Base 0x4005.0000
 Offset 0x44C
 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 | | | | | | | | | | | | | | | ID |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:1 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 0 | ID | R/W1C | 0 | ID Valid Detect Interrupt Status and Clear |

| Value | Description |
|-------|---|
| 1 | The ID bits in the USBIDVRIS and USBIDVIM registers are set, providing an interrupt to the interrupt controller. |
| 0 | No interrupt has occurred or the interrupt is masked. |

This bit is cleared by writing a 1. Clearing this bit also clears the **ID** bit in the **USBIDVRIS** register.

Register 332: USB DMA Select (USBDMASEL), offset 0x450

OTG A /
Host

This 32-bit register specifies which endpoints are mapped to the 6 allocated μ DMA channels, see Table 9-1 on page 260 for more information on channel assignments.

OTG B /
Device

USB DMA Select (USBDMASEL)

Base 0x4005.0000
Offset 0x450
Type R/W, reset 0x0033.2211

| | | | | | | | | | | | | | | | | |
|-------|----------|-----|-----|-----|--------|-----|-----|-----|--------|-----|-----|-----|--------|-----|-----|-----|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | reserved | | | | | | | | DMACTX | | | | DMACRX | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DMABTX | | | | DMABRX | | | | DMAATX | | | | DMAARX | | | |
| 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 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

| 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. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 23:20 | DMACTX | R/W | 0x3 | DMA C TX Select Specifies the TX mapping of the third USB endpoint on μ DMA channel 5 (primary assignment). <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>reserved</td></tr> <tr><td>0x1</td><td>Endpoint 1 TX</td></tr> <tr><td>0x2</td><td>Endpoint 2 TX</td></tr> <tr><td>0x3</td><td>Endpoint 3 TX</td></tr> <tr><td>0x4</td><td>Endpoint 4 TX</td></tr> <tr><td>0x5</td><td>Endpoint 5 TX</td></tr> <tr><td>0x6</td><td>Endpoint 6 TX</td></tr> <tr><td>0x7</td><td>Endpoint 7 TX</td></tr> <tr><td>0x8</td><td>Endpoint 8 TX</td></tr> <tr><td>0x9</td><td>Endpoint 9 TX</td></tr> <tr><td>0xA</td><td>Endpoint 10 TX</td></tr> <tr><td>0xB</td><td>Endpoint 11 TX</td></tr> <tr><td>0xC</td><td>Endpoint 12 TX</td></tr> <tr><td>0xD</td><td>Endpoint 13 TX</td></tr> <tr><td>0xE</td><td>Endpoint 14 TX</td></tr> <tr><td>0xF</td><td>Endpoint 15 TX</td></tr> </tbody> </table> | Value | Description | 0x0 | reserved | 0x1 | Endpoint 1 TX | 0x2 | Endpoint 2 TX | 0x3 | Endpoint 3 TX | 0x4 | Endpoint 4 TX | 0x5 | Endpoint 5 TX | 0x6 | Endpoint 6 TX | 0x7 | Endpoint 7 TX | 0x8 | Endpoint 8 TX | 0x9 | Endpoint 9 TX | 0xA | Endpoint 10 TX | 0xB | Endpoint 11 TX | 0xC | Endpoint 12 TX | 0xD | Endpoint 13 TX | 0xE | Endpoint 14 TX | 0xF | Endpoint 15 TX |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | Endpoint 1 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | Endpoint 2 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | Endpoint 3 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | Endpoint 4 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | Endpoint 5 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | Endpoint 6 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Endpoint 7 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | Endpoint 8 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9 | Endpoint 9 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA | Endpoint 10 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB | Endpoint 11 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xC | Endpoint 12 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD | Endpoint 13 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xE | Endpoint 14 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF | Endpoint 15 TX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|----------------|------|-------|--|-------|-------------|-----|----------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|---------------|-----|----------------|-----|----------------|-----|----------------|-----|----------------|-----|----------------|-----|----------------|
| 19:16 | DMACRX | R/W | 0x3 | <p>DMA C RX Select</p> <p>Specifies the RX and TX mapping of the third USB endpoint on μDMA channel 4 (primary assignment).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>reserved</td></tr> <tr><td>0x1</td><td>Endpoint 1 RX</td></tr> <tr><td>0x2</td><td>Endpoint 2 RX</td></tr> <tr><td>0x3</td><td>Endpoint 3 RX</td></tr> <tr><td>0x4</td><td>Endpoint 4 RX</td></tr> <tr><td>0x5</td><td>Endpoint 5 RX</td></tr> <tr><td>0x6</td><td>Endpoint 6 RX</td></tr> <tr><td>0x7</td><td>Endpoint 7 RX</td></tr> <tr><td>0x8</td><td>Endpoint 8 RX</td></tr> <tr><td>0x9</td><td>Endpoint 9 RX</td></tr> <tr><td>0xA</td><td>Endpoint 10 RX</td></tr> <tr><td>0xB</td><td>Endpoint 11 RX</td></tr> <tr><td>0xC</td><td>Endpoint 12 RX</td></tr> <tr><td>0xD</td><td>Endpoint 13 RX</td></tr> <tr><td>0xE</td><td>Endpoint 14 RX</td></tr> <tr><td>0xF</td><td>Endpoint 15 RX</td></tr> </tbody> </table> | Value | Description | 0x0 | reserved | 0x1 | Endpoint 1 RX | 0x2 | Endpoint 2 RX | 0x3 | Endpoint 3 RX | 0x4 | Endpoint 4 RX | 0x5 | Endpoint 5 RX | 0x6 | Endpoint 6 RX | 0x7 | Endpoint 7 RX | 0x8 | Endpoint 8 RX | 0x9 | Endpoint 9 RX | 0xA | Endpoint 10 RX | 0xB | Endpoint 11 RX | 0xC | Endpoint 12 RX | 0xD | Endpoint 13 RX | 0xE | Endpoint 14 RX | 0xF | Endpoint 15 RX |
| Value | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0 | reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1 | Endpoint 1 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x2 | Endpoint 2 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x3 | Endpoint 3 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x4 | Endpoint 4 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x5 | Endpoint 5 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x6 | Endpoint 6 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x7 | Endpoint 7 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x8 | Endpoint 8 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x9 | Endpoint 9 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xA | Endpoint 10 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xB | Endpoint 11 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xC | Endpoint 12 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xD | Endpoint 13 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xE | Endpoint 14 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0xF | Endpoint 15 RX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15:12 | DMABTX | R/W | 0x2 | <p>DMA B TX Select</p> <p>Specifies the TX mapping of the second USB endpoint on μDMA channel 3 (primary assignment).</p> <p>Same bit definitions as the <code>DMACTX</code> field.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11:8 | DMABRX | R/W | 0x2 | <p>DMA B RX Select</p> <p>Specifies the RX mapping of the second USB endpoint on μDMA channel 2 (primary assignment).</p> <p>Same bit definitions as the <code>DMACRX</code> field.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7:4 | DMAATX | R/W | 0x1 | <p>DMA A TX Select</p> <p>Specifies the TX mapping of the first USB endpoint on μDMA channel 1 (primary assignment).</p> <p>Same bit definitions as the <code>DMACTX</code> field.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3:0 | DMAARX | R/W | 0x1 | <p>DMA A RX Select</p> <p>Specifies the RX mapping of the first USB endpoint on μDMA channel 0 (primary assignment).</p> <p>Same bit definitions as the <code>DMACRX</code> field.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

22 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.

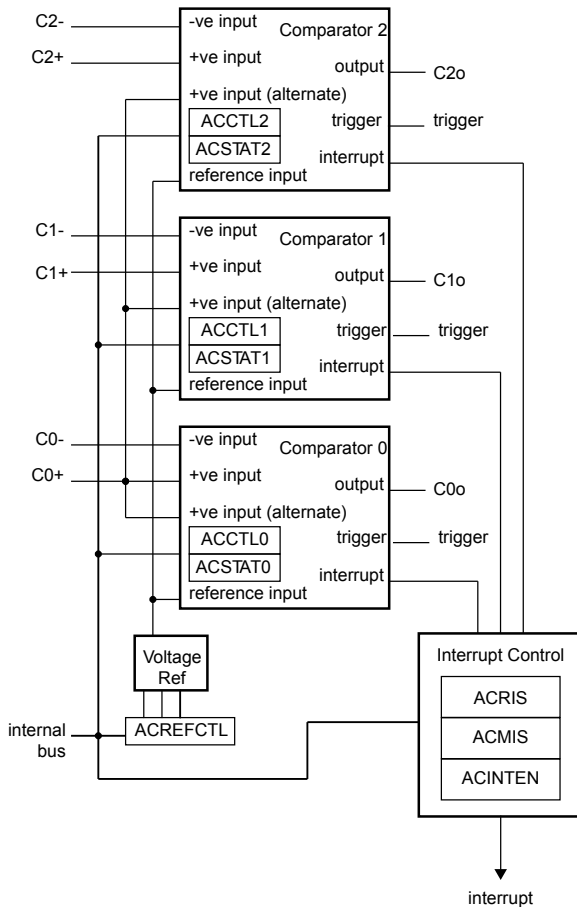
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board. In addition, the comparator can signal the application via interrupts or trigger the start of a sample sequence in the ADC. The interrupt generation and ADC triggering logic is separate and independent. This flexibility means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The Stellaris® LM3S9B90 microcontroller provides three independent integrated analog comparators with the following functions:

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

22.1 Block Diagram

Figure 22-1. Analog Comparator Module Block Diagram



22.2 Signal Description

Table 22-1 on page 996 lists the external signals of the Analog Comparators and describes the function of each. The Analog Comparator output signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 340) should be set to choose the Analog Comparator function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOCTL)** register (page 358) to assign the Analog Comparator signal to the specified GPIO port pin. The positive and negative input signals are configured by clearing the `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 318.

Table 22-1. Signals for Analog Comparators

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|-------------------------------------|
| C0+ | 90 | PB6 | I | Analog | Analog comparator 0 positive input. |
| C0- | 92 | PB4 | I | Analog | Analog comparator 0 negative input. |

Table 22-1. Signals for Analog Comparators (*continued*)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|-----------------------------|---|----------|--------------------------|-------------------------------------|
| C0o | 24 42 90 91 100 | PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2) | O | TTL | Analog comparator 0 output. |
| C1+ | 24 | PC5 | I | Analog | Analog comparator 1 positive input. |
| C1- | 91 | PB5 | I | Analog | Analog comparator 1 negative input. |
| C1o | 2 22 24 41 84 | PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2) | O | TTL | Analog comparator 1 output. |
| C2+ | 23 | PC6 | I | Analog | Analog comparator 2 positive input. |
| C2- | 22 | PC7 | I | Analog | Analog comparator 2 negative input. |
| C2o | 1 23 | PE7 (2) PC6 (3) | O | TTL | Analog comparator 2 output. |

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

22.3 Functional Description

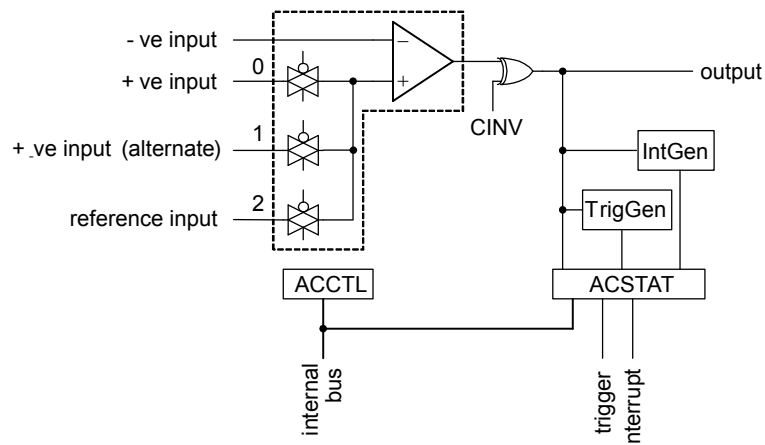
The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

$$VIN- < VIN+, VOUT = 1$$

$$VIN- > VIN+, VOUT = 0$$

As shown in Figure 22-2 on page 997, the input source for VIN- is an external input, C_{n-}. In addition to an external input, C_{n+}, input sources for VIN+ can be the C0+ or an internal reference, V_{IREF}.

Figure 22-2. Structure of Comparator Unit



A comparator is configured through two status/control registers, **Analog Comparator Control (ACCTL)** and **Analog Comparator Status (ACSTAT)**. The internal reference is configured through one control register, **Analog Comparator Reference Voltage Control (ACREFCTL)**. Interrupt status and control are configured through three registers, **Analog Comparator Masked Interrupt**

Status (ACMIS), Analog Comparator Raw Interrupt Status (ACRIS), and Analog Comparator Interrupt Enable (ACINTEN).

Typically, the comparator output is used internally to generate an interrupt as controlled by the I_{SEN} bit in the **ACCTL** register. The output may also be used to drive an external pin, Co or generate an analog-to-digital converter (ADC) trigger.

Important: The $ASRCP$ bits in the **ACCTL** register must be set before using the analog comparators.

22.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 22-3 on page 998. The internal reference is controlled by a single configuration register (**ACREFCTL**). Table 22-2 on page 998 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally (V_{IREF}).

Figure 22-3. Comparator Internal Reference Structure

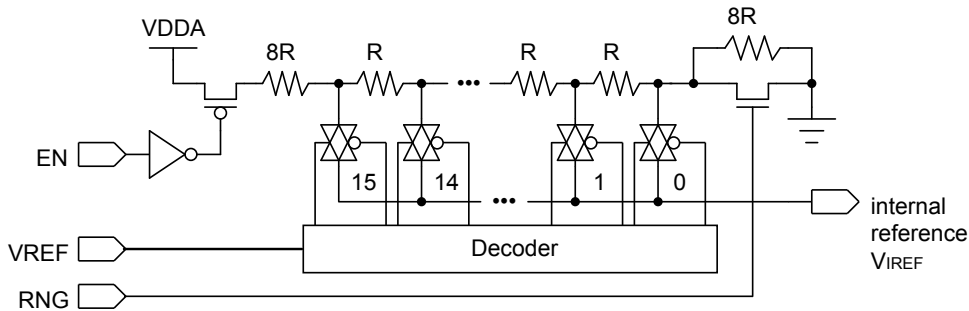


Table 22-2. Internal Reference Voltage and ACREFCTL Field Values

| ACREFCTL Register | | Output Reference Voltage Based on VREF Field Value |
|-------------------|---------------|---|
| EN Bit Value | RNG Bit Value | |
| EN=0 | RNG=X | 0 V (GND) for any value of VREF; however, it is recommended that RNG=1 and VREF=0 for the least noisy ground reference. |

Table 22-2. Internal Reference Voltage and ACREFCTL Field Values (continued)

| ACREFCTL Register | | Output Reference Voltage Based on VREF Field Value |
|-------------------|---------------|--|
| EN Bit Value | RNG Bit Value | |
| EN=1 | RNG=0 | <p>Total resistance in ladder is 31 R.</p> $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{(V_{REF} + 8)}{31}$ $V_{IREF} = 0.85 + 0.106 \times V_{REF}$ <p>The range of internal reference in this mode is 0.85-2.448 V.</p> |
| | RNG=1 | <p>Total resistance in ladder is 23 R.</p> $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{V_{REF}}{23}$ $V_{IREF} = 0.143 \times V_{REF}$ <p>The range of internal reference for this mode is 0-2.152 V.</p> |

22.4 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 (see page 175).
2. In the GPIO module, enable the GPIO port/pin associated with the input signals as GPIO inputs. To determine which GPIO to configure, see Table 24-4 on page 1028.
3. Configure the **PMCn** fields in the **GPIOCTL** register to assign the analog comparator output signals to the appropriate pins (see page 358 and Table 24-5 on page 1035).
4. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.

5. Configure the comparator to use the internal voltage reference and to *not* invert the output by writing the **ACCTLn** register with the value of 0x0000.040C.
6. Delay for 10 μ s.
7. Read the comparator output value by reading the **ACSTATn** register's **OVAL** value.

Change the level of the comparator negative input signal **C-** to see the **OVAL** value change.

22.5 Register Map

Table 22-3 on page 1000 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000. Note that the analog comparator clock must be enabled before the registers can be programmed (see page 175).

Table 22-3. Analog Comparators Register Map

| Offset | Name | Type | Reset | Description | See page |
|--------|----------|-------|-------------|---|----------|
| 0x000 | ACMIS | R/W1C | 0x0000.0000 | Analog Comparator Masked Interrupt Status | 1001 |
| 0x004 | ACRIS | RO | 0x0000.0000 | Analog Comparator Raw Interrupt Status | 1002 |
| 0x008 | ACINTEN | R/W | 0x0000.0000 | Analog Comparator Interrupt Enable | 1003 |
| 0x010 | ACREFCTL | R/W | 0x0000.0000 | Analog Comparator Reference Voltage Control | 1004 |
| 0x020 | ACSTAT0 | RO | 0x0000.0000 | Analog Comparator Status 0 | 1005 |
| 0x024 | ACCTL0 | R/W | 0x0000.0000 | Analog Comparator Control 0 | 1006 |
| 0x040 | ACSTAT1 | RO | 0x0000.0000 | Analog Comparator Status 1 | 1005 |
| 0x044 | ACCTL1 | R/W | 0x0000.0000 | Analog Comparator Control 1 | 1006 |
| 0x060 | ACSTAT2 | RO | 0x0000.0000 | Analog Comparator Status 2 | 1005 |
| 0x064 | ACCTL2 | R/W | 0x0000.0000 | Analog Comparator Control 2 | 1006 |

22.6 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

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-------|-------|-------|
| | 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 | | | | | | | | | | | | | IN2 | IN1 | IN0 |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | R/W1C | R/W1C | R/W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|-------|------------|---|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | IN2 | R/W1C | 0 | <p>Comparator 2 Masked Interrupt Status</p> <p>Value Description</p> <p>1 The IN2 bits in the ACRIS register and the ACINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the IN2 bit in the ACRIS register.</p> |
| 1 | IN1 | R/W1C | 0 | <p>Comparator 1 Masked Interrupt Status</p> <p>Value Description</p> <p>1 The IN1 bits in the ACRIS register and the ACINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the IN1 bit in the ACRIS register.</p> |
| 0 | IN0 | R/W1C | 0 | <p>Comparator 0 Masked Interrupt Status</p> <p>Value Description</p> <p>1 The IN0 bits in the ACRIS register and the ACINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the IN0 bit in the ACRIS register.</p> |

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparator. The bits in this register must be enabled to generate interrupts using the **ACINTEN** register.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
 Offset 0x004
 Type RO, reset 0x0000.0000

| | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|
| | 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 | | | | | | | | | | | | | IN2 | IN1 | IN0 |
| 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 |
|-----------|----------|------|------------|--|
| 31:3 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 2 | IN2 | RO | 0 | Comparator 2 Interrupt Status Value Description 1 Comparator 2 has generated an interrupt for an event as configured by the ISEN bit in the ACCTL2 register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN2 bit in the ACMIS register. |
| 1 | IN1 | RO | 0 | Comparator 1 Interrupt Status Value Description 1 Comparator 1 has generated an interrupt for an event as configured by the ISEN bit in the ACCTL1 register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN1 bit in the ACMIS register. |
| 0 | IN0 | RO | 0 | Comparator 0 Interrupt Status Value Description 1 Comparator 0 has generated an interrupt for an event as configured by the ISEN bit in the ACCTL0 register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN0 bit in the ACMIS register. |

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparators.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000

Offset 0x008

Type 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 | | | | | | | | | | | | | IN2 | IN1 | IN0 | |
| Type | RO | 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 | 0 |

| 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 Value Description 1 The raw interrupt signal comparator 2 is sent to the interrupt controller. 0 A comparator 2 interrupt does not affect the interrupt status. |
| 1 | IN1 | R/W | 0 | Comparator 1 Interrupt Enable Value Description 1 The raw interrupt signal comparator 1 is sent to the interrupt controller. 0 A comparator 1 interrupt does not affect the interrupt status. |
| 0 | IN0 | R/W | 0 | Comparator 0 Interrupt Enable Value Description 1 The raw interrupt signal comparator 0 is sent to the interrupt controller. 0 A comparator 0 interrupt does not affect the interrupt status. |

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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|-----|-----|----------|----|----|----|-----|------|-----|-----|--|
| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| | reserved | | | | | | | | | | | | | | | | |
| Type | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | reserved | | | | | | EN | RNG | reserved | | | | | VREF | | | |
| Type | RO | RO | RO | RO | RO | RO | 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 | |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|----------|---|
| 31:10 | reserved | RO | 0x0000.0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 9 | EN | R/W | 0 | Resistor Ladder Enable Value Description 0 The resistor ladder is unpowered. 1 Powers on the resistor ladder. The resistor ladder is connected to V_{DDA} . This bit is cleared at reset so that the internal reference consumes the least amount of power if it is not used. |
| 8 | RNG | R/W | 0 | Resistor Ladder Range Value Description 0 The resistor ladder has a total resistance of 31 R. 1 The resistor ladder has a total resistance of 23 R. |
| 7:4 | reserved | RO | 0x0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 3:0 | VREF | R/W | 0x0 | Resistor Ladder Voltage Ref The V_{REF} 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 22-2 on page 998 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

| | | | | | | | | | | | | | | | | | |
|-------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------|----------|
| | 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 | | | | | | | | | | | | | | | OVAL | reserved |
| Type | RO | 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 | 0 |

| Bit/Field | Name | Type | Reset | Description |
|-----------|----------|------|------------|--|
| 31:2 | reserved | RO | 0x0000.000 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |
| 1 | OVAL | RO | 0 | Comparator Output Value Value Description 0 VIN- > VIN+ 1 VIN- < VIN+ VIN- is the voltage on the C _{n-} pin. VIN+ is the voltage on the C _{n+} pin, the C0+ pin, or the internal voltage reference (V _{IREF}) as defined by the ASRCP bit in the ACCTL register. |
| 0 | reserved | RO | 0 | Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. |

Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x024

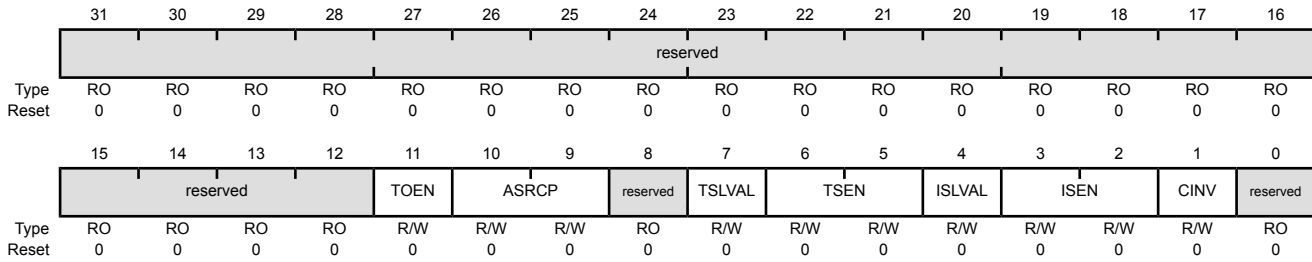
Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044

Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000
 Offset 0x024
 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 | TOEN | R/W | 0 | Trigger Output Enable Value Description 0 ADC events are suppressed and not sent to the ADC. 1 ADC events are sent to the ADC. |
| 10:9 | ASRCP | R/W | 0x0 | 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 Description 0x0 Pin value of Cn+ 0x1 Pin value of C0+ 0x2 Internal voltage reference (V _{IREF}) 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 Value Description 0 An ADC event is generated if the comparator output is Low. 1 An ADC event is generated if the comparator output is High. |

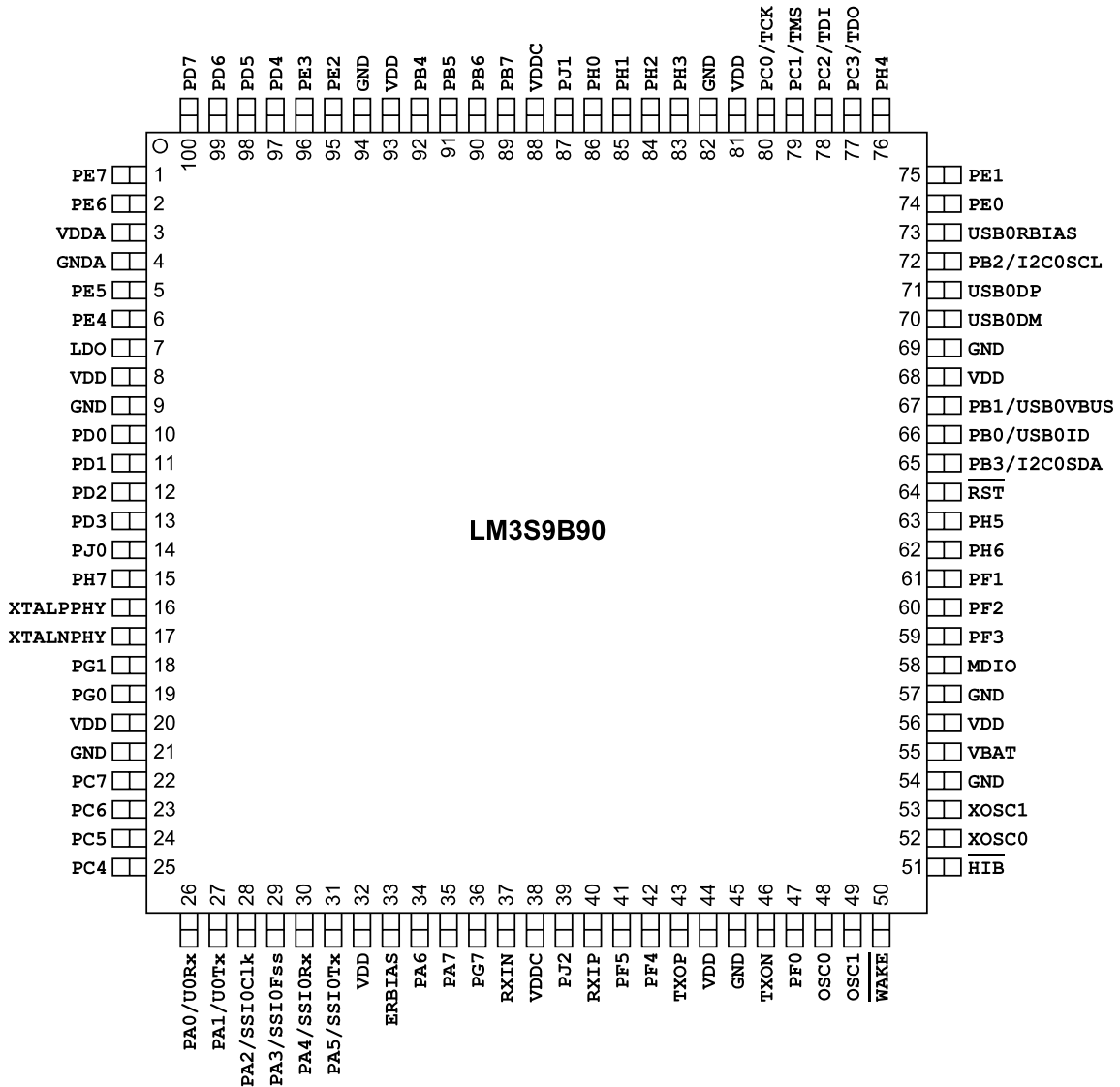
| Bit/Field | Name | Type | Reset | Description | | | | | | | | | | |
|-----------|--|------|-------|--|-------|-------------|-----|--|-----|--|-----|-------------|-----|-------------|
| 6:5 | TSEN | R/W | 0x0 | <p>Trigger Sense</p> <p>The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Level sense, see TSLVAL</td> </tr> <tr> <td>0x1</td> <td>Falling edge</td> </tr> <tr> <td>0x2</td> <td>Rising edge</td> </tr> <tr> <td>0x3</td> <td>Either edge</td> </tr> </tbody> </table> | Value | Description | 0x0 | Level sense, see TSLVAL | 0x1 | Falling edge | 0x2 | Rising edge | 0x3 | Either edge |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Level sense, see TSLVAL | | | | | | | | | | | | | |
| 0x1 | Falling edge | | | | | | | | | | | | | |
| 0x2 | Rising edge | | | | | | | | | | | | | |
| 0x3 | Either edge | | | | | | | | | | | | | |
| 4 | ISLVAL | R/W | 0 | <p>Interrupt Sense Level Value</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>An interrupt is generated if the comparator output is Low.</td> </tr> <tr> <td>1</td> <td>An interrupt is generated if the comparator output is High.</td> </tr> </tbody> </table> | Value | Description | 0 | An interrupt is generated if the comparator output is Low. | 1 | An interrupt is generated if the comparator output is High. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | An interrupt is generated if the comparator output is Low. | | | | | | | | | | | | | |
| 1 | An interrupt is generated if the comparator output is High. | | | | | | | | | | | | | |
| 3:2 | ISEN | R/W | 0x0 | <p>Interrupt Sense</p> <p>The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Level sense, see ISLVAL</td> </tr> <tr> <td>0x1</td> <td>Falling edge</td> </tr> <tr> <td>0x2</td> <td>Rising edge</td> </tr> <tr> <td>0x3</td> <td>Either edge</td> </tr> </tbody> </table> | Value | Description | 0x0 | Level sense, see ISLVAL | 0x1 | Falling edge | 0x2 | Rising edge | 0x3 | Either edge |
| Value | Description | | | | | | | | | | | | | |
| 0x0 | Level sense, see ISLVAL | | | | | | | | | | | | | |
| 0x1 | Falling edge | | | | | | | | | | | | | |
| 0x2 | Rising edge | | | | | | | | | | | | | |
| 0x3 | Either edge | | | | | | | | | | | | | |
| 1 | CINV | R/W | 0 | <p>Comparator Output Invert</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output of the comparator is unchanged.</td> </tr> <tr> <td>1</td> <td>The output of the comparator is inverted prior to being processed by hardware.</td> </tr> </tbody> </table> | Value | Description | 0 | The output of the comparator is unchanged. | 1 | The output of the comparator is inverted prior to being processed by hardware. | | | | |
| Value | Description | | | | | | | | | | | | | |
| 0 | The output of the comparator is unchanged. | | | | | | | | | | | | | |
| 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. | | | | | | | | | | |

23 Pin Diagram

The LM3S9B90 microcontroller pin diagram is shown below.

Each GPIO signal is identified by its GPIO port unless it defaults to an alternate function on reset. In this case, the GPIO port name is followed by the default alternate function. To see a complete list of possible functions for each pin, see Table 24-5 on page 1035.

Figure 23-1. 100-Pin LQFP Package Pin Diagram



24 Signal Tables

The following tables list the signals available for each pin. Signals are configured as GPIOs on reset, except for those noted below. Use the **GPIOAMSEL** register (see page 356) to select analog mode. For a GPIO pin to be used for an alternate digital function, the corresponding bit in the **GPIOAFSEL** register (see page 340) must be set. Further pin muxing options are provided through the PMC_x bit field in the **GPIOPCTL** register (see page 358), which selects one of several available peripheral functions for that GPIO.

Important: All GPIO pins are configured as GPIOs by default with the exception of the pins shown in the table below. A Power-On-Reset (\overline{POR}) or asserting \overline{RST} puts the pins back to their default state.

Table 24-1. GPIO Pins With Default Alternate Functions

| GPIO Pin | Default State | GPIOAFSEL Bit | GPIOPCTL PMC_x Bit Field |
|----------|-------------------|---------------|----------------------------|
| PA[1:0] | UART0 | 1 | 0x1 |
| PA[5:2] | SSI0 | 1 | 0x1 |
| PB[3:2] | I ² C0 | 1 | 0x1 |
| PC[3:0] | JTAG/SWD | 1 | 0x3 |

Table 24-2 on page 1009 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Each possible alternate analog and digital function is listed for each pin.

Table 24-3 on page 1019 lists the signals in alphabetical order by signal name. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed. The "Pin Mux" column indicates the GPIO and the encoding needed in the PMC_x bit field in the **GPIOPCTL** register.

Table 24-4 on page 1028 groups the signals by functionality, except for GPIOs. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed.

Table 24-5 on page 1035 lists the GPIO pins and their analog and digital alternate functions. The AIN_x and $VREF_A$ analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding $AMSEL$ bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry ($C0-$, $C0+$, $C1-$, $C1+$, $C2-$, $C2+$, $USB0VBUS$, $USB0ID$). These signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. The digital signals are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the PMC_x bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric encoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Table 24-2. Signals by Pin Number

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|----------|----------|--------------------------|--|
| 1 | PE7 | I/O | TTL | GPIO port E bit 7. |
| | AIN0 | I | Analog | Analog-to-digital converter input 0. |
| | C2o | O | TTL | Analog comparator 2 output. |
| | U1DCD | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|-----------------|----------|--------------------------|---|
| 2 | PE6 | I/O | TTL | GPIO port E bit 6. |
| | AIN1 | I | Analog | Analog-to-digital converter input 1. |
| | Cl _o | O | TTL | Analog comparator 1 output. |
| | U1CTS | I | TTL | UART module 1 Clear To Send modem status input signal. |
| 3 | VDDA | - | Power | The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. |
| 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 | PE5 | I/O | TTL | GPIO port E bit 5. |
| | AIN2 | I | Analog | Analog-to-digital converter input 2. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | I2S0TXSD | I/O | TTL | I ² S module 0 transmit data. |
| 6 | PE4 | I/O | TTL | GPIO port E bit 4. |
| | AIN3 | I | Analog | Analog-to-digital converter input 3. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | I2S0TXWS | I/O | TTL | I ² S module 0 transmit word select. |
| 7 | LDO | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 μF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| 8 | VDD | - | Power | Positive supply for I/O and some logic. |
| 9 | GND | - | Power | Ground reference for logic and I/O pins. |
| 10 | PD0 | I/O | TTL | GPIO port D bit 0. |
| | AIN15 | I | Analog | Analog-to-digital converter input 15. |
| | CAN0Rx | I | TTL | CAN module 0 receive. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | I2S0RXSCK | I/O | TTL | I ² S module 0 receive clock. |
| | U1CTS | I | TTL | UART module 1 Clear To Send modem status input signal. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|----------|----------|--------------------------|---|
| 11 | PD1 | I/O | TTL | GPIO port D bit 1. |
| | AIN14 | I | Analog | Analog-to-digital converter input 14. |
| | CAN0Tx | O | TTL | CAN module 0 transmit. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | I2S0RXWS | I/O | TTL | I ² S module 0 receive word select. |
| | U1DCD | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| 12 | PD2 | I/O | TTL | GPIO port D bit 2. |
| | AIN13 | I | Analog | Analog-to-digital converter input 13. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | EPI0S20 | I/O | TTL | EPI module 0 signal 20. |
| 13 | PD3 | I/O | TTL | GPIO port D bit 3. |
| | AIN12 | I | Analog | Analog-to-digital converter input 12. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | EPI0S21 | I/O | TTL | EPI module 0 signal 21. |
| 14 | PJ0 | I/O | TTL | GPIO port J bit 0. |
| | EPI0S16 | I/O | TTL | EPI module 0 signal 16. |
| | I2C1SCL | I/O | OD | I ² C module 1 clock. |
| 15 | PH7 | I/O | TTL | GPIO port H bit 7. |
| | EPI0S27 | I/O | TTL | EPI module 0 signal 27. |
| | SSI1Tx | O | TTL | SSI module 1 transmit. |
| 16 | XTALPPHY | O | Analog | XTALP of the Ethernet PHY. |
| 17 | XTALNPHY | I | Analog | XTALN of the Ethernet PHY. |
| 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. |
| | I2C1SDA | I/O | OD | I ² C module 1 data. |
| | EPI0S14 | I/O | TTL | EPI module 0 signal 14. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|----------|----------|--------------------------|--|
| 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. |
| | I2C1SCL | I/O | OD | I ² C module 1 clock. |
| | USB0EPEN | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |
| | EPI0S13 | I/O | TTL | EPI module 0 signal 13. |
| 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. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | USB0PFLT | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| | C1o | O | TTL | Analog comparator 1 output. |
| | EPI0S5 | I/O | TTL | EPI module 0 signal 5. |
| 23 | PC6 | I/O | TTL | GPIO port C bit 6. |
| | C2+ | I | Analog | Analog comparator 2 positive input. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | C2o | O | TTL | Analog comparator 2 output. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | USB0PFLT | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| | EPI0S4 | I/O | TTL | EPI module 0 signal 4. |
| 24 | PC5 | I/O | TTL | GPIO port C bit 5. |
| | C1+ | I | Analog | Analog comparator 1 positive input. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | C1o | O | TTL | Analog comparator 1 output. |
| | C0o | O | TTL | Analog comparator 0 output. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | USB0EPEN | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |
| | EPI0S3 | I/O | TTL | EPI module 0 signal 3. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|------------|----------|--------------------------|---|
| 25 | PC4 | I/O | TTL | GPIO port C bit 4. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | EPI0S2 | I/O | TTL | EPI module 0 signal 2. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| 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. |
| | I2C1SCL | I/O | OD | I ² C module 1 clock. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| 27 | PA1 | I/O | TTL | GPIO port A bit 1. |
| | U0Tx | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | I2C1SDA | I/O | OD | I ² C module 1 data. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| 28 | PA2 | I/O | TTL | GPIO port A bit 2. |
| | SSI0Clk | I/O | TTL | SSI module 0 clock. |
| | I2S0RXSD | I/O | TTL | I ² S module 0 receive data. |
| 29 | PA3 | I/O | TTL | GPIO port A bit 3. |
| | SSI0Fss | I/O | TTL | SSI module 0 frame. |
| | I2S0RXMCLK | I/O | TTL | I ² S module 0 receive master clock. |
| 30 | PA4 | I/O | TTL | GPIO port A bit 4. |
| | SSI0Rx | I | TTL | SSI module 0 receive. |
| | CAN0Rx | I | TTL | CAN module 0 receive. |
| | I2S0TXSCK | I/O | TTL | I ² S module 0 transmit clock. |
| 31 | PA5 | I/O | TTL | GPIO port A bit 5. |
| | SSI0Tx | O | TTL | SSI module 0 transmit. |
| | CAN0Tx | O | TTL | CAN module 0 transmit. |
| | I2S0TXWS | I/O | TTL | I ² S module 0 transmit word select. |
| 32 | VDD | - | Power | Positive supply for I/O and some logic. |
| 33 | ERBIAS | O | Analog | 12.4-kΩ resistor (1% precision) used internally for Ethernet PHY. |
| 34 | PA6 | I/O | TTL | GPIO port A bit 6. |
| | I2C1SCL | I/O | OD | I ² C module 1 clock. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CAN0Rx | I | TTL | CAN module 0 receive. |
| | USB0EPEN | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |
| | U1CTS | I | TTL | UART module 1 Clear To Send modem status input signal. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|----------|----------|--------------------------|--|
| 35 | PA7 | I/O | TTL | GPIO port A bit 7. |
| | I2C1SDA | I/O | OD | I ² C module 1 data. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | CAN0Tx | O | TTL | CAN module 0 transmit. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | USB0PFLT | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| | U1DCD | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| 36 | PG7 | I/O | TTL | GPIO port G bit 7. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | EPI0S31 | I/O | TTL | EPI module 0 signal 31. |
| 37 | RXIN | I | Analog | RXIN of the Ethernet PHY. |
| 38 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| 39 | PJ2 | I/O | TTL | GPIO port J bit 2. |
| | EPI0S18 | I/O | TTL | EPI module 0 signal 18. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| 40 | RXIP | I | Analog | RXIP of the Ethernet PHY. |
| 41 | PF5 | I/O | TTL | GPIO port F bit 5. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | C1o | O | TTL | Analog comparator 1 output. |
| | EPI0S15 | I/O | TTL | EPI module 0 signal 15. |
| | SSI1Tx | O | TTL | SSI module 1 transmit. |
| 42 | PF4 | I/O | TTL | GPIO port F bit 4. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | C0o | O | TTL | Analog comparator 0 output. |
| | EPI0S12 | I/O | TTL | EPI module 0 signal 12. |
| | SSI1Rx | I | TTL | SSI module 1 receive. |
| 43 | TXOP | O | TTL | 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 | TTL | TXON of the Ethernet PHY. |
| 47 | PF0 | I/O | TTL | GPIO port F bit 0. |
| | CAN1Rx | I | TTL | CAN module 1 receive. |
| | I2S0TXSD | I/O | TTL | I ² S module 0 transmit data. |
| | U1DSR | I | TTL | UART module 1 Data Set Ready modem output control line. |
| 48 | OSC0 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| 49 | OSC1 | O | Analog | Main oscillator crystal output. |
| 50 | WAKE | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| 51 | HIB | O | OD | An open-drain output that indicates the processor is in Hibernate mode. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|------------|----------|--------------------------|---|
| 52 | XOSC0 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register. |
| 53 | XOSC1 | O | Analog | Hibernation module oscillator crystal output. |
| 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 | OD | MDIO of the Ethernet PHY. |
| 59 | PF3 | I/O | TTL | GPIO port F bit 3. |
| | LED0 | O | TTL | MII LED 0. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame. |
| 60 | PF2 | I/O | TTL | GPIO port F bit 2. |
| | LED1 | O | TTL | MII LED 1. |
| | SSI1Clk | I/O | TTL | SSI module 1 clock. |
| 61 | PF1 | I/O | TTL | GPIO port F bit 1. |
| | CAN1Tx | O | TTL | CAN module 1 transmit. |
| | I2S0TXMCLK | I/O | TTL | I ² S module 0 transmit master clock. |
| | U1RTS | O | TTL | UART module 1 Request to Send modem output control line. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| 62 | PH6 | I/O | TTL | GPIO port H bit 6. |
| | EPI0S26 | I/O | TTL | EPI module 0 signal 26. |
| | SSI1Rx | I | TTL | SSI module 1 receive. |
| 63 | PH5 | I/O | TTL | GPIO port H bit 5. |
| | EPI0S11 | I/O | TTL | EPI module 0 signal 11. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame. |
| 64 | RST | I | TTL | System reset input. |
| 65 | PB3 | I/O | TTL | GPIO port B bit 3. |
| | I2C0SDA | I/O | OD | I ² C module 0 data. |
| | USB0PFLT | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| 66 | PB0 | I/O | TTL | GPIO port B bit 0. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | USB0ID | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is an A device and pulled up is a B device). |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|-----------|----------|--------------------------|--|
| 67 | PB1 | I/O | TTL | GPIO port B bit 1. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | USB0VBUS | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |
| 68 | VDD | - | Power | Positive supply for I/O and some logic. |
| 69 | GND | - | Power | Ground reference for logic and I/O pins. |
| 70 | USB0DM | I/O | Analog | Bidirectional differential data pin (D- per USB specification). |
| 71 | USB0DP | I/O | Analog | Bidirectional differential data pin (D+ per USB specification). |
| 72 | PB2 | I/O | TTL | GPIO port B bit 2. |
| | I2C0SCL | I/O | OD | I ² C module 0 clock. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | USB0EPEN | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |
| 73 | USB0RBIAS | O | Analog | 9.1-kΩ resistor (1% precision) used internally for USB analog circuitry. |
| 74 | PE0 | I/O | TTL | GPIO port E bit 0. |
| | SSI1C1k | I/O | TTL | SSI module 1 clock. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | EPI0S8 | I/O | TTL | EPI module 0 signal 8. |
| | USB0PFLT | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| 75 | PE1 | I/O | TTL | GPIO port E bit 1. |
| | SSI1Fss | I/O | TTL | SSI module 1 frame. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | EPI0S9 | I/O | TTL | EPI module 0 signal 9. |
| 76 | PH4 | I/O | TTL | GPIO port H bit 4. |
| | USB0PFLT | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| | EPI0S10 | I/O | TTL | EPI module 0 signal 10. |
| | SSI1C1k | I/O | TTL | SSI module 1 clock. |
| 77 | PC3 | I/O | TTL | GPIO port C bit 3. |
| | TDO | O | TTL | JTAG TDO and SWO. |
| | SWO | O | TTL | JTAG TDO and SWO. |
| 78 | PC2 | I/O | TTL | GPIO port C bit 2. |
| | TDI | I | TTL | JTAG TDI. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|-----------|----------|--------------------------|---|
| 79 | PC1 | I/O | TTL | GPIO port C bit 1. |
| | TMS | I | TTL | JTAG TMS and SWDIO. |
| | SWDIO | I/O | TTL | JTAG TMS and SWDIO. |
| 80 | PC0 | I/O | TTL | GPIO port C bit 0. |
| | TCK | I | TTL | JTAG/SWD CLK. |
| | SWCLK | I | TTL | JTAG/SWD CLK. |
| 81 | VDD | - | Power | Positive supply for I/O and some logic. |
| 82 | GND | - | Power | Ground reference for logic and I/O pins. |
| 83 | PH3 | I/O | TTL | GPIO port H bit 3. |
| | USB0EPEN | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |
| | EPI0S0 | I/O | TTL | EPI module 0 signal 0. |
| 84 | PH2 | I/O | TTL | GPIO port H bit 2. |
| | C1o | O | TTL | Analog comparator 1 output. |
| | EPI0S1 | I/O | TTL | EPI module 0 signal 1. |
| 85 | PH1 | I/O | TTL | GPIO port H bit 1. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | EPI0S7 | I/O | TTL | EPI module 0 signal 7. |
| 86 | PH0 | I/O | TTL | GPIO port H bit 0. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | EPI0S6 | I/O | TTL | EPI module 0 signal 6. |
| 87 | PJ1 | I/O | TTL | GPIO port J bit 1. |
| | EPI0S17 | I/O | TTL | EPI module 0 signal 17. |
| | USB0PFLT | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| | I2C1SDA | I/O | OD | I ² C module 1 data. |
| 88 | VDDC | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| 89 | PB7 | I/O | TTL | GPIO port B bit 7. |
| | NMI | I | TTL | Non-maskable interrupt. |
| 90 | PB6 | I/O | TTL | GPIO port B bit 6. |
| | VREFA | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AIN _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-2 on page 1039. |
| | C0+ | I | Analog | Analog comparator 0 positive input. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | C0o | O | TTL | Analog comparator 0 output. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | I2S0TXSCK | I/O | TTL | I ² S module 0 transmit clock. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|----------|----------|--------------------------|---|
| 91 | PB5 | I/O | TTL | GPIO port B bit 5. |
| | AIN11 | I | Analog | Analog-to-digital converter input 11. |
| | C1- | I | Analog | Analog comparator 1 negative input. |
| | C0o | O | TTL | Analog comparator 0 output. |
| | CCP5 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | I/O | TTL | Capture/Compare/PWM 6. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CAN0Tx | O | TTL | CAN module 0 transmit. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | U1Tx | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| EPI0S22 | I/O | TTL | EPI module 0 signal 22. | |
| 92 | PB4 | I/O | TTL | GPIO port B bit 4. |
| | AIN10 | I | Analog | Analog-to-digital converter input 10. |
| | C0- | I | Analog | Analog comparator 0 negative input. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| | CAN0Rx | I | TTL | CAN module 0 receive. |
| | U1Rx | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| EPI0S23 | I/O | TTL | EPI module 0 signal 23. | |
| 93 | VDD | - | Power | Positive supply for I/O and some logic. |
| 94 | GND | - | Power | Ground reference for logic and I/O pins. |
| 95 | PE2 | I/O | TTL | GPIO port E bit 2. |
| | AIN9 | I | Analog | Analog-to-digital converter input 9. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | SSI1Rx | I | TTL | SSI module 1 receive. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | EPI0S24 | I/O | TTL | EPI module 0 signal 24. |
| 96 | PE3 | I/O | TTL | GPIO port E bit 3. |
| | AIN8 | I | Analog | Analog-to-digital converter input 8. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | SSI1Tx | O | TTL | SSI module 1 transmit. |
| | CCP7 | I/O | TTL | Capture/Compare/PWM 7. |
| | EPI0S25 | I/O | TTL | EPI module 0 signal 25. |
| 97 | PD4 | I/O | TTL | GPIO port D bit 4. |
| | AIN7 | I | Analog | Analog-to-digital converter input 7. |
| | CCP0 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP3 | I/O | TTL | Capture/Compare/PWM 3. |
| | I2S0RXSD | I/O | TTL | I ² S module 0 receive data. |
| | U1RI | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| | EPI0S19 | I/O | TTL | EPI module 0 signal 19. |

Table 24-2. Signals by Pin Number (continued)

| Pin Number | Pin Name | Pin Type | Buffer Type ^a | Description |
|------------|------------|----------|--------------------------|---|
| 98 | PD5 | I/O | TTL | GPIO port D bit 5. |
| | AIN6 | I | Analog | Analog-to-digital converter input 6. |
| | CCP2 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP4 | I/O | TTL | Capture/Compare/PWM 4. |
| | I2S0RXMCLK | I/O | TTL | I ² S module 0 receive master clock. |
| | U2Rx | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| | EPI0S28 | I/O | TTL | EPI module 0 signal 28. |
| 99 | PD6 | I/O | TTL | GPIO port D bit 6. |
| | AIN5 | I | Analog | Analog-to-digital converter input 5. |
| | I2S0TXSCK | I/O | TTL | I ² S module 0 transmit clock. |
| | U2Tx | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | EPI0S29 | I/O | TTL | EPI module 0 signal 29. |
| 100 | PD7 | I/O | TTL | GPIO port D bit 7. |
| | AIN4 | I | Analog | Analog-to-digital converter input 4. |
| | CO0 | O | TTL | Analog comparator 0 output. |
| | CCP1 | I/O | TTL | Capture/Compare/PWM 1. |
| | I2S0TXWS | I/O | TTL | I ² S module 0 transmit word select. |
| | U1DTR | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| | EPI0S30 | I/O | TTL | EPI module 0 signal 30. |

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

Table 24-3. Signals by Signal Name

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|---------------------------------------|
| AIN0 | 1 | PE7 | I | Analog | Analog-to-digital converter input 0. |
| AIN1 | 2 | PE6 | I | Analog | Analog-to-digital converter input 1. |
| AIN2 | 5 | PE5 | I | Analog | Analog-to-digital converter input 2. |
| AIN3 | 6 | PE4 | I | Analog | Analog-to-digital converter input 3. |
| AIN4 | 100 | PD7 | I | Analog | Analog-to-digital converter input 4. |
| AIN5 | 99 | PD6 | I | Analog | Analog-to-digital converter input 5. |
| AIN6 | 98 | PD5 | I | Analog | Analog-to-digital converter input 6. |
| AIN7 | 97 | PD4 | I | Analog | Analog-to-digital converter input 7. |
| AIN8 | 96 | PE3 | I | Analog | Analog-to-digital converter input 8. |
| AIN9 | 95 | PE2 | I | Analog | Analog-to-digital converter input 9. |
| AIN10 | 92 | PB4 | I | Analog | Analog-to-digital converter input 10. |
| AIN11 | 91 | PB5 | I | Analog | Analog-to-digital converter input 11. |
| AIN12 | 13 | PD3 | I | Analog | Analog-to-digital converter input 12. |
| AIN13 | 12 | PD2 | I | Analog | Analog-to-digital converter input 13. |
| AIN14 | 11 | PD1 | I | Analog | Analog-to-digital converter input 14. |
| AIN15 | 10 | PD0 | I | Analog | Analog-to-digital converter input 15. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|--|---|----------|--------------------------|-------------------------------------|
| C0+ | 90 | PB6 | I | Analog | Analog comparator 0 positive input. |
| C0- | 92 | PB4 | I | Analog | Analog comparator 0 negative input. |
| C0o | 24 42 90 91 100 | PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2) | O | TTL | Analog comparator 0 output. |
| C1+ | 24 | PC5 | I | Analog | Analog comparator 1 positive input. |
| C1- | 91 | PB5 | I | Analog | Analog comparator 1 negative input. |
| C1o | 2 22 24 41 84 | PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2) | O | TTL | Analog comparator 1 output. |
| C2+ | 23 | PC6 | I | Analog | Analog comparator 2 positive input. |
| C2- | 22 | PC7 | I | Analog | Analog comparator 2 negative input. |
| C2o | 1 23 | PE7 (2) PC6 (3) | O | TTL | Analog comparator 2 output. |
| CAN0Rx | 10 30 34 92 | PD0 (2) PA4 (5) PA6 (6) PB4 (5) | I | TTL | CAN module 0 receive. |
| CAN0Tx | 11 31 35 91 | PD1 (2) PA5 (5) PA7 (6) PB5 (5) | O | TTL | CAN module 0 transmit. |
| CAN1Rx | 47 | PF0 (1) | I | TTL | CAN module 1 receive. |
| CAN1Tx | 61 | PF1 (1) | O | TTL | CAN module 1 transmit. |
| CCP0 | 13 22 23 39 42 66 72 91 97 | PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PB0 (1) PB2 (5) PB5 (4) PD4 (1) | I/O | TTL | Capture/Compare/PWM 0. |
| CCP1 | 24 25 34 67 90 96 100 | PC5 (1) PC4 (9) PA6 (2) PB1 (4) PB6 (1) PE3 (1) PD7 (3) | I/O | TTL | Capture/Compare/PWM 1. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|---|--|----------|--------------------------|-------------------------|
| CCP2 | 6 11 25 41 67 75 91 95 98 | PE4 (6) PD1 (10) PC4 (5) PF5 (1) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1) | I/O | TTL | Capture/Compare/PWM 2. |
| CCP3 | 6 23 24 35 61 72 74 97 | PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2) | I/O | TTL | Capture/Compare/PWM 3. |
| CCP4 | 22 25 35 95 98 | PC7 (1) PC4 (6) PA7 (2) PE2 (1) PD5 (2) | I/O | TTL | Capture/Compare/PWM 4. |
| CCP5 | 5 12 25 36 90 91 | PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2) | I/O | TTL | Capture/Compare/PWM 5. |
| CCP6 | 10 12 75 86 91 | PD0 (6) PD2 (2) PE1 (5) PH0 (1) PB5 (3) | I/O | TTL | Capture/Compare/PWM 6. |
| CCP7 | 11 13 85 90 96 | PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5) | I/O | TTL | Capture/Compare/PWM 7. |
| EPI0S0 | 83 | PH3 (8) | I/O | TTL | EPI module 0 signal 0. |
| EPI0S1 | 84 | PH2 (8) | I/O | TTL | EPI module 0 signal 1. |
| EPI0S2 | 25 | PC4 (8) | I/O | TTL | EPI module 0 signal 2. |
| EPI0S3 | 24 | PC5 (8) | I/O | TTL | EPI module 0 signal 3. |
| EPI0S4 | 23 | PC6 (8) | I/O | TTL | EPI module 0 signal 4. |
| EPI0S5 | 22 | PC7 (8) | I/O | TTL | EPI module 0 signal 5. |
| EPI0S6 | 86 | PH0 (8) | I/O | TTL | EPI module 0 signal 6. |
| EPI0S7 | 85 | PH1 (8) | I/O | TTL | EPI module 0 signal 7. |
| EPI0S8 | 74 | PE0 (8) | I/O | TTL | EPI module 0 signal 8. |
| EPI0S9 | 75 | PE1 (8) | I/O | TTL | EPI module 0 signal 9. |
| EPI0S10 | 76 | PH4 (8) | I/O | TTL | EPI module 0 signal 10. |
| EPI0S11 | 63 | PH5 (8) | I/O | TTL | EPI module 0 signal 11. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|---|---|----------|--------------------------|---|
| EPI0S12 | 42 | PF4 (8) | I/O | TTL | EPI module 0 signal 12. |
| EPI0S13 | 19 | PG0 (8) | I/O | TTL | EPI module 0 signal 13. |
| EPI0S14 | 18 | PG1 (8) | I/O | TTL | EPI module 0 signal 14. |
| EPI0S15 | 41 | PF5 (8) | I/O | TTL | EPI module 0 signal 15. |
| EPI0S16 | 14 | PJ0 (8) | I/O | TTL | EPI module 0 signal 16. |
| EPI0S17 | 87 | PJ1 (8) | I/O | TTL | EPI module 0 signal 17. |
| EPI0S18 | 39 | PJ2 (8) | I/O | TTL | EPI module 0 signal 18. |
| EPI0S19 | 97 | PD4 (10) | I/O | TTL | EPI module 0 signal 19. |
| EPI0S20 | 12 | PD2 (8) | I/O | TTL | EPI module 0 signal 20. |
| EPI0S21 | 13 | PD3 (8) | I/O | TTL | EPI module 0 signal 21. |
| EPI0S22 | 91 | PB5 (8) | I/O | TTL | EPI module 0 signal 22. |
| EPI0S23 | 92 | PB4 (8) | I/O | TTL | EPI module 0 signal 23. |
| EPI0S24 | 95 | PE2 (8) | I/O | TTL | EPI module 0 signal 24. |
| EPI0S25 | 96 | PE3 (8) | I/O | TTL | EPI module 0 signal 25. |
| EPI0S26 | 62 | PH6 (8) | I/O | TTL | EPI module 0 signal 26. |
| EPI0S27 | 15 | PH7 (8) | I/O | TTL | EPI module 0 signal 27. |
| EPI0S28 | 98 | PD5 (10) | I/O | TTL | EPI module 0 signal 28. |
| EPI0S29 | 99 | PD6 (10) | I/O | TTL | EPI module 0 signal 29. |
| EPI0S30 | 100 | PD7 (10) | I/O | TTL | EPI module 0 signal 30. |
| EPI0S31 | 36 | PG7 (9) | I/O | TTL | EPI module 0 signal 31. |
| ERBIAS | 33 | fixed | O | Analog | 12.4-kΩ resistor (1% precision) used internally for Ethernet PHY. |
| GND | 9 21 45 54 57 69 82 94 | fixed | - | Power | Ground reference for logic and I/O pins. |
| GND A | 4 | fixed | - | Power | The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions. |
| HIB | 51 | fixed | O | OD | An open-drain output that indicates the processor is in Hibernate mode. |
| I2C0SCL | 72 | PB2 (1) | I/O | OD | I ² C module 0 clock. |
| I2C0SDA | 65 | PB3 (1) | I/O | OD | I ² C module 0 data. |
| I2C1SCL | 14 19 26 34 | PJ0 (11) PG0 (3) PA0 (8) PA6 (1) | I/O | OD | I ² C module 1 clock. |
| I2C1SDA | 18 27 35 87 | PG1 (3) PA1 (8) PA7 (1) PJ1 (11) | I/O | OD | I ² C module 1 data. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|------------|----------------|-------------------------------|----------|--------------------------|--|
| I2S0RXMCLK | 29 98 | PA3 (9) PD5 (8) | I/O | TTL | I ² S module 0 receive master clock. |
| I2S0RXSCK | 10 | PD0 (8) | I/O | TTL | I ² S module 0 receive clock. |
| I2S0RXSD | 28 97 | PA2 (9) PD4 (8) | I/O | TTL | I ² S module 0 receive data. |
| I2S0RXWS | 11 | PD1 (8) | I/O | TTL | I ² S module 0 receive word select. |
| I2S0TXMCLK | 61 | PF1 (8) | I/O | TTL | I ² S module 0 transmit master clock. |
| I2S0TXSCK | 30 90 99 | PA4 (9) PB6 (9) PD6 (8) | I/O | TTL | I ² S module 0 transmit clock. |
| I2S0TXSD | 5 47 | PE5 (9) PF0 (8) | I/O | TTL | I ² S module 0 transmit data. |
| I2S0TXWS | 6 31 100 | PE4 (9) PA5 (9) PD7 (8) | I/O | TTL | I ² S module 0 transmit word select. |
| LDO | 7 | fixed | - | Power | Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 μ F or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| LED0 | 59 | PF3 (1) | O | TTL | MII LED 0. |
| LED1 | 60 | PF2 (1) | O | TTL | MII LED 1. |
| MDIO | 58 | fixed | I/O | OD | MDIO of the Ethernet PHY. |
| NMI | 89 | PB7 (4) | I | TTL | Non-maskable interrupt. |
| OSC0 | 48 | fixed | I | Analog | Main oscillator crystal input or an external clock reference input. |
| OSC1 | 49 | fixed | O | Analog | Main oscillator crystal output. |
| 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 | 72 | - | I/O | TTL | GPIO port B bit 2. |
| PB3 | 65 | - | 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. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|----------|------------|--------------------------|----------|--------------------------|--------------------|
| PC0 | 80 | - | I/O | TTL | GPIO port C bit 0. |
| PC1 | 79 | - | I/O | TTL | GPIO port C bit 1. |
| PC2 | 78 | - | I/O | TTL | GPIO port C bit 2. |
| PC3 | 77 | - | I/O | TTL | GPIO port C bit 3. |
| PC4 | 25 | - | I/O | TTL | GPIO port C bit 4. |
| PC5 | 24 | - | I/O | TTL | GPIO port C bit 5. |
| PC6 | 23 | - | I/O | TTL | GPIO port C bit 6. |
| PC7 | 22 | - | I/O | TTL | GPIO port C bit 7. |
| PD0 | 10 | - | I/O | TTL | GPIO port D bit 0. |
| PD1 | 11 | - | I/O | TTL | GPIO port D bit 1. |
| PD2 | 12 | - | I/O | TTL | GPIO port D bit 2. |
| PD3 | 13 | - | I/O | TTL | GPIO port D bit 3. |
| PD4 | 97 | - | I/O | TTL | GPIO port D bit 4. |
| PD5 | 98 | - | I/O | TTL | GPIO port D bit 5. |
| PD6 | 99 | - | I/O | TTL | GPIO port D bit 6. |
| PD7 | 100 | - | I/O | TTL | GPIO port D bit 7. |
| PE0 | 74 | - | I/O | TTL | GPIO port E bit 0. |
| PE1 | 75 | - | I/O | TTL | GPIO port E bit 1. |
| PE2 | 95 | - | I/O | TTL | GPIO port E bit 2. |
| PE3 | 96 | - | I/O | TTL | GPIO port E bit 3. |
| PE4 | 6 | - | I/O | TTL | GPIO port E bit 4. |
| PE5 | 5 | - | I/O | TTL | GPIO port E bit 5. |
| PE6 | 2 | - | I/O | TTL | GPIO port E bit 6. |
| PE7 | 1 | - | I/O | TTL | GPIO port E bit 7. |
| PF0 | 47 | - | I/O | TTL | GPIO port F bit 0. |
| PF1 | 61 | - | I/O | TTL | GPIO port F bit 1. |
| PF2 | 60 | - | I/O | TTL | GPIO port F bit 2. |
| PF3 | 59 | - | I/O | TTL | GPIO port F bit 3. |
| PF4 | 42 | - | I/O | TTL | GPIO port F bit 4. |
| PF5 | 41 | - | I/O | TTL | GPIO port F bit 5. |
| PG0 | 19 | - | I/O | TTL | GPIO port G bit 0. |
| PG1 | 18 | - | I/O | TTL | GPIO port G bit 1. |
| PG7 | 36 | - | I/O | TTL | GPIO port G bit 7. |
| PH0 | 86 | - | I/O | TTL | GPIO port H bit 0. |
| PH1 | 85 | - | I/O | TTL | GPIO port H bit 1. |
| PH2 | 84 | - | I/O | TTL | GPIO port H bit 2. |
| PH3 | 83 | - | I/O | TTL | GPIO port H bit 3. |
| PH4 | 76 | - | I/O | TTL | GPIO port H bit 4. |
| PH5 | 63 | - | I/O | TTL | GPIO port H bit 5. |
| PH6 | 62 | - | I/O | TTL | GPIO port H bit 6. |
| PH7 | 15 | - | I/O | TTL | GPIO port H bit 7. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|-------------------------|----------------|--------------------------------|----------|--------------------------|---|
| PJ0 | 14 | - | I/O | TTL | GPIO port J bit 0. |
| PJ1 | 87 | - | I/O | TTL | GPIO port J bit 1. |
| PJ2 | 39 | - | I/O | TTL | GPIO port J bit 2. |
| $\overline{\text{RST}}$ | 64 | fixed | I | TTL | System reset input. |
| RXIN | 37 | fixed | I | Analog | RXIN of the Ethernet PHY. |
| RXIP | 40 | fixed | I | Analog | RXIP of the Ethernet PHY. |
| SSI0Clk | 28 | PA2 (1) | I/O | TTL | SSI module 0 clock. |
| SSI0Fss | 29 | PA3 (1) | I/O | TTL | SSI module 0 frame. |
| SSI0Rx | 30 | PA4 (1) | I | TTL | SSI module 0 receive. |
| SSI0Tx | 31 | PA5 (1) | O | TTL | SSI module 0 transmit. |
| SSI1Clk | 60 74 76 | PF2 (9) PE0 (2) PH4 (11) | I/O | TTL | SSI module 1 clock. |
| SSI1Fss | 59 63 75 | PF3 (9) PH5 (11) PE1 (2) | I/O | TTL | SSI module 1 frame. |
| SSI1Rx | 42 62 95 | PF4 (9) PH6 (11) PE2 (2) | I | TTL | SSI module 1 receive. |
| SSI1Tx | 15 41 96 | PH7 (11) PF5 (9) PE3 (2) | O | TTL | SSI module 1 transmit. |
| SWCLK | 80 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| SWDIO | 79 | PC1 (3) | I/O | TTL | JTAG TMS and SWDIO. |
| SWO | 77 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TCK | 80 | PC0 (3) | I | TTL | JTAG/SWD CLK. |
| TDI | 78 | PC2 (3) | I | TTL | JTAG TDI. |
| TDO | 77 | PC3 (3) | O | TTL | JTAG TDO and SWO. |
| TMS | 79 | PC1 (3) | I | TTL | JTAG TMS and SWDIO. |
| TXON | 46 | fixed | O | TTL | TXON of the Ethernet PHY. |
| TXOP | 43 | fixed | O | TTL | TXOP of the Ethernet PHY. |
| U0Rx | 26 | PA0 (1) | I | TTL | UART module 0 receive. When in IrDA mode, this signal has IrDA modulation. |
| U0Tx | 27 | PA1 (1) | O | TTL | UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U1CTS | 2 10 34 | PE6 (9) PD0 (9) PA6 (9) | I | TTL | UART module 1 Clear To Send modem status input signal. |
| U1DCD | 1 11 35 | PE7 (9) PD1 (9) PA7 (9) | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| U1DSR | 47 | PF0 (9) | I | TTL | UART module 1 Data Set Ready modem output control line. |
| U1DTR | 100 | PD7 (9) | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|-----------|--|---|----------|--------------------------|---|
| U1RI | 97 | PD4 (9) | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| U1RTS | 61 | PF1 (9) | O | TTL | UART module 1 Request to Send modem output control line. |
| U1Rx | 10 12 23 26 66 92 | PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7) | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| U1Tx | 11 13 22 27 67 91 | PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7) | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| U2Rx | 10 19 92 98 | PD0 (4) PG0 (1) PB4 (4) PD5 (9) | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| U2Tx | 6 11 18 99 | PE4 (5) PD1 (4) PG1 (1) PD6 (9) | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |
| USB0DM | 70 | fixed | I/O | Analog | Bidirectional differential data pin (D- per USB specification). |
| USB0DP | 71 | fixed | I/O | Analog | Bidirectional differential data pin (D+ per USB specification). |
| USB0EPEN | 19 24 34 72 83 | PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4) | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |
| USB0ID | 66 | PB0 | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is an A device and pulled up is a B device). |
| USB0PFLT | 22 23 35 65 74 76 87 | PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9) | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| USB0RBIAS | 73 | fixed | O | Analog | 9.1-kΩ resistor (1% precision) used internally for USB analog circuitry. |
| USB0VBUS | 67 | PB1 | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |

Table 24-3. Signals by Signal Name (continued)

| Pin Name | Pin Number | Pin Mux / Pin Assignment | Pin Type | Buffer Type ^a | Description |
|--------------------------|---|--------------------------|----------|--------------------------|---|
| VBAT | 55 | fixed | - | Power | Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply. |
| VDD | 8 20 32 44 56 68 81 93 | fixed | - | Power | Positive supply for I/O and some logic. |
| VDDA | 3 | fixed | - | Power | The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. |
| VDDC | 38 88 | fixed | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| VREFA | 90 | PB6 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AIN _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-2 on page 1039. |
| $\overline{\text{WAKE}}$ | 50 | fixed | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| XOSC0 | 52 | fixed | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register. |
| XOSC1 | 53 | fixed | O | Analog | Hibernation module oscillator crystal output. |
| XTALNPHY | 17 | fixed | I | Analog | XTALN of the Ethernet PHY. |
| XTALPPHY | 16 | fixed | O | Analog | XTALP of the Ethernet PHY. |

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

Table 24-4. Signals by Function, Except for GPIO

| Function | Pin Name | Pin Number | Pin Type | Buffer Type ^a | Description |
|--------------------|----------|-----------------------------|----------|--------------------------|---|
| ADC | AIN0 | 1 | I | Analog | Analog-to-digital converter input 0. |
| | AIN1 | 2 | I | Analog | Analog-to-digital converter input 1. |
| | AIN2 | 5 | I | Analog | Analog-to-digital converter input 2. |
| | AIN3 | 6 | I | Analog | Analog-to-digital converter input 3. |
| | AIN4 | 100 | I | Analog | Analog-to-digital converter input 4. |
| | AIN5 | 99 | I | Analog | Analog-to-digital converter input 5. |
| | AIN6 | 98 | I | Analog | Analog-to-digital converter input 6. |
| | AIN7 | 97 | I | Analog | Analog-to-digital converter input 7. |
| | AIN8 | 96 | I | Analog | Analog-to-digital converter input 8. |
| | AIN9 | 95 | I | Analog | Analog-to-digital converter input 9. |
| | AIN10 | 92 | I | Analog | Analog-to-digital converter input 10. |
| | AIN11 | 91 | I | Analog | Analog-to-digital converter input 11. |
| | AIN12 | 13 | I | Analog | Analog-to-digital converter input 12. |
| | AIN13 | 12 | I | Analog | Analog-to-digital converter input 13. |
| | AIN14 | 11 | I | Analog | Analog-to-digital converter input 14. |
| | AIN15 | 10 | I | Analog | Analog-to-digital converter input 15. |
| | VREFA | 90 | I | Analog | This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AIN _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-2 on page 1039. |
| Analog Comparators | C0+ | 90 | I | Analog | Analog comparator 0 positive input. |
| | C0- | 92 | I | Analog | Analog comparator 0 negative input. |
| | C0o | 24 42 90 91 100 | 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 | 2 22 24 41 84 | 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 | 1 23 | O | TTL | Analog comparator 2 output. |

Table 24-4. Signals by Function, Except for GPIO (continued)

| Function | Pin Name | Pin Number | Pin Type | Buffer Type ^a | Description |
|-------------------------|----------|------------|----------|----------------------------|---|
| Controller Area Network | CAN0Rx | 10 | I | TTL | CAN module 0 receive. |
| | | 30 | | | |
| | | 34 | | | |
| | | 92 | | | |
| CAN0Tx | CAN0Tx | 11 | O | TTL | CAN module 0 transmit. |
| | | 31 | | | |
| | | 35 | | | |
| | | 91 | | | |
| CAN1Rx | 47 | I | TTL | CAN module 1 receive. | |
| CAN1Tx | 61 | O | TTL | CAN module 1 transmit. | |
| Ethernet PHY | ERBIAS | 33 | O | Analog | 12.4-kΩ resistor (1% precision) used internally for Ethernet PHY. |
| | LED0 | 59 | O | TTL | MII LED 0. |
| | LED1 | 60 | O | TTL | MII LED 1. |
| | MDIO | 58 | I/O | OD | 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 | TTL | TXON of the Ethernet PHY. |
| | TXOP | 43 | O | TTL | TXOP of the Ethernet PHY. |
| | XTALNPHY | 17 | I | Analog | XTALN of the Ethernet PHY. |
| XTALPPHY | 16 | O | Analog | XTALP of the Ethernet PHY. | |

Table 24-4. Signals by Function, Except for GPIO (continued)

| Function | Pin Name | Pin Number | Pin Type | Buffer Type ^a | Description |
|-------------------------------|----------|------------|----------|--------------------------|-------------------------|
| External Peripheral Interface | EPI0S0 | 83 | I/O | TTL | EPI module 0 signal 0. |
| | EPI0S1 | 84 | I/O | TTL | EPI module 0 signal 1. |
| | EPI0S2 | 25 | I/O | TTL | EPI module 0 signal 2. |
| | EPI0S3 | 24 | I/O | TTL | EPI module 0 signal 3. |
| | EPI0S4 | 23 | I/O | TTL | EPI module 0 signal 4. |
| | EPI0S5 | 22 | I/O | TTL | EPI module 0 signal 5. |
| | EPI0S6 | 86 | I/O | TTL | EPI module 0 signal 6. |
| | EPI0S7 | 85 | I/O | TTL | EPI module 0 signal 7. |
| | EPI0S8 | 74 | I/O | TTL | EPI module 0 signal 8. |
| | EPI0S9 | 75 | I/O | TTL | EPI module 0 signal 9. |
| | EPI0S10 | 76 | I/O | TTL | EPI module 0 signal 10. |
| | EPI0S11 | 63 | I/O | TTL | EPI module 0 signal 11. |
| | EPI0S12 | 42 | I/O | TTL | EPI module 0 signal 12. |
| | EPI0S13 | 19 | I/O | TTL | EPI module 0 signal 13. |
| | EPI0S14 | 18 | I/O | TTL | EPI module 0 signal 14. |
| | EPI0S15 | 41 | I/O | TTL | EPI module 0 signal 15. |
| | EPI0S16 | 14 | I/O | TTL | EPI module 0 signal 16. |
| | EPI0S17 | 87 | I/O | TTL | EPI module 0 signal 17. |
| | EPI0S18 | 39 | I/O | TTL | EPI module 0 signal 18. |
| | EPI0S19 | 97 | I/O | TTL | EPI module 0 signal 19. |
| | EPI0S20 | 12 | I/O | TTL | EPI module 0 signal 20. |
| | EPI0S21 | 13 | I/O | TTL | EPI module 0 signal 21. |
| | EPI0S22 | 91 | I/O | TTL | EPI module 0 signal 22. |
| | EPI0S23 | 92 | I/O | TTL | EPI module 0 signal 23. |
| | EPI0S24 | 95 | I/O | TTL | EPI module 0 signal 24. |
| | EPI0S25 | 96 | I/O | TTL | EPI module 0 signal 25. |
| | EPI0S26 | 62 | I/O | TTL | EPI module 0 signal 26. |
| | EPI0S27 | 15 | I/O | TTL | EPI module 0 signal 27. |
| | EPI0S28 | 98 | I/O | TTL | EPI module 0 signal 28. |
| | EPI0S29 | 99 | I/O | TTL | EPI module 0 signal 29. |
| | EPI0S30 | 100 | I/O | TTL | EPI module 0 signal 30. |
| EPI0S31 | 36 | I/O | TTL | EPI module 0 signal 31. | |

Table 24-4. Signals by Function, Except for GPIO (continued)

| Function | Pin Name | Pin Number | Pin Type | Buffer Type ^a | Description |
|------------------------|----------|--|----------|--------------------------|------------------------|
| General-Purpose Timers | CCP0 | 13 22 23 39 42 66 72 91 97 | I/O | TTL | Capture/Compare/PWM 0. |
| | CCP1 | 24 25 34 67 90 96 100 | I/O | TTL | Capture/Compare/PWM 1. |
| | CCP2 | 6 11 25 41 67 75 91 95 98 | I/O | TTL | Capture/Compare/PWM 2. |
| | CCP3 | 6 23 24 35 61 72 74 97 | I/O | TTL | Capture/Compare/PWM 3. |
| | CCP4 | 22 25 35 95 98 | I/O | TTL | Capture/Compare/PWM 4. |
| | CCP5 | 5 12 25 36 90 91 | I/O | TTL | Capture/Compare/PWM 5. |
| | CCP6 | 10 12 75 86 91 | I/O | TTL | Capture/Compare/PWM 6. |
| | CCP7 | 11 13 85 90 96 | I/O | TTL | Capture/Compare/PWM 7. |

Table 24-4. 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 that indicates the processor is in Hibernate mode. |
| | VBAT | 55 | - | Power | Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply. |
| | WAKE | 50 | I | TTL | An external input that brings the processor out of Hibernate mode when asserted. |
| | XOSC0 | 52 | I | Analog | Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register. |
| | XOSC1 | 53 | O | Analog | Hibernation module oscillator crystal output. |
| I2C | I2C0SCL | 72 | I/O | OD | I ² C module 0 clock. |
| | I2C0SDA | 65 | I/O | OD | I ² C module 0 data. |
| | I2C1SCL | 14 19 26 34 | I/O | OD | I ² C module 1 clock. |
| | I2C1SDA | 18 27 35 87 | I/O | OD | I ² C module 1 data. |
| I2S | I2S0RXMCLK | 29 98 | I/O | TTL | I ² S module 0 receive master clock. |
| | I2S0RXSCK | 10 | I/O | TTL | I ² S module 0 receive clock. |
| | I2S0RXSD | 28 97 | I/O | TTL | I ² S module 0 receive data. |
| | I2S0RXWS | 11 | I/O | TTL | I ² S module 0 receive word select. |
| | I2S0TXMCLK | 61 | I/O | TTL | I ² S module 0 transmit master clock. |
| | I2S0TXSCK | 30 90 99 | I/O | TTL | I ² S module 0 transmit clock. |
| | I2S0TXSD | 5 47 | I/O | TTL | I ² S module 0 transmit data. |
| | I2S0TXWS | 6 31 100 | I/O | TTL | I ² S module 0 transmit word select. |
| JTAG/SWD/SWO | SWCLK | 80 | I | TTL | JTAG/SWD CLK. |
| | SWDIO | 79 | I/O | TTL | JTAG TMS and SWDIO. |
| | SWO | 77 | O | TTL | JTAG TDO and SWO. |
| | TCK | 80 | I | TTL | JTAG/SWD CLK. |
| | TDI | 78 | I | TTL | JTAG TDI. |
| | TDO | 77 | O | TTL | JTAG TDO and SWO. |
| | TMS | 79 | I | TTL | JTAG TMS and SWDIO. |

Table 24-4. Signals by Function, Except for GPIO (continued)

| Function | Pin Name | Pin Number | Pin Type | Buffer Type ^a | Description |
|----------|----------|---|----------|--------------------------|--|
| Power | GND | 9 21 45 54 57 69 82 94 | - | Power | Ground reference for logic and I/O pins. |
| | GNDA | 4 | - | 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 μ F or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s). |
| | VDD | 8 20 32 44 56 68 81 93 | - | Power | Positive supply for I/O and some logic. |
| | VDDA | 3 | - | Power | The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. |
| | VDDC | 38 88 | - | Power | Positive supply for most of the logic function, including the processor core and most peripherals. |
| SSI | SSI0Clk | 28 | I/O | TTL | SSI module 0 clock. |
| | SSI0Fss | 29 | I/O | TTL | SSI module 0 frame. |
| | SSI0Rx | 30 | I | TTL | SSI module 0 receive. |
| | SSI0Tx | 31 | O | TTL | SSI module 0 transmit. |
| | SSI1Clk | 60 74 76 | I/O | TTL | SSI module 1 clock. |
| | SSI1Fss | 59 63 75 | I/O | TTL | SSI module 1 frame. |
| | SSI1Rx | 42 62 95 | I | TTL | SSI module 1 receive. |
| | SSI1Tx | 15 41 96 | O | TTL | SSI module 1 transmit. |

Table 24-4. Signals by Function, Except for GPIO (continued)

| Function | Pin Name | Pin Number | Pin Type | Buffer Type ^a | Description |
|-------------------------|----------|----------------------------------|----------|--------------------------|---|
| System Control & Clocks | NMI | 89 | I | TTL | Non-maskable interrupt. |
| | OSC0 | 48 | I | Analog | Main oscillator crystal input or an external clock reference input. |
| | OSC1 | 49 | O | Analog | Main oscillator crystal output. |
| | 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. |
| | U1CTS | 2 10 34 | I | TTL | UART module 1 Clear To Send modem status input signal. |
| | U1DCD | 1 11 35 | I | TTL | UART module 1 Data Carrier Detect modem status input signal. |
| | U1DSR | 47 | I | TTL | UART module 1 Data Set Ready modem output control line. |
| | U1DTR | 100 | O | TTL | UART module 1 Data Terminal Ready modem status input signal. |
| | U1RI | 97 | I | TTL | UART module 1 Ring Indicator modem status input signal. |
| | U1RTS | 61 | O | TTL | UART module 1 Request to Send modem output control line. |
| | U1Rx | 10 12 23 26 66 92 | I | TTL | UART module 1 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U1Tx | 11 13 22 27 67 91 | O | TTL | UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation. |
| | U2Rx | 10 19 92 98 | I | TTL | UART module 2 receive. When in IrDA mode, this signal has IrDA modulation. |
| | U2Tx | 6 11 18 99 | O | TTL | UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation. |

Table 24-4. Signals by Function, Except for GPIO (continued)

| Function | Pin Name | Pin Number | Pin Type | Buffer Type ^a | Description |
|----------|-----------|--|----------|--------------------------|---|
| USB | USB0DM | 70 | I/O | Analog | Bidirectional differential data pin (D- per USB specification). |
| | USB0DP | 71 | I/O | Analog | Bidirectional differential data pin (D+ per USB specification). |
| | USB0EPEN | 19 24 34 72 83 | O | TTL | Used in Host mode to control an external power source to supply power to the USB bus. |
| | USB0ID | 66 | I | Analog | This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is an A device and pulled up is a B device). |
| | USB0PFLT | 22 23 35 65 74 76 87 | I | TTL | Used in Host mode by an external power source to indicate an error state by that power source. |
| | USB0RBIAS | 73 | O | Analog | 9.1-kΩ resistor (1% precision) used internally for USB analog circuitry. |
| | USB0VBUS | 67 | I/O | Analog | This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing. |

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

Table 24-5. GPIO Pins and Alternate Functions

| IO | Pin | Analog Function | Digital Function (GPIOPCTL PMCx Bit Field Encoding) ^a | | | | | | | | | | | |
|-----|-----|-----------------|--|------|------|------|--------|--------|------|----------|----------|------------|----|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | |
| PA0 | 26 | - | U0Rx | - | - | - | - | - | - | - | I2C1SCL | U1Rx | - | - |
| PA1 | 27 | - | U0Tx | - | - | - | - | - | - | - | I2C1SDA | U1Tx | - | - |
| PA2 | 28 | - | SSI0Clk | - | - | - | - | - | - | - | - | I2S0RXSD | - | - |
| PA3 | 29 | - | SSI0Fss | - | - | - | - | - | - | - | - | I2S0RXMCLK | - | - |
| PA4 | 30 | - | SSI0Rx | - | - | - | - | CAN0Rx | - | - | - | I2S0TXSCK | - | - |
| PA5 | 31 | - | SSI0Tx | - | - | - | - | CAN0Tx | - | - | - | I2S0TXWS | - | - |
| PA6 | 34 | - | I2C1SCL | CCP1 | - | - | - | CAN0Rx | - | - | USB0EPEN | U1CTS | - | - |
| PA7 | 35 | - | I2C1SDA | CCP4 | - | - | - | CAN0Tx | CCP3 | USB0PFLT | U1DCD | - | - | - |
| PB0 | 66 | USB0ID | CCP0 | - | - | - | U1Rx | - | - | - | - | - | - | - |
| PB1 | 67 | USB0VBUS | CCP2 | - | - | CCP1 | U1Tx | - | - | - | - | - | - | - |
| PB2 | 72 | - | I2C0SCL | - | - | - | CCP3 | CCP0 | - | - | USB0EPEN | - | - | - |
| PB3 | 65 | - | I2C0SDA | - | - | - | - | - | - | - | USB0PFLT | - | - | - |
| PB4 | 92 | AIN10 C0- | - | - | - | U2Rx | CAN0Rx | - | U1Rx | EPI0S23 | - | - | - | - |
| PB5 | 91 | AIN11 C1- | C0o | CCP5 | CCP6 | CCP0 | CAN0Tx | CCP2 | U1Tx | EPI0S22 | - | - | - | - |

Table 24-5. GPIO Pins and Alternate Functions (continued)

| IO | Pin | Analog Function | Digital Function (GPIOCTL PMCx Bit Field Encoding) ^a | | | | | | | | | | |
|-----|-----|-----------------|---|---------|--------------|------|------|----------|----------|-----------|-----------|---------|----|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PB6 | 90 | VREFA C0+ | CCP1 | CCP7 | C0o | - | - | CCP5 | - | - | I2S0TXSCK | - | - |
| PB7 | 89 | - | - | - | - | NMI | - | - | - | - | - | - | - |
| PC0 | 80 | - | - | - | TCK SWCLK | - | - | - | - | - | - | - | - |
| PC1 | 79 | - | - | - | TMS SWDIO | - | - | - | - | - | - | - | - |
| PC2 | 78 | - | - | - | TDI | - | - | - | - | - | - | - | - |
| PC3 | 77 | - | - | - | TDO SWO | - | - | - | - | - | - | - | - |
| PC4 | 25 | - | CCP5 | - | - | - | CCP2 | CCP4 | - | EPI0S2 | CCP1 | - | - |
| PC5 | 24 | C1+ | CCP1 | C1o | C0o | - | CCP3 | USB0EPEN | - | EPI0S3 | - | - | - |
| PC6 | 23 | C2+ | CCP3 | - | C2o | - | U1Rx | CCP0 | USB0PFLT | EPI0S4 | - | - | - |
| PC7 | 22 | C2- | CCP4 | - | - | CCP0 | U1Tx | USB0PFLT | C1o | EPI0S5 | - | - | - |
| PD0 | 10 | AIN15 | - | CAN0Rx | - | U2Rx | U1Rx | CCP6 | - | I2S0RXSCK | U1CTS | - | - |
| PD1 | 11 | AIN14 | - | CAN0Tx | - | U2Tx | U1Tx | CCP7 | - | I2S0RXWS | U1DCD | CCP2 | - |
| PD2 | 12 | AIN13 | U1Rx | CCP6 | - | CCP5 | - | - | - | EPI0S20 | - | - | - |
| PD3 | 13 | AIN12 | U1Tx | CCP7 | - | CCP0 | - | - | - | EPI0S21 | - | - | - |
| PD4 | 97 | AIN7 | CCP0 | CCP3 | - | - | - | - | - | I2S0RXSD | U1RI | EPI0S19 | - |
| PD5 | 98 | AIN6 | CCP2 | CCP4 | - | - | - | - | - | I2S0RXCLK | U2Rx | EPI0S28 | - |
| PD6 | 99 | AIN5 | - | - | - | - | - | - | - | I2S0TXSCK | U2Tx | EPI0S29 | - |
| PD7 | 100 | AIN4 | - | C0o | CCP1 | - | - | - | - | I2S0TXWS | U1DTR | EPI0S30 | - |
| PE0 | 74 | - | - | SSI1Clk | CCP3 | - | - | - | - | EPI0S8 | USB0PFLT | - | - |
| PE1 | 75 | - | - | SSI1Fss | - | CCP2 | CCP6 | - | - | EPI0S9 | - | - | - |
| PE2 | 95 | AIN9 | CCP4 | SSI1Rx | - | - | CCP2 | - | - | EPI0S24 | - | - | - |
| PE3 | 96 | AIN8 | CCP1 | SSI1Tx | - | - | CCP7 | - | - | EPI0S25 | - | - | - |
| PE4 | 6 | AIN3 | CCP3 | - | - | - | U2Tx | CCP2 | - | - | I2S0TXWS | - | - |
| PE5 | 5 | AIN2 | CCP5 | - | - | - | - | - | - | - | I2S0TXSD | - | - |
| PE6 | 2 | AIN1 | - | C1o | - | - | - | - | - | - | U1CTS | - | - |
| PE7 | 1 | AIN0 | - | C2o | - | - | - | - | - | - | U1DCD | - | - |
| PF0 | 47 | - | CAN1Rx | - | - | - | - | - | - | I2S0TXSD | U1DSR | - | - |
| PF1 | 61 | - | CAN1Tx | - | - | - | - | - | - | I2S0TXCLK | U1RTS | CCP3 | - |
| PF2 | 60 | - | LED1 | - | - | - | - | - | - | - | SSI1Clk | - | - |
| PF3 | 59 | - | LED0 | - | - | - | - | - | - | - | SSI1Fss | - | - |
| PF4 | 42 | - | CCP0 | C0o | - | - | - | - | - | EPI0S12 | SSI1Rx | - | - |
| PF5 | 41 | - | CCP2 | C1o | - | - | - | - | - | EPI0S15 | SSI1Tx | - | - |
| PG0 | 19 | - | U2Rx | - | I2C1SCL | - | - | - | USB0EPEN | EPI0S13 | - | - | - |
| PG1 | 18 | - | U2Tx | - | I2C1SDA | - | - | - | - | EPI0S14 | - | - | - |
| PG7 | 36 | - | - | - | - | - | - | - | - | CCP5 | EPI0S31 | - | - |
| PH0 | 86 | - | CCP6 | - | - | - | - | - | - | EPI0S6 | - | - | - |
| PH1 | 85 | - | CCP7 | - | - | - | - | - | - | EPI0S7 | - | - | - |
| PH2 | 84 | - | - | C1o | - | - | - | - | - | EPI0S1 | - | - | - |

Table 24-5. GPIO Pins and Alternate Functions (*continued*)

| IO | Pin | Analog Function | Digital Function (GPIOCTL PMCx Bit Field Encoding) ^a | | | | | | | | | | |
|-----|-----|-----------------|---|---|---|----------|---|---|---|---------|----------|----|---------|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| PH3 | 83 | - | - | - | - | USBOEPEN | - | - | - | EPIOS0 | - | - | - |
| PH4 | 76 | - | - | - | - | USB0PFLT | - | - | - | EPIOS10 | - | - | SSI1Clk |
| PH5 | 63 | - | - | - | - | - | - | - | - | EPIOS11 | - | - | SSI1Fss |
| PH6 | 62 | - | - | - | - | - | - | - | - | EPIOS26 | - | - | SSI1Rx |
| PH7 | 15 | - | - | - | - | - | - | - | - | EPIOS27 | - | - | SSI1Tx |
| PJ0 | 14 | - | - | - | - | - | - | - | - | EPIOS16 | - | - | I2C1SCL |
| PJ1 | 87 | - | - | - | - | - | - | - | - | EPIOS17 | USB0PFLT | - | I2C1SDA |
| PJ2 | 39 | - | - | - | - | - | - | - | - | EPIOS18 | CCP0 | - | - |

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

25 Operating Characteristics

Table 25-1. Temperature Characteristics

| Characteristic ^a | Symbol | Value | Unit |
|--|--------|------------|------|
| Industrial operating temperature range | T_A | -40 to +85 | °C |

a. Maximum storage temperature is 150°C.

Table 25-2. Thermal Characteristics

| Characteristic | Symbol | Value | Unit |
|---|---------------|-------------------------------------|------|
| Thermal resistance (junction to ambient) ^a | Θ_{JA} | 34 | °C/W |
| Average junction temperature ^b | T_J | $T_A + (P_{AVG} \cdot \Theta_{JA})$ | °C |

a. Junction to ambient thermal resistance Θ_{JA} numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

26 Electrical Characteristics

26.1 DC Characteristics

26.1.1 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 26-1. Maximum Ratings

| Parameter | Parameter Name ^a | Value | | Unit |
|------------------|--|-------|-----|------|
| | | Min | Max | |
| V _{DD} | I/O supply voltage (V _{DD}) | 0 | 4 | V |
| V _{DDA} | Analog supply voltage (V _{DDA}) | 0 | 4 | V |
| V _{BAT} | Battery supply voltage (V _{BAT}) | 0 | 4 | V |
| V _{IN} | Input voltage | -0.3 | 5.5 | V |
| I | Maximum current per output pins | - | 25 | mA |

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}).

26.1.2 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 with the total number of high-current GPIO outputs not exceeding four for the entire package.

Table 26-2. 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 _{DDA} | Analog supply voltage | 3.0 | 3.3 | 3.6 | V |
| V _{DDC} ^a | Core supply voltage | 1.08 | 1.2 | 1.32 | V |
| V _{IH} | High-level input voltage | 2.0 | - | 5.0 | V |
| V _{IL} | Low-level input voltage | -0.3 | - | 1.3 | V |
| V _{SIH} | High-level input voltage for Schmitt trigger inputs | 0.8 * V _{DD} | - | V _{DD} | V |
| V _{SIL} | Low-level input voltage for Schmitt trigger inputs | 0 | - | 0.2 * V _{DD} | V |
| V _{OH} ^b | High-level output voltage | 2.4 | - | - | V |

Table 26-2. Recommended DC Operating Conditions (continued)

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|------------|---|-----|-----|-----|------|
| V_{OL}^a | Low-level output voltage | - | - | 0.4 | V |
| I_{OH} | High-level source current, $V_{OH}=2.4$ V | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |
| | 4-mA Drive | 4.0 | - | - | mA |
| | 8-mA Drive | 8.0 | - | - | mA |
| I_{OL} | Low-level sink current, $V_{OL}=0.4$ V | | | | |
| | 2-mA Drive | 2.0 | - | - | mA |
| | 4-mA Drive | 4.0 | - | - | mA |
| | 8-mA Drive | 8.0 | - | - | mA |

a. V_{DDC} is supplied from the output of the LDO.

b. V_{OL} and V_{OH} shift to 1.2 V when using high-current GPIOs.

26.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 26-3. LDO Regulator Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|-----------|--|------|-----|------|---------|
| C_{LDO} | External filter capacitor size for internal power supply | 1.0 | - | 3.0 | μ F |
| V_{LDO} | LDO output voltage | 1.08 | 1.2 | 1.32 | V |

26.1.4 Hibernation Module Characteristics

Table 26-4. Hibernation Module DC Characteristics

| Parameter | Parameter Name | Min | Nominal | Max | Unit |
|--------------|----------------------------|-----|---------|-----|------|
| V_{BAT} | Battery supply voltage | 2.4 | 3.0 | 3.6 | V |
| V_{LOWBAT} | Low battery detect voltage | - | 2.35 | - | V |

26.1.5 Flash Memory Characteristics

Table 26-5. Flash Memory Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|-------------|--|--------|-----|-----|--------|
| PE_{CYC} | Number of guaranteed mass program/erase cycles before failure ^a | 15,000 | - | - | cycles |
| T_{RET} | Data retention at average operating temperature of 125°C | 10 | - | - | years |
| T_{PROG} | Word program time | - | - | 1 | ms |
| T_{BPROG} | Buffer program time | - | - | 1 | ms |
| T_{ERASE} | Page erase time | - | - | 12 | ms |
| T_{ME} | Mass erase time | - | - | 12 | ms |

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1. Caution should be used when performing block erases, as repeated block erases can shorten the number of guaranteed erase cycles, see "Flash Memory Programming" on page 228.

26.1.6 GPIO Module Characteristics

Table 26-6. 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 |

26.1.7 USB Module Characteristics

The Stellaris® USB controller DC electrical specifications are compliant with the “Universal Serial Bus Specification Rev. 2.0” (full-speed and low-speed support) and the “On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0”. Some components of the USB system are integrated within the LM3S9B90 microcontroller and specific to the Stellaris® microcontroller design. An external component resistor is needed as specified in Table 26-7.

Table 26-7. USB Controller DC Characteristics

| Parameter | Parameter Name | Value | Unit |
|--------------------|--|------------|------|
| R _{UBIAS} | Value of the pull-down resistor on the USB0RBIAS pin | 9.1K ± 1 % | Ω |

26.1.8 Ethernet Controller Characteristics

Table 26-8. 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 |

26.1.9 Current Specifications

This section provides information on typical and maximum power consumption under various conditions.

26.1.9.1 Preliminary Current Consumption Specifications

The following table provides preliminary figures for current consumption while ongoing characterization is completed.

Table 26-9. Preliminary Current Consumption

| Parameter | Parameter Name | Conditions | Nom | Max | Unit |
|-----------------------|-------------------------|--|-----|-----|------|
| I _{DD_RUN} | Run mode 1 (Flash loop) | V _{DD} = 3.3 V Code= while(1){} executed in Flash Peripherals = All ON System Clock = 50 MHz (with PLL) Temp = 25°C | 56 | - | mA |
| I _{DD_SLEEP} | Sleep mode | V _{DD} = 3.3 V Peripherals = All clock gated System Clock = 50 MHz (with PLL) Temp = 25°C | 8 | - | mA |

Table 26-9. Preliminary Current Consumption (*continued*)

| Parameter | Parameter Name | Conditions | Nom | Max | Unit |
|---------------------|---|--|-----|-----|---------|
| $I_{DD_DEEPSLEEP}$ | Deep-sleep mode | Peripherals = All OFF System Clock = IOSC30KHZ/64 Temp = 25°C | - | 550 | μA |
| I_{HIB_NORTC} | Hibernate mode (external wake, RTC disabled, I/O not powered ^a) | $V_{BAT} = 3.0 V$ $V_{DD} = 0 V$ $V_{DDA} = 0 V$ Peripherals = All OFF System Clock = OFF Hibernate Module = 0 kHz | 8 | - | μA |
| I_{HIB_RTC} | Hibernate mode (RTC enabled, I/O not powered ^a) | $V_{BAT} = 3.0 V$ $V_{DD} = 3.3 V$ $V_{DDA} = 3.3 V$ Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz | 18 | - | μA |

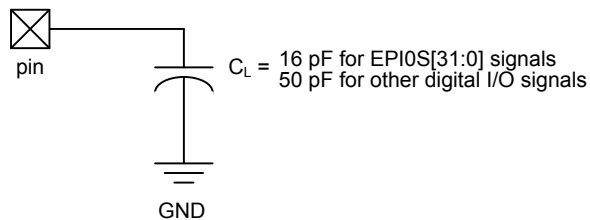
a. The VDD3ON mode must be disabled for the I/O ring to be unpowered.

26.2 AC Characteristics

26.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements.

Figure 26-1. Load Conditions



26.2.2 Clocks

The following sections provide specifications on the various clock sources and mode.

26.2.2.1 PLL Specifications

The following tables provide specifications for using the PLL.

Table 26-10. Phase Locked Loop (PLL) Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|-----------------|---------------------------------------|----------|-----|--------|------|
| f_{REF_XTAL} | Crystal reference ^a | 3.579545 | - | 16.384 | MHz |
| f_{REF_EXT} | External clock reference ^a | 3.579545 | - | 16.384 | MHz |
| f_{PLL} | PLL frequency ^b | - | 400 | - | MHz |

Table 26-10. Phase Locked Loop (PLL) Characteristics (continued)

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|--------------------|----------------|--------------------|-----|-------------------|------|
| T _{READY} | PLL lock time | 0.562 ^c | - | 1.38 ^d | ms |

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

c. Using a 16.384-MHz crystal

d. Using 3.5795-MHz crystal

Table 26-11 on page 1043 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

Table 26-11. Actual PLL Frequency

| XTAL | Crystal Frequency (MHz) | PLL Frequency (MHz) | Error |
|------|-------------------------|---------------------|---------|
| 0x04 | 3.5795 | 400.904 | 0.0023% |
| 0x05 | 3.6864 | 398.1312 | 0.0047% |
| 0x06 | 4.0 | 400 | - |
| 0x07 | 4.096 | 401.408 | 0.0035% |
| 0x08 | 4.9152 | 398.1312 | 0.0047% |
| 0x09 | 5.0 | 400 | - |
| 0x0A | 5.12 | 399.36 | 0.0016% |
| 0x0B | 6.0 | 400 | - |
| 0x0C | 6.144 | 399.36 | 0.0016% |
| 0x0D | 7.3728 | 398.1312 | 0.0047% |
| 0x0E | 8.0 | 400 | 0.0047% |
| 0x0F | 8.192 | 398.6773333 | 0.0033% |
| 0x10 | 10.0 | 400 | - |
| 0x11 | 12.0 | 400 | - |
| 0x12 | 12.288 | 401.408 | 0.0035% |
| 0x13 | 13.56 | 397.76 | 0.0056% |
| 0x14 | 14.318 | 400.90904 | 0.0023% |
| 0x15 | 16.0 | 400 | - |
| 0x16 | 16.384 | 404.1386667 | 0.010% |

26.2.2.2 PIOSC Specifications

Table 26-12. PIOSC Clock Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|------------------------|--|-----|--------|-----|------|
| f _{PIOSC25} | Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C | - | ±0.25% | ±1% | - |
| f _{PIOSCT} | Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C, across specified temperature range | - | - | ±3% | - |
| f _{PIOSCUCAL} | Internal 16-MHz precision oscillator frequency variance, user calibrated at a chosen temperature | - | ±0.25% | ±1% | - |

26.2.2.3 Internal 30-kHz Oscillator Specifications

Table 26-13. 30-kHz Clock Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|------------------------|--------------------------------------|-----|-----|-----|------|
| f _{IOSC30KHZ} | Internal 30-KHz oscillator frequency | 15 | 30 | 45 | KHz |

26.2.2.4 Hibernation Clock Source Specifications

Table 26-14. Hibernation Clock Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|----------------------------|---|----------------------|----------|---------|------|
| f _{HIBOSC} | Hibernation module oscillator frequency | - | 4.194304 | - | MHz |
| f _{HIBOSC_XTAL} | Crystal reference for hibernation oscillator | - | 4.194304 | - | MHz |
| f _{HIBOSC_EXT} | External clock reference for hibernation module | - | 32.768 | - | KHz |
| t _{HIBOSC_SETTLE} | Hibernation oscillator settling time ^a | pending ^b | - | pending | |

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

b. Pending characterization completion.

Table 26-15. HIB Oscillator Input Characteristics

| Name | Value | Condition |
|------------------------------------|----------|-----------|
| Frequency | 4.194304 | MHz |
| Frequency tolerance | ±100 | PPM |
| Oscillation mode | parallel | - |
| Equivalent series resistance (max) | 200 | Ω |
| Load capacitance | 16 | pF |
| Drive level (typ) | 100 | μW |

26.2.2.5 Main Oscillator Specifications

Table 26-16. Main Oscillator Clock Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|------------------------------|---|------|-----|--------|------|
| f _{MOSC} | Main oscillator frequency | 1 | - | 16.384 | MHz |
| t _{MOSC_PER} | Main oscillator period | 61 | - | 1000 | ns |
| t _{MOSC_SETTLE} | Main oscillator settling time | 17.5 | - | 20 | ms |
| f _{REF_XTAL_BYPASS} | Crystal reference using the main oscillator (PLL in BYPASS mode) ^a | 1 | - | 16.384 | MHz |
| f _{REF_EXT_BYPASS} | External clock reference (PLL in BYPASS mode) ^a | 0 | - | 80 | MHz |

a. The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Table 26-17. MOSC Oscillator Input Characteristics

| Name | Value | | | | | | Condition |
|---------------------|-------|------|------|------|------|------|-----------|
| Frequency | 16 | 12 | 8 | 6 | 4 | 3.5 | MHz |
| Frequency tolerance | ±100 | ±100 | ±100 | ±100 | ±100 | ±100 | PPM |

Table 26-17. MOSC Oscillator Input Characteristics (continued)

| Name | Value | | | | | | Condition |
|------------------------------------|----------|----------|----------|----------|----------|----------|---------------|
| | parallel | parallel | parallel | parallel | parallel | parallel | |
| Oscillation mode | parallel | parallel | parallel | parallel | parallel | parallel | - |
| Equivalent series resistance (max) | 70 | 90 | 120 | 160 | 200 | 220 | Ω |
| Load capacitance | 16 | 16 | 16 | 16 | 16 | 16 | pF |
| Drive level (typ) | 100 | 100 | 100 | 100 | 100 | 100 | μw |

26.2.3 JTAG and Boundary Scan

Table 26-18. 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 | $t_{\text{TCK_LOW}}$ | TCK clock Low time | - | t_{TCK} | - | ns |
| J4 | $t_{\text{TCK_HIGH}}$ | TCK clock High time | - | t_{TCK} | - | ns |
| J5 | $t_{\text{TCK_R}}$ | TCK rise time | 0 | - | 10 | ns |
| J6 | $t_{\text{TCK_F}}$ | TCK fall time | 0 | - | 10 | ns |
| J7 | $t_{\text{TMS_SU}}$ | TMS setup time to TCK rise | 20 | - | - | ns |
| J8 | $t_{\text{TMS_HLD}}$ | TMS hold time from TCK rise | 20 | - | - | ns |
| J9 | $t_{\text{TDI_SU}}$ | TDI setup time to TCK rise | 25 | - | - | ns |
| J10 | $t_{\text{TDI_HLD}}$ | TDI hold time from TCK rise | 25 | - | - | ns |
| J11 $t_{\text{TDO_ZDV}}$ | 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 $t_{\text{TDO_DV}}$ | 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 $t_{\text{TDO_DVZ}}$ | 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 |

Figure 26-2. JTAG Test Clock Input Timing

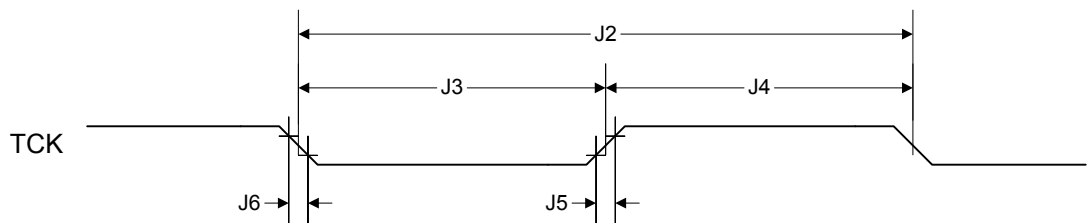
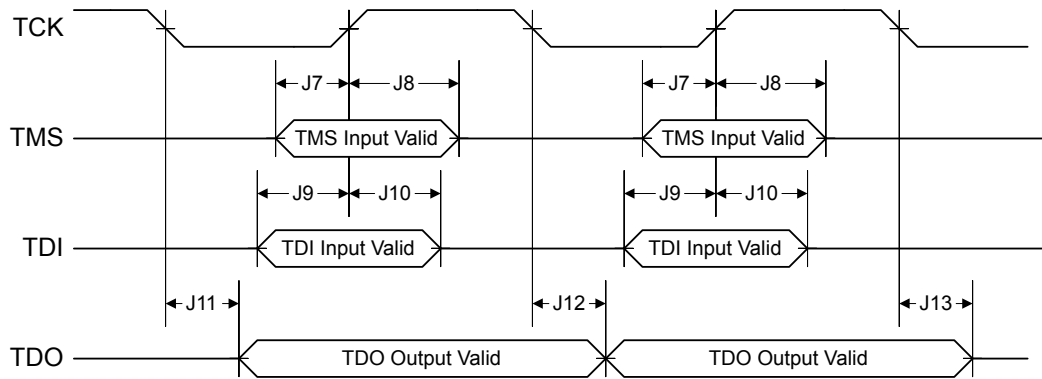


Figure 26-3. JTAG Test Access Port (TAP) Timing



26.2.4 Reset

Table 26-19. 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 | - | μ s |
| R5 | T_{IRPOR} | Internal reset timeout after POR | - | - | 95 | system clocks |
| R6 | T_{IRBOR} | Internal reset timeout after BOR | - | - | 7 | system clocks |
| R7 | T_{IRHWR} | Internal reset timeout after hardware reset (\overline{RST} pin) | - | - | 7 | system clocks |
| R8 | T_{IRSWR} | Internal reset timeout after software-initiated system reset | - | - | 16 | system clocks |
| R9 | T_{IRWDR} | Internal reset timeout after watchdog reset | - | - | 16 | system clocks |
| R10 | T_{IRMFR} | Internal reset timeout after MOSC failure reset | - | - | 32 | system clocks |
| R11 | $T_{VDDRISE}$ | Supply voltage (V_{DD}) rise time (0V-3.3V) | - | - | 250 | ms |
| R12 | T_{MIN} | Minimum \overline{RST} pulse width | 2 | - | - | μ s |

Figure 26-4. External Reset Timing (\overline{RST})

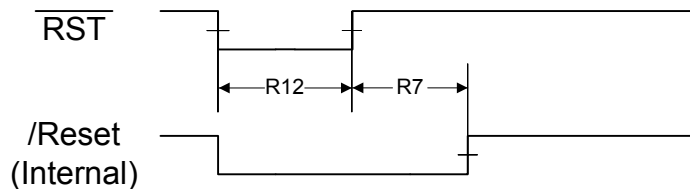


Figure 26-5. Power-On Reset Timing

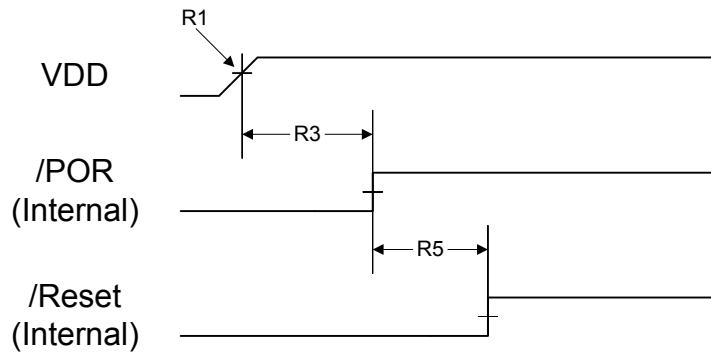


Figure 26-6. Brown-Out Reset Timing

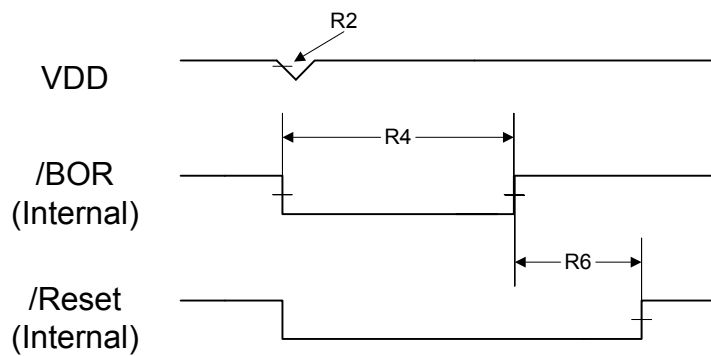


Figure 26-7. Software Reset Timing

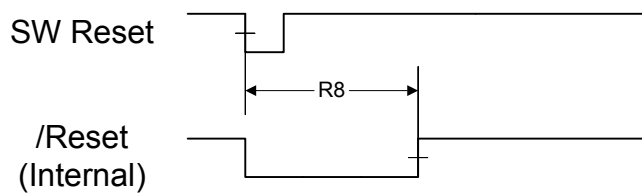


Figure 26-8. Watchdog Reset Timing

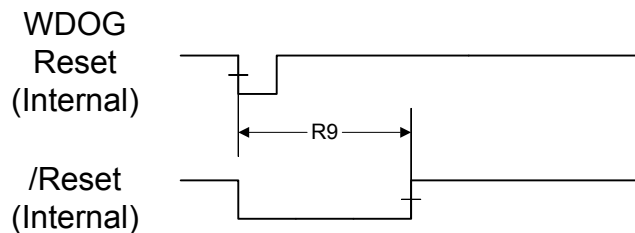
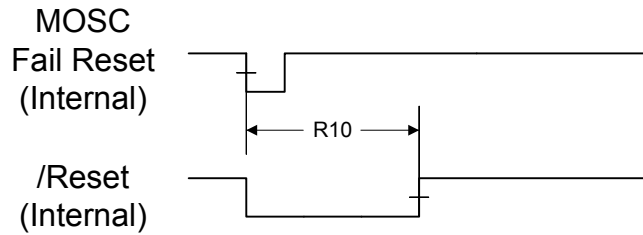


Figure 26-9. MOSC Failure Reset Timing



26.2.5 Hibernation Module

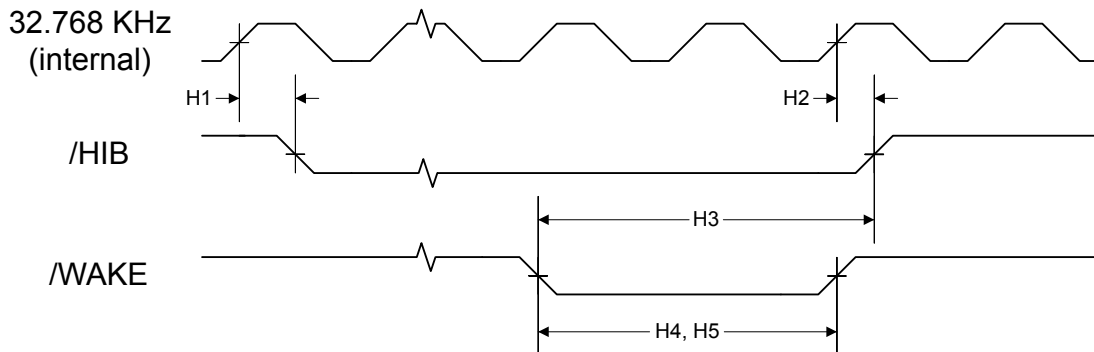
The Hibernation Module requires special system implementation considerations because it is intended to power down all other sections of its host device, refer to “Hibernation Module” on page 200.

Table 26-20. Hibernation Module AC Characteristics

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|--------------|----------------------------|--|---------|---------|---------|---------------|
| H1 | $t_{\text{HIB_LOW}}$ | Internal 32.768 KHz clock reference rising edge to /HIB asserted | 20 | - | - | μs |
| H2 | $t_{\text{HIB_HIGH}}$ | Internal 32.768 KHz clock reference rising edge to /HIB deasserted | - | 30 | - | μs |
| H3 | $t_{\text{WAKE_TO_HIB}}$ | /WAKE assert to /HIB desassert | 62 | - | 124 | μs |
| H4 | $t_{\text{WAKE_CLOCK}}$ | /WAKE assertion time when Hibernation oscillator is running | pending | pending | pending | μs |
| H5 | $t_{\text{WAKE_NOCLOCK}}$ | /WAKE assertion time when Hibernation oscillator is stopped ^a | pending | pending | pending | μs |

a. This mode is used when the `PINWEN` bit is set and the `RTCEN` bit is clear in the `HIBCTL` register.

Figure 26-10. Hibernation Module Timing



26.2.6 General-Purpose I/O (GPIO)

Note: All GPIOs are 5-V tolerant.

Table 26-21. GPIO Characteristics

| Parameter | Parameter Name | Condition | Min | Nom | Max | Unit |
|--------------------|---|-----------------------------------|-----|-----|-----|------|
| $t_{GPIO\text{R}}$ | GPIO Rise Time (from 20% to 80% of V_{DD}) | 2-mA drive | - | 14 | 20 | ns |
| | | 4-mA drive | | 7 | 10 | ns |
| | | 8-mA drive | | 4 | 5 | ns |
| | | 8-mA drive with slew rate control | | 6 | 8 | ns |
| $t_{GPIO\text{F}}$ | GPIO Fall Time (from 80% to 20% of V_{DD}) | 2-mA drive | - | 14 | 21 | ns |
| | | 4-mA drive | | 7 | 11 | ns |
| | | 8-mA drive | | 4 | 6 | ns |
| | | 8-mA drive with slew rate control | | 6 | 8 | ns |

26.2.7 External Peripheral Interface (EPI)

When the EPI module is in SDRAM mode, the drive strength must be configured to 8 mA. Table 26-22 on page 1049 shows the rise and fall times in SDRAM mode with 16 pF load conditions. When the EPI module is in Host-Bus or General-Purpose mode, the values in Table 26-21 on page 1049 should be used.

Table 26-22. EPI SDRAM Characteristics

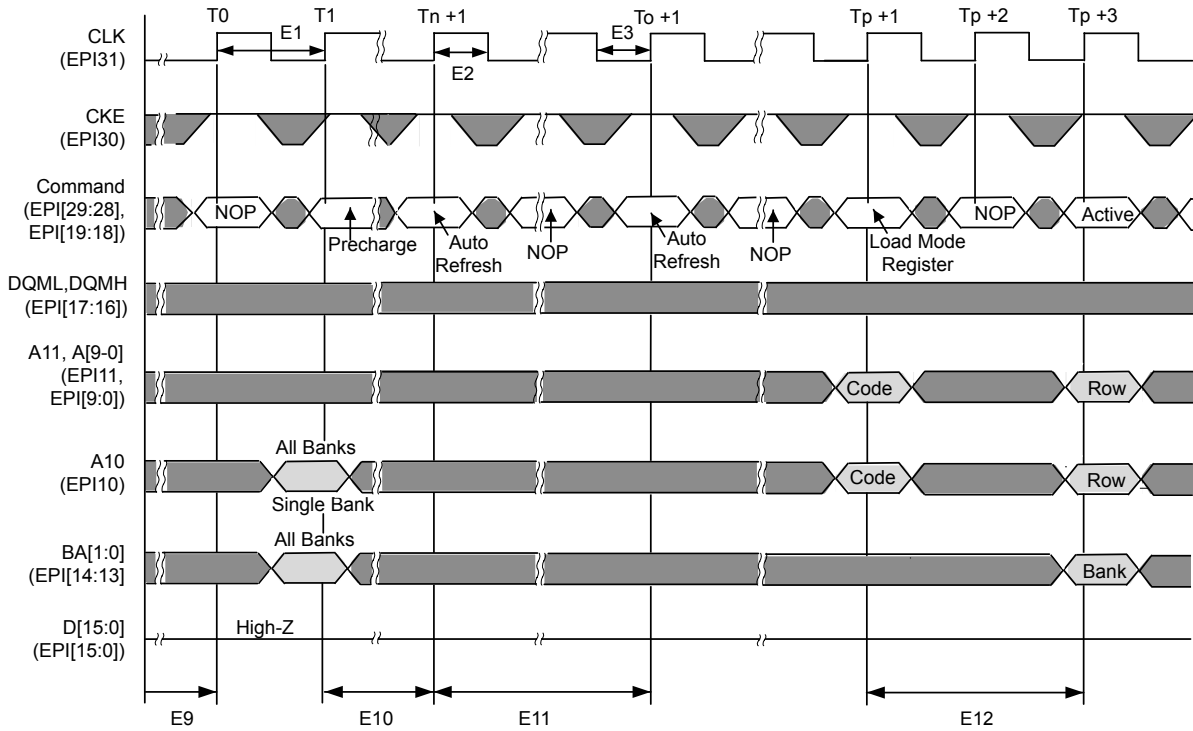
| Parameter | Parameter Name | Condition | Min | Nom | Max | Unit |
|---------------------|--|---------------------------|-----|-----|-----|------|
| $t_{SDRAM\text{R}}$ | EPI Rise Time (from 20% to 80% of V_{DD}) | 8-mA drive, $C_L = 16$ pF | - | 2 | 3 | ns |
| $t_{SDRAM\text{F}}$ | EPI Fall Time (from 80% to 20% of V_{DD}) | 8-mA drive, $C_L = 16$ pF | - | 2 | 3 | ns |

Table 26-23. EPI SDRAM Interface Characteristics^a

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|--------------|-----------|------------------------|-----|-----|-----|---------|
| E1 | t_{CK} | SDRAM Clock period | 20 | - | - | ns |
| E2 | t_{CH} | SDRAM Clock high time | 10 | - | - | ns |
| E3 | t_{CL} | SDRAM Clock low time | 10 | - | - | ns |
| E4 | t_{COV} | CLK to output valid | -5 | - | 5 | ns |
| E5 | t_{COI} | CLK to output invalid | -5 | - | 5 | ns |
| E6 | t_{COT} | CLK to output tristate | -5 | - | 5 | ns |
| E7 | t_S | Input set up to CLK | 1.5 | - | - | ns |
| E8 | t_H | CLK to input hold | 0.8 | - | - | ns |
| E9 | t_{PU} | Power-up time | 100 | - | - | μ s |
| E10 | t_{RP} | Precharge all banks | 20 | - | - | ns |
| E11 | t_{RFC} | Auto refresh | 66 | - | - | ns |
| E12 | t_{MRD} | Program mode register | 40 | 40 | 40 | ns |

a. The EPI SDRAM interface must use 8-mA drive.

Figure 26-11. SDRAM Initialization and Load Mode Register Timing



Notes:

1. If CS is high at clock high time, all applied commands are NOP.
2. The **Mode** register may be loaded prior to the auto refresh cycles if desired.
3. JEDEC and PC100 specify three clocks.
4. Outputs are guaranteed High-Z after command is issued.

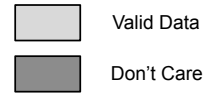


Figure 26-12. SDRAM Read Timing

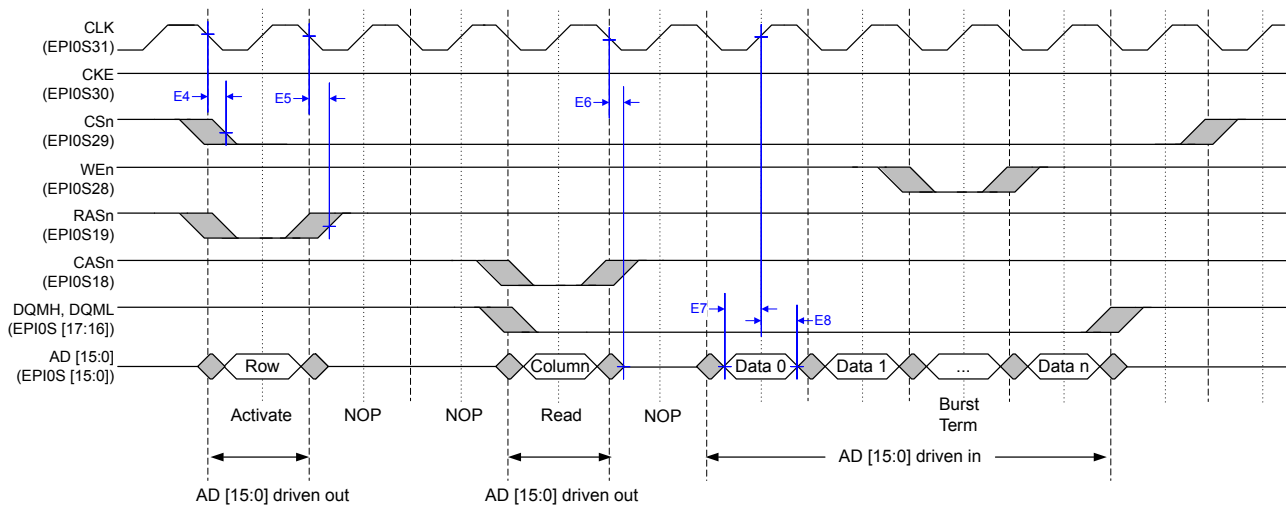


Figure 26-13. SDRAM Write Timing

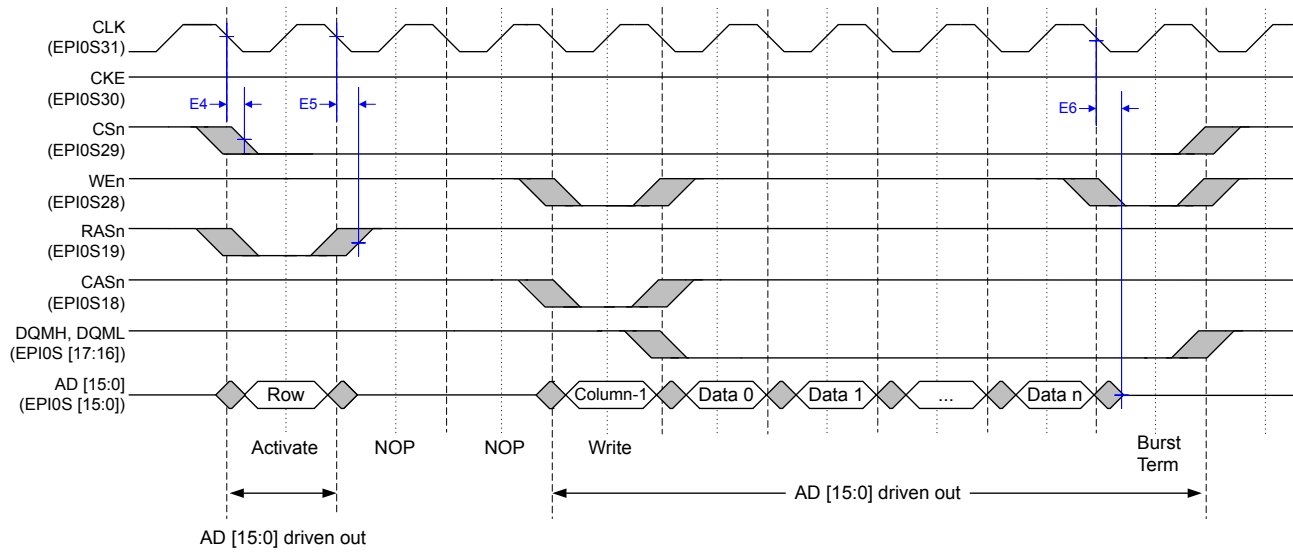


Table 26-24. EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|--------------|------------|----------------------------|-----|-----|-----|------|
| E14 | t_{ISU} | Data set up time | 10 | - | - | ns |
| E15 | t_{IH} | Data hold time | - | - | 10 | ns |
| E16 | t_{DV} | WEn to data valid | - | - | 5 | ns |
| E17 | t_{DI} | Data hold from WEn invalid | 40 | - | - | ns |
| E18 | t_{OV} | CSn to output valid | -5 | - | 5 | ns |
| E19 | t_{OINV} | CSn to output invalid | -5 | - | 5 | ns |

Figure 26-14. Host-Bus 8/16 Mode Read Timing

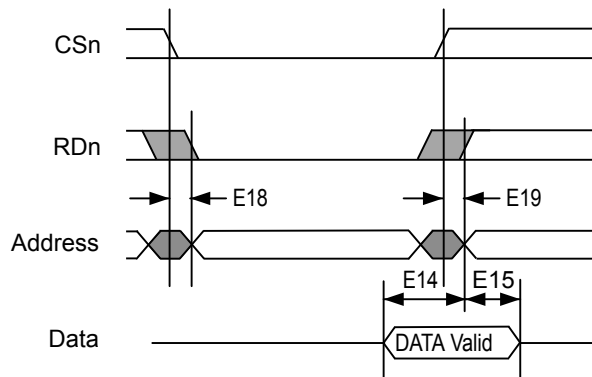


Figure 26-15. Host-Bus 8/16 Mode Write Timing

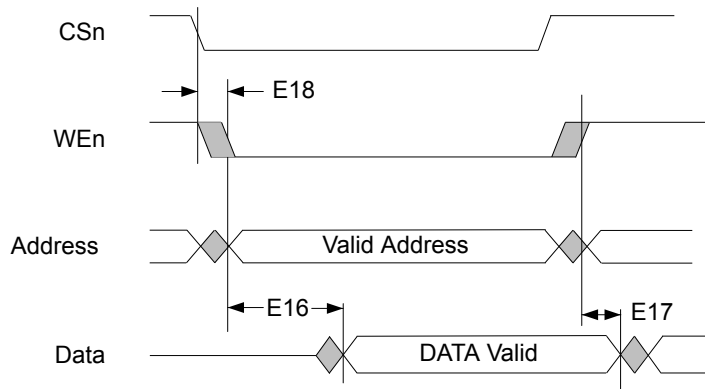
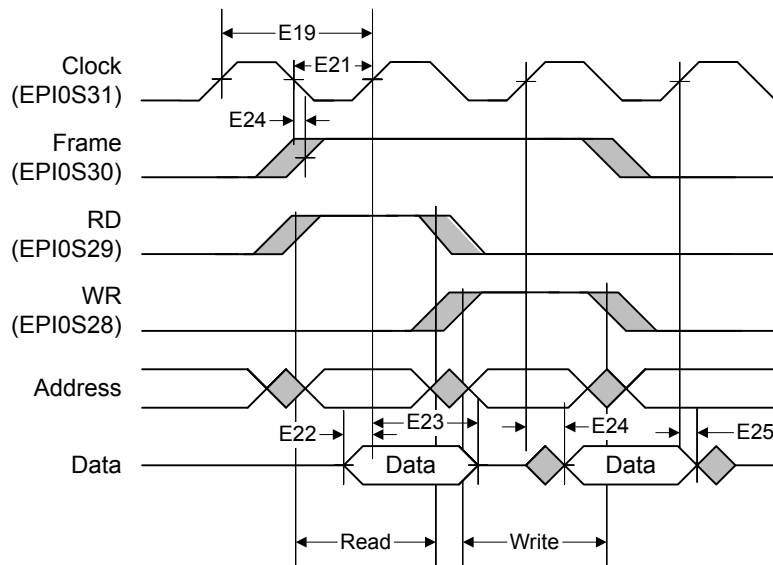


Table 26-25. EPI General-Purpose Interface Characteristics

| Parameter No | Parameter | Parameter Name | Min | Nom | Max | Unit |
|--------------|-----------|---|-----|-----|-----|------|
| E19 | t_{CK} | General-Purpose Clock period | 40 | - | - | ns |
| E20 | t_{CH} | General-Purpose Clock high time | 20 | - | - | ns |
| E21 | t_{CL} | General-Purpose Clock low time | 20 | - | - | ns |
| E22 | t_{ISU} | Input signal set up time to rising clock edge | 10 | - | - | ns |
| E23 | t_{IH} | Input signal hold time from rising clock edge | - | - | 10 | ns |
| E24 | t_{DV} | Falling clock edge to output valid | -5 | - | 5 | ns |
| E25 | t_{DI} | Falling clock edge to output invalid | -5 | - | 5 | ns |

Figure 26-16. General-Purpose Mode Read and Write Timing



The above figure illustrates accesses where the FRMCNT field is 0x0, the RD2CYC bit is clear, and the WR2CYC bit is clear.

26.2.8 Analog-to-Digital Converter

Table 26-26. ADC Characteristics^a

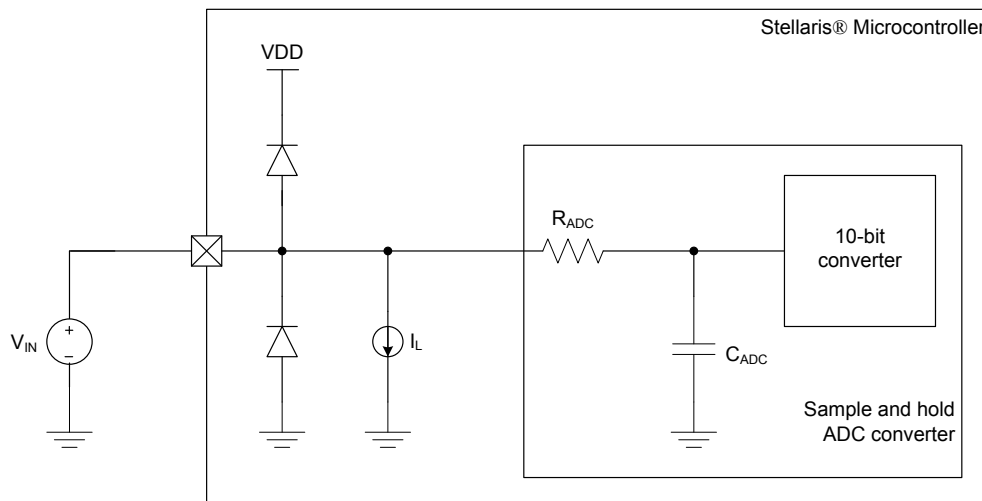
| Parameter | Parameter Name | Min | Nom | Max | Unit |
|----------------------|---|-----|------|---------|-----------------------------------|
| V_{ADCIN} | Maximum single-ended, full-scale analog input voltage | - | - | 3.0 | V |
| | Minimum single-ended, full-scale analog input voltage | 0.0 | - | - | 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 | 16 | | | MHz |
| t_{ADCCONV} | Conversion time | 16 | | | $t_{\text{ADCcycles}}^{\text{c}}$ |
| f_{ADCCONV} | Conversion rate | 875 | 1000 | 1125 | k samples/s |
| t_{LT} | Latency from trigger to start of conversion | - | 2 | - | system clocks |
| R_{ADC} | ADC equivalent resistance | - | 1.2 | 2 | k Ω |
| C_{ADC} | ADC equivalent capacitance | 0.9 | 1.0 | 1.1 | pF |
| E_{L} | Integral nonlinearity error | - | - | ± 1 | LSB |
| E_{D} | Differential nonlinearity error | - | - | ± 1 | LSB |
| E_{O} | Offset error | - | - | ± 1 | LSB |
| E_{G} | Full-scale gain error | - | - | ± 3 | LSB |
| E_{T} | Total unadjusted error | - | - | pending | LSB |
| E_{TS} | Temperature sensor accuracy | - | - | ± 5 | $^{\circ}\text{C}$ |

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 a 16-MHz clock source to operate properly.

c. $t_{\text{ADC}} = 1/f_{\text{ADC clock}}$

Figure 26-17. ADC Input Equivalency Diagram



26.2.9 ADC Module Characteristics

Table 26-27. ADC Module External Reference Characteristics^a

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|------------|---|-----|------------|----------|---------|
| V_{REFA} | External voltage reference for ADC ^b | 2.4 | - | V_{DD} | V |
| I_L | External voltage reference leakage current | - | ± 0.25 | - | μA |

a. Care must be taken to supply a reference voltage of acceptable quality.

b. Ground is always used as the reference level for the minimum conversion value.

Table 26-28. 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 | % |

26.2.10 Synchronous Serial Interface (SSI)

Table 26-29. 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 | - | 7.4 | 26 | ns |
| S5 | t_{DMD} | Data from master valid delay time | 0 | - | 20 | ns |
| S6 | t_{DMS} | Data from master setup time | 20 | - | - | ns |
| S7 | t_{DMH} | Data from master hold time | 40 | - | - | ns |
| S8 | t_{DSS} | Data from slave setup time | 20 | - | - | ns |
| S9 | t_{DSH} | Data from slave hold time | 40 | - | - | ns |

Figure 26-18. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

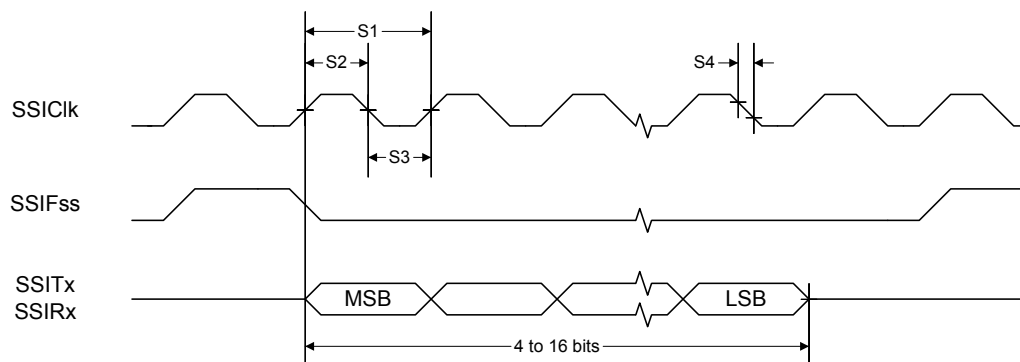


Figure 26-19. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

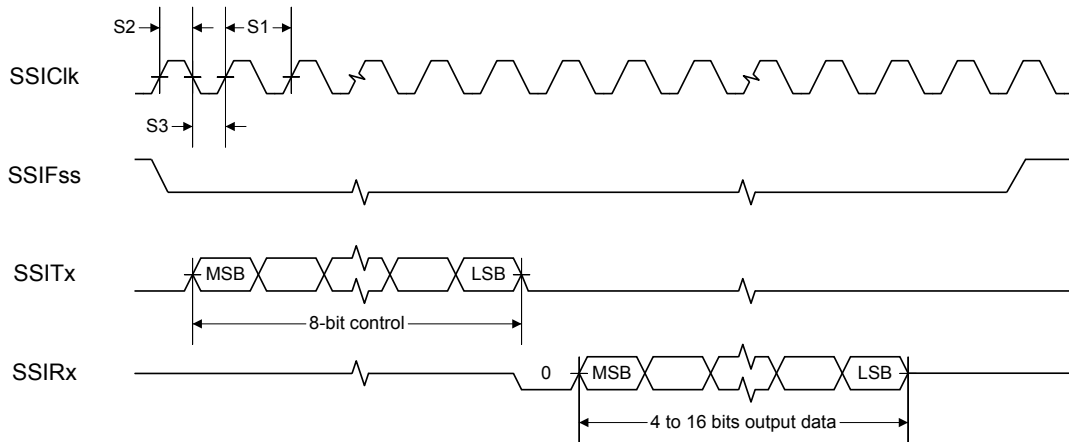
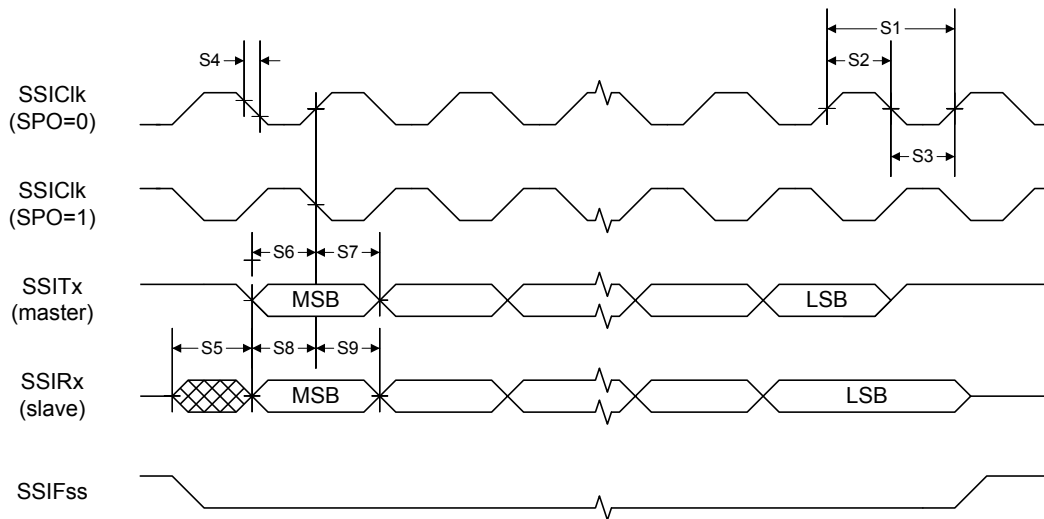
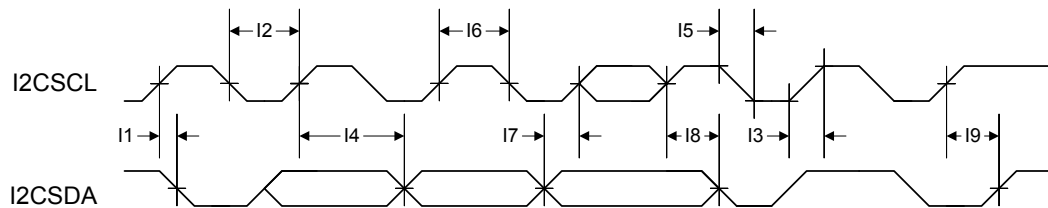


Figure 26-20. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



26.2.11 Inter-Integrated Circuit (I²C) Interface

Figure 26-21. I²C Timing



26.2.12 Inter-Integrated Circuit Sound (I²S) Interface

Table 26-30. I²S Master Clock (Receive and Transmit)

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---------------|------------------------|----------------|-------------------------------|-----|-----|------|
| M1 | t _{MCLK_PER} | Cycle time | 20.3 | - | - | ns |
| M2 | t _{MCLKRF} | Rise/fall time | See Table 26-21 on page 1049. | | | ns |
| M3 | t _{MCLK_HIGH} | High time | 10 | - | - | ns |
| M4 | t _{MCLK_LOW} | Low time | 10 | - | - | ns |
| M5 | t _{MDC} | Duty cycle | 48 | - | 52 | % |
| M6 | t _{MJITTER} | Jitter | - | - | 2.5 | ns |

Table 26-31. I²S Slave Clock (Receive and Transmit)

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---------------|------------------------|----------------|-----|-----|-----|------|
| M7 | t _{SCLK_PER} | Cycle time | 80 | - | - | ns |
| M8 | t _{SCLK_HIGH} | High time | 40 | - | - | ns |
| M9 | t _{SCLK_LOW} | Low time | 40 | - | - | ns |
| M10 | t _{SDC} | Duty cycle | - | 50 | - | % |

Table 26-32. I²S Master Mode

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---------------|-------------------|------------------------------|-----|-----|-----|------|
| M11 | t _{MSWS} | SCK fall to WS valid | - | - | 10 | ns |
| M12 | t _{MSD} | SCK fall to TXSD valid | - | - | 10 | ns |
| M13 | t _{MSDS} | RXSD setup time to SCK rise | 10 | - | - | ns |
| M14 | t _{MSDH} | RXSD hold time from SCK rise | 10 | - | - | ns |

Figure 26-22. I²S Master Mode Transmit Timing

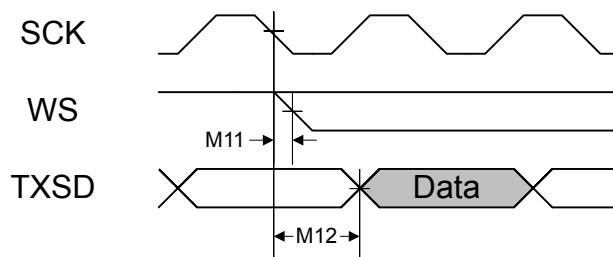


Figure 26-23. I²S Master Mode Receive Timing

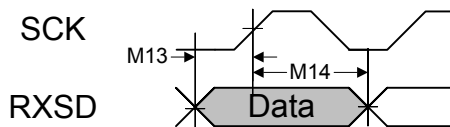
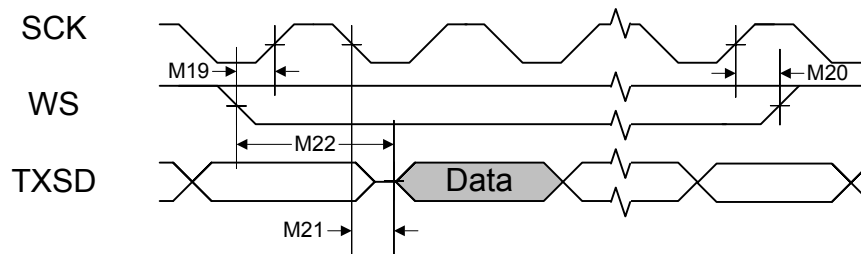
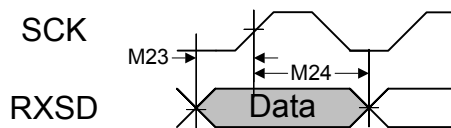


Table 26-33. I²S Slave Mode

| Parameter No. | Parameter | Parameter Name | Min | Nom | Max | Unit |
|---------------|------------------|---------------------------------|-----|-----|-----|------|
| M15 | t_{SCLK_PER} | Cycle time | 80 | - | - | ns |
| M16 | t_{SCLK_HIGH} | High time | 40 | - | - | ns |
| M17 | t_{SCLK_LOW} | Low time | 40 | - | - | ns |
| M18 | t_{SDC} | Duty cycle | - | 50 | - | % |
| M19 | t_{SSETUP} | WS setup time to SCK rise | - | - | 25 | ns |
| M20 | t_{SHOLD} | WS hold time from SCK rise | - | - | 10 | ns |
| M21 | t_{SSD} | SCK fall to TXSD valid | - | - | 20 | ns |
| M22 | t_{SLSD} | Left-justified mode, WS to TXSD | - | - | 20 | ns |
| M23 | t_{SSDS} | RXSD setup time to SCK rise | 10 | - | - | ns |
| M24 | t_{SSDH} | RXSD hold time from SCK rise | 10 | - | - | ns |

Figure 26-24. I²S Slave Mode Transmit TimingFigure 26-25. I²S Slave Mode Receive Timing

26.2.13 Ethernet Controller

Table 26-34. 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 26-35. 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 26-36. 100BASE-TX Receiver Characteristics

| Parameter Name | Min | Nom | Max | Unit |
|--------------------------------------|-----|-----|------|-------|
| Signal detect assertion threshold | 600 | 700 | - | mVppd |
| Signal detect de-assertion threshold | 350 | 425 | - | mVppd |
| Differential input resistance | - | 3.6 | - | kΩ |
| Jitter tolerance (pk-pk) | 4 | - | - | ns |
| Baseline wander tracking | -80 | - | +80 | % |
| Signal detect assertion time | - | - | 1000 | μs |
| Signal detect de-assertion time | - | - | 4 | μs |

Table 26-37. 10BASE-T Transmitter Characteristics^a

| Parameter Name | Min | Nom | Max | Unit |
|---------------------------------|-----|------------|-----|------|
| Peak differential output signal | 2.2 | - | 2.7 | V |
| Harmonic content | 27 | - | - | dB |
| Link pulse width | - | 100 | - | ns |
| Start-of-idle pulse width | - | 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 26-38. 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 26-39. 10BASE-T Receiver Characteristics

| Parameter Name | Min | Nom | Max | Unit |
|-------------------------------|-----|-----|-----|-------|
| Jitter tolerance (pk-pk) | 30 | 26 | - | ns |
| Input squelched threshold | 340 | 440 | 540 | mVppd |
| Differential input resistance | - | 3.6 | - | kΩ |
| Common-mode rejection | 25 | - | - | V |

Table 26-40. 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.

Table 26-41. Ethernet Reference Crystal

| Name | Value | Condition |
|---|--------------------------------------|-----------|
| Frequency | 25.00000 | MHz |
| Frequency tolerance | ±100 | PPM |
| Aging | ±2 | PPM/yr |
| Temperature stability (-40° to 85°) | ±5 | PPM |
| 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 |
| Serious resistance (max) | 60 | Ω |
| Spurious response (max) | > 5 dB below main within 500 kHz | |

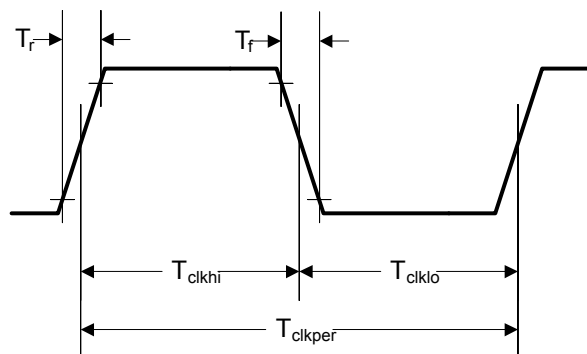
Figure 26-26. External XTLP Oscillator Characteristics

Table 26-42. External XTLP Oscillator Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|---------------------------------|-----------------------------|----------|------|----------|------|
| XTLN _{ILV} | XTLN Input Low Voltage | - | - | 0.8 | - |
| XTLP _F | XTLP Frequency ^a | - | 25.0 | - | - |
| T _{CLKPER} | XTLP Period ^a | - | 40 | - | - |
| XTLP _{DC} | XTLP Duty Cycle | 40 40 | - | 60 60 | % |
| T _r , T _f | Rise/Fall Time | - | - | 4.0 | ns |
| T _{JITTER} | Absolute Jitter | - | - | 0.1 | ns |

a. IEEE 802.3 frequency tolerance ± 50 ppm.

26.2.14 Universal Serial Bus (USB) Controller

The Stellaris[®] USB controller AC electrical specifications are compliant with the “Universal Serial Bus Specification Rev. 2.0” (full-speed and low-speed support) and the “On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0”.

26.2.15 Analog Comparator

Table 26-43. Analog Comparator Characteristics

| Parameter | Parameter Name | Min | Nom | Max | Unit |
|------------------|--|-----|----------|----------------------|---------|
| V _{OS} | Input offset voltage | - | ± 10 | ± 25 | mV |
| 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 | μ s |
| T _{MC} | Comparator mode change to Output Valid | - | - | 10 | μ s |

Table 26-44. 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 | - | - | $\pm 1/2$ | LSB |
| A _{LR} | Absolute accuracy low range | - | - | $\pm 1/4$ | LSB |

A Boot Loader

A.1 Boot Loader Overview

The Stellaris® Boot Loader is executed from the ROM when the Flash memory is empty and is used to download code to the Flash memory of a device without the use of a debug interface. The boot loader uses a simple packet interface to provide synchronous communication with the device. The speed of the boot loader is determined by the internal oscillator (PIOSC) frequency as it does not enable the PLL. The following serial interfaces can be used:

- UART0
- SSI0
- I²C0
- Ethernet

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

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

A.2 Serial Interfaces

This section describes how the boot loader operates using a serial interface. The following section describes the Ethernet operation.

A.2.1 Serial Configuration

Once communication with the boot loader is established via one of the serial interfaces, that interface is used until the boot loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the boot loader via the UART are disabled until the device is reset.

A.2.1.1 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 boot 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 internal oscillator (PIOSC) frequency of the board that is running the boot loader (which is at least 8.4 MHz, providing support for up to 262,500 baud). The maximum regular speed baud rate for any UART on a Stellaris® device is calculated as follows:

$$\text{Max Baud Rate} = \text{System Clock Frequency} / 16$$

In order to determine the baud rate, the boot loader must determine the relationship between the internal oscillator and the baud rate. With this information, the boot loader can configure the UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate to communicate with the device.

The method used to perform this automatic synchronization requires the host to send the boot loader two bytes that are both 0x55. With this series of pulses, the boot loader can 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 boot 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 boot loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the boot loader should be calculated as at least $2 * (20(\text{bits}/\text{sync}) / \text{baud rate} (\text{bits}/\text{sec}))$. For a baud rate of 115200, this time is $2 * (20/115200)$ or 0.35 ms.

A.2.1.2 SSI

The Synchronous Serial Interface (SSI) port also uses a fixed serial format for communications, with the default framing defined as Motorola format with both the `SPH` and `SPO` bits set in the **SSICRO** register. See “Frame Formats” on page 638 for more information on formats for this transfer protocol. Like the UART, this interface has hardware requirements that limit the maximum frequency of the `SSIClk` signal to be at most 1/12 the internal oscillator (`PIOSC`) frequency of the board running the boot loader (which is at least 8.4 MHz, providing support for up to 700 KHz). Because the host device is the master, the SSI on the boot loader device does not need to determine the clock as it is provided directly by the host.

A.2.1.3 I²C

The Inter-Integrated Circuit (I²C) port operates in slave mode with a slave address of 0x42. The I²C port works at both 100-kHz and 400-kHz `I2CSCL` clock frequency. Because the host device is the master, the I²C on the boot loader device does not need to determine the clock as it is provided directly by the host.

A.2.2 Serial 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.2.2.1 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[];
};
```

| | |
|-------------------------|--|
| <code>ucSize</code> | The first byte received holds the total size of the transfer including the size and checksum bytes. |
| <code>ucChecksum</code> | This holds a simple checksum of the bytes in the data buffer only. The algorithm is <code>Data[0]+Data[1]+...+ Data[ucSize-3]</code> . |
| <code>Data</code> | This is the raw data intended for the device, which is formatted in some form of command interface. There should be <code>ucSize-2</code> bytes of data provided in this buffer to or from the device. |

A.2.2.2 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 memory programming. This limitation is discussed further in the section that describes the boot loader command, `COMMAND_SEND_DATA` (see “`COMMAND_SEND_DATA (0x24)`” on page 1064).

Once the packet has been formatted correctly by the host, it should be sent out over the serial interface. Then the host should poll the interface for the first non-zero data returned from the device. The first non-zero byte is either an ACK (0xCC) or a NAK (0x33) byte from the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This response 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.2.2.3 Receiving Packets

The boot loader sends a packet of data in the same format that it receives a packet. The boot 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. The data is sent without a break after the first non-zero byte is sent from the boot loader. Once the device communicating with the boot 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 boot 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 boot loader, as the boot 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 boot loader.

A.2.3 Serial Commands

The next section defines the list of commands that can be sent to the boot 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.2.3.1 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. Because the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the boot loader.

A.2.3.2 COMMAND_DOWNLOAD (0x21)

This command is sent to the boot 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 and 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 boot 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.2.3.3 COMMAND_RUN (0x22)

This command is used to tell the boot 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 boot loader responds with an ACK signal back to the host device before actually executing the code at the given address. The ACK response tells the host that the command was received successfully, and the code is 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.2.3.4 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 boot loader knows that the data has been read.

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

A.2.3.5 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. For packets which do not contain the final portion of the downloaded data, a multiple of four bytes should always be transferred. 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 memory. If the boot loader sends a NAK to this command, the boot loader does not

increment the current address to allow retransmission of the previous data. The following example shows a `COMMAND_SEND_DATA` packet with 8 bytes of packet 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.2.3.6 `COMMAND_RESET (0x25)`

This command is used to tell the boot loader device to reset. Unlike the `COMMAND_RUN`, this command allows the initial stack pointer to be read by the hardware and set up for the new code.

`COMMAND_RESET` can also be used to reset the boot loader if a critical error occurs, and the host device wants to restart communication with the boot loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

The boot loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the boot loader. The ACK tells the host that the command was received successfully and the part will be reset.

A.3 Ethernet Interface

The Ethernet port uses the bootstrap protocol (BOOTP), which is a predecessor to the DHCP protocol and is used to discover the IP address of the client, the IP address of the server, and the name of the firmware image to use. BOOTP uses UDP/IP packets to communicate between the client and the server; the boot loader acts as the client. First, it sends a BOOTP request using a broadcast message. When the server receives the request, it replies, thereby informing the client of its IP address, the IP address of the server, and the name of the firmware image. Once this reply is received, the BOOTP protocol has completed. At this point, the trivial file transfer protocol (TFTP) is used to transfer the firmware image from the server to the client. TFTP also uses UDP/IP packets to communicate between the client and the server, and the boot loader also acts as the client in this protocol. As each data block is received, it is programmed into Flash memory. Once all data blocks are received and programmed, the device is reset, causing it to start running the new firmware image. The uIP stack is used to implement the UDP/IP connections. TCP support is not needed and is therefore disabled, greatly reducing the size of the stack.

The following IETF specifications define the protocols used by the Ethernet update mechanism:

- RFC951 (<http://tools.ietf.org/html/rfc951.html>) defines the bootstrap protocol.
- RFC1350 (<http://tools.ietf.org/html/rfc1350.html>) defines the trivial file transfer protocol.

B ROM DriverLib Functions

B.1 DriverLib Functions Included in the Integrated ROM

The Stellaris® Peripheral Driver Library (DriverLib) APIs that are available in the integrated ROM of the Stellaris® family of devices are listed below. The detailed description of each function is available in the *Stellaris® ROM User's Guide*.

ROM_ADCHardwareOversampleConfigure
// Configures the hardware oversampling factor of the ADC.

ROM_ADCIntClear
// Clears sample sequence interrupt source.

ROM_ADCIntDisable
// Disables a sample sequence interrupt.

ROM_ADCIntEnable
// Enables a sample sequence interrupt.

ROM_ADCIntStatus
// Gets the current interrupt status.

ROM_ADCProcessorTrigger
// Causes a processor trigger for a sample sequence.

ROM_ADCSequenceConfigure
// Configures the trigger source and priority of a sample sequence.

ROM_ADCSequenceDataGet
// Gets the captured data for a sample sequence.

ROM_ADCSequenceDisable
// Disables a sample sequence.

ROM_ADCSequenceEnable
// Enables a sample sequence.

ROM_ADCSequenceOverflow
// Determines if a sample sequence overflow occurred.

ROM_ADCSequenceOverflowClear
// Clears the overflow condition on a sample sequence.

ROM_ADCSequenceStepConfigure
// Configure a step of the sample sequencer.

ROM_ADCSequenceUnderflow
// Determines if a sample sequence underflow occurred.

ROM_ADCSequenceUnderflowClear
// Clears the underflow condition on a sample sequence.

ROM_CANBitRateSet
// This function is used to set the CAN bit timing values to a nominal setting based on a desired bit rate.

ROM_CANBitTimingGet
// Reads the current settings for the CAN controller bit timing.

ROM_CANBitTimingSet
// Configures the CAN controller bit timing.

ROM_CANDisable
// Disables the CAN controller.

ROM_CANEnable
// Enables the CAN controller.

ROM_CANErrCntGet
// Reads the CAN controller error counter register.

ROM_CANInit
// Initializes the CAN controller after reset.

ROM_CANIntClear
// Clears a CAN interrupt source.

ROM_CANIntDisable
// Disables individual CAN controller interrupt sources.

ROM_CANIntEnable
// Enables individual CAN controller interrupt sources.

ROM_CANIntStatus
// Returns the current CAN controller interrupt status.

ROM_CANMessageClear
// Clears a message object so that it is no longer used.

ROM_CANMessageGet
// Reads a CAN message from one of the message object buffers.

ROM_CANMessageSet
// Configures a message object in the CAN controller.

ROM_CANRetryGet
// Returns the current setting for automatic retransmission.

ROM_CANRetrySet
// Sets the CAN controller automatic retransmission behavior.

ROM_CANStatusGet
// Reads one of the controller status registers.

ROM_ComparatorConfigure
// Configures a comparator.

ROM_ComparatorIntClear
// Clears a comparator interrupt.

ROM_ComparatorIntDisable
// Disables the comparator interrupt.

ROM_ComparatorIntEnable
// Enables the comparator interrupt.

ROM_ComparatorIntStatus
// Gets the current interrupt status.

ROM_ComparatorRefSet
// Sets the internal reference voltage.

ROM_ComparatorValueGet
// Gets the current comparator output value.

ROM_Crc16Array
// Calculates the CRC-16 of an array of words.

ROM_Crc16Array3
// Calculates three CRC-16s of an array of words.

ROM_EthernetConfigGet
// Gets the current configuration of the Ethernet controller.

ROM_EthernetConfigSet
// Sets the configuration of the Ethernet controller.

ROM_EthernetDisable
// Disables the Ethernet controller.

ROM_EthernetEnable
// Enables the Ethernet controller for normal operation.

ROM_EthernetInitExpClk
// Initializes the Ethernet controller for operation.

ROM_EthernetIntClear
// Clears Ethernet interrupt sources.

ROM_EthernetIntDisable
// Disables individual Ethernet interrupt sources.

ROM_EthernetIntEnable
// Enables individual Ethernet interrupt sources.

ROM_EthernetIntStatus
// Gets the current Ethernet interrupt status.

ROM_EthernetMACAddrGet
// Gets the MAC address of the Ethernet controller.

ROM_EthernetMACAddrSet
// Sets the MAC address of the Ethernet controller.

ROM_EthernetPacketAvail
// Check for packet available from the Ethernet controller.

ROM_EthernetPacketGet
// Waits for a packet from the Ethernet controller.

ROM_EthernetPacketGetNonBlocking
// Receives a packet from the Ethernet controller.

ROM_EthernetPacketPut
// Waits to send a packet from the Ethernet controller.

ROM_EthernetPacketPutNonBlocking
// Sends a packet to the Ethernet controller.

ROM_EthernetPHYRead
// Reads from a PHY register.

ROM_EthernetPHYWrite
// Writes to the PHY register.

ROM_EthernetSpaceAvail
// Checks for packet space available in the Ethernet controller.

ROM_FlashErase
// Erases a block of flash.

ROM_FlashIntClear
// Clears flash controller interrupt sources.

ROM_FlashIntDisable
// Disables individual flash controller interrupt sources.

ROM_FlashIntEnable
// Enables individual flash controller interrupt sources.

ROM_FlashIntGetStatus
// Gets the current interrupt status.

ROM_FlashProgram
// Programs flash.

ROM_FlashProtectGet
// Gets the protection setting for a block of flash.

ROM_FlashProtectSave
// Saves the flash protection settings.

ROM_FlashProtectSet
// Sets the protection setting for a block of flash.

ROM_FlashUsecGet
// Gets the number of processor clocks per micro-second.

ROM_FlashUsecSet
// Sets the number of processor clocks per micro-second.

ROM_FlashUserGet
// Gets the user registers.

ROM_FlashUserSave
// Saves the user registers.

ROM_FlashUserSet
// Sets the user registers.

ROM_GPIODirModeGet
// Gets the direction and mode of a pin.

ROM_GPIODirModeSet
// Sets the direction and mode of the specified pin(s).

ROM_GPIOIntTypeGet
// Gets the interrupt type for a pin.

ROM_GPIOIntTypeSet
// Sets the interrupt type for the specified pin(s).

ROM_GPIOPadConfigGet
// Gets the pad configuration for a pin.

ROM_GPIOPadConfigSet
// Sets the pad configuration for the specified pin(s).

ROM_GPIOPinConfigure
// Configures the alternate function of a GPIO pin.

ROM_GPIOPinIntClear
// Clears the interrupt for the specified pin(s).

ROM_GPIOPinIntDisable
// Disables interrupts for the specified pin(s).

ROM_GPIOPinIntEnable
// Enables interrupts for the specified pin(s).

ROM_GPIOPinIntStatus
// Gets interrupt status for the specified GPIO port.

ROM_GPIOPinRead
// Reads the values present of the specified pin(s).

ROM_GPIOPinTypeADC
// Configures pin(s) for use as analog-to-digital converter inputs.

```
ROM_GPIOPinTypeCAN
    // Configures pin(s) for use as a CAN device.

ROM_GPIOPinTypeComparator
    // Configures pin(s) for use as an analog comparator input.

ROM_GPIOPinTypeGPIOInput
    // Configures pin(s) for use as GPIO inputs.

ROM_GPIOPinTypeGPIOOutput
    // Configures pin(s) for use as GPIO outputs.

ROM_GPIOPinTypeGPIOOutputOD
    // Configures pin(s) for use as GPIO open drain outputs.

ROM_GPIOPinTypeI2C
    // Configures pin(s) for use by the I2C peripheral.

ROM_GPIOPinTypeI2S
    // Configures pin(s) for use by the I2S peripheral.

ROM_GPIOPinTypeSSI
    // Configures pin(s) for use by the SSI peripheral.

ROM_GPIOPinTypeTimer
    // Configures pin(s) for use by the Timer peripheral.

ROM_GPIOPinTypeUART
    // Configures pin(s) for use by the UART peripheral.

ROM_GPIOPinTypeUSBDigital
    // Configures pin(s) for use by the USB peripheral.

ROM_GPIOPinWrite
    // Writes a value to the specified pin(s).

ROM_HibernateClockSelect
    // Selects the clock input for the Hibernation module.

ROM_HibernateDataGet
    // Reads a set of data from the non-volatile memory of the Hibernation module.

ROM_HibernateDataSet
    // Stores data in the non-volatile memory of the Hibernation module.

ROM_HibernateDisable
    // Disables the Hibernation module for operation.

ROM_HibernateEnableExpClk
    // Enables the Hibernation module for operation.

ROM_HibernateIntClear
    // Clears pending interrupts from the Hibernation module.
```

ROM_HibernateIntDisable
// Disables interrupts for the Hibernation module.

ROM_HibernateIntEnable
// Enables interrupts for the Hibernation module.

ROM_HibernateIntStatus
// Gets the current interrupt status of the Hibernation module.

ROM_HibernateIsActive
// Checks to see if the Hibernation module is already powered up.

ROM_HibernateLowBatGet
// Gets the currently configured low battery detection behavior.

ROM_HibernateLowBatSet
// Configures the low battery detection.

ROM_HibernateRequest
// Requests hibernation mode.

ROM_HibernateRTCDisable
// Disables the RTC feature of the Hibernation module.

ROM_HibernateRTCEnable
// Enables the RTC feature of the Hibernation module.

ROM_HibernateRTCGet
// Gets the value of the real time clock (RTC) counter.

ROM_HibernateRTCMatch0Get
// Gets the value of the RTC match 0 register.

ROM_HibernateRTCMatch0Set
// Sets the value of the RTC match 0 register.

ROM_HibernateRTCMatch1Get
// Gets the value of the RTC match 1 register.

ROM_HibernateRTCMatch1Set
// Sets the value of the RTC match 1 register.

ROM_HibernateRTCSet
// Sets the value of the real time clock (RTC) counter.

ROM_HibernateRTCTrimGet
// Gets the value of the RTC predivider trim register.

ROM_HibernateRTCTrimSet
// Sets the value of the RTC predivider trim register.

ROM_HibernateWakeGet
// Gets the currently configured wake conditions for the Hibernation module.

ROM_HibernateWakeSet
// Configures the wake conditions for the Hibernation module.

ROM_I2CMasterBusBusy
// Indicates whether or not the I2C bus is busy.

ROM_I2CMasterBusy
// Indicates whether or not the I2C Master is busy.

ROM_I2CMasterControl
// Controls the state of the I2C Master module.

ROM_I2CMasterDataGet
// Receives a byte that has been sent to the I2C Master.

ROM_I2CMasterDataPut
// Transmits a byte from the I2C Master.

ROM_I2CMasterDisable
// Disables the I2C master block.

ROM_I2CMasterEnable
// Enables the I2C Master block.

ROM_I2CMasterErr
// Gets the error status of the I2C Master module.

ROM_I2CMasterInitExpClk
// Initializes the I2C Master block.

ROM_I2CMasterIntClear
// Clears I2C Master interrupt sources.

ROM_I2CMasterIntDisable
// Disables the I2C Master interrupt.

ROM_I2CMasterIntEnable
// Enables the I2C Master interrupt.

ROM_I2CMasterIntStatus
// Gets the current I2C Master interrupt status.

ROM_I2CMasterSlaveAddrSet
// Sets the address that the I2C Master will place on the bus.

ROM_I2CSlaveDataGet
// Receives a byte that has been sent to the I2C Slave.

ROM_I2CSlaveDataPut
// Transmits a byte from the I2C Slave.

ROM_I2CSlaveDisable
// Disables the I2C slave block.

ROM_I2CSlaveEnable
// Enables the I2C Slave block.

ROM_I2CSlaveInit
// Initializes the I2C Slave block.

ROM_I2CSlaveIntClear
// Clears I2C Slave interrupt sources.

ROM_I2CSlaveIntClearEx
// Clears I2C Slave interrupt sources.

ROM_I2CSlaveIntDisable
// Disables the I2C Slave interrupt.

ROM_I2CSlaveIntDisableEx
// Disables individual I2C Slave interrupt sources.

ROM_I2CSlaveIntEnable
// Enables the I2C Slave interrupt.

ROM_I2CSlaveIntEnableEx
// Enables individual I2C Slave interrupt sources.

ROM_I2CSlaveIntStatus
// Gets the current I2C Slave interrupt status.

ROM_I2CSlaveIntStatusEx
// Gets the current I2C Slave interrupt status.

ROM_I2CSlaveStatus
// Gets the I2C Slave module status.

ROM_I2SIntClear
// Clears pending I2S interrupt sources.

ROM_I2SIntDisable
// Disables I2S interrupt sources.

ROM_I2SIntEnable
// Enables I2S interrupt sources.

ROM_I2SIntStatus
// Gets the I2S interrupt status.

ROM_I2SMasterClockSelect
// Selects the source of the master clock, internal or external.

ROM_I2SRxConfigSet
// Configures the I2S receive module.

ROM_I2SRxDataGet
// Reads data samples from the I2S receive FIFO with blocking.

ROM_I2SRxDataGetNonBlocking
// Reads data samples from the I2S receive FIFO without blocking.

ROM_I2SRxDisable
// Disables the I2S receive module for operation.

ROM_I2SRxEnable
// Enables the I2S receive module for operation.

ROM_I2SRxFIFOLevelGet
// Gets the number of samples in the receive FIFO.

ROM_I2SRxFIFOLimitGet
// Gets the current setting of the FIFO service request level.

ROM_I2SRxFIFOLimitSet
// Sets the FIFO level at which a service request is generated.

ROM_I2STxConfigSet
// Configures the I2S transmit module.

ROM_I2STxDataPut
// Writes data samples to the I2S transmit FIFO with blocking.

ROM_I2STxDataPutNonBlocking
// Writes data samples to the I2S transmit FIFO without blocking.

ROM_I2STxDisable
// Disables the I2S transmit module for operation.

ROM_I2STxEnable
// Enables the I2S transmit module for operation.

ROM_I2STxFIFOLevelGet
// Gets the number of samples in the transmit FIFO.

ROM_I2STxFIFOLimitGet
// Gets the current setting of the FIFO service request level.

ROM_I2STxFIFOLimitSet
// Sets the FIFO level at which a service request is generated.

ROM_I2STxRxConfigSet
// Configures the I2S transmit and receive modules.

ROM_I2STxRxDisable
// Disables the I2S transmit and receive modules.

ROM_I2STxRxEnable
// Enables the I2S transmit and receive modules for operation.

ROM_IntDisable
// Disables an interrupt.

ROM_IntEnable
// Enables an interrupt.

ROM_IntMasterDisable
// Disables the processor interrupt.

ROM_IntMasterEnable
// Enables the processor interrupt.

ROM_IntPriorityGet
// Gets the priority of an interrupt.

ROM_IntPriorityGroupingGet
// Gets the priority grouping of the interrupt controller.

ROM_IntPriorityGroupingSet
// Sets the priority grouping of the interrupt controller.

ROM_IntPrioritySet
// Sets the priority of an interrupt.

ROM_MPUDisable
// Disables the MPU for use.

ROM_MPUEnable
// Enables and configures the MPU for use.

ROM_MPURegionCountGet
// Gets the count of regions supported by th MPU.

ROM_MPURegionDisable
// Disables a specific region.

ROM_MPURegionEnable
// Enables a specific region.

ROM_MPURegionGet
// Gets the current settings for a specific region.

ROM_MPURegionSet
// Sets up the access rules for a specific region.

ROM_pvAESTable
// AES forward, reverse, S-box, and reverse S-box tables.

ROM_SSIConfigSetExpClk
// Configures the synchronous serial interface.

ROM_SSIDataGet
// Gets a data element from the SSI receive FIFO.

ROM_SSIDataGetNonBlocking
// Gets a data element from the SSI receive FIFO.

```
ROM_SSIDataPut
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDataPutNonBlocking
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDisable
    // Disables the synchronous serial interface.

ROM_SSIDMADisable
    // Disable SSI DMA operation.

ROM_SSIDMAEnable
    // Enable SSI DMA operation.

ROM_SSIEnable
    // Enables the synchronous serial interface.

ROM_SSIIntClear
    // Clears SSI interrupt sources.

ROM_SSIIntDisable
    // Disables individual SSI interrupt sources.

ROM_SSIIntEnable
    // Enables individual SSI interrupt sources.

ROM_SSIIntStatus
    // Gets the current interrupt status.

ROM_SysCtlADCSpeedGet
    // Gets the sample rate of the ADC.

ROM_SysCtlADCSpeedSet
    // Sets the sample rate of the ADC.

ROM_SysCtlClockGet
    // Gets the processor clock rate.

ROM_SysCtlClockSet
    // Sets the clocking of the device.

ROM_SysCtlDeepSleep
    // Puts the processor into deep-sleep mode.

ROM_SysCtlDelay
    // Provides a small delay.

ROM_SysCtlFlashSizeGet
    // Gets the size of the flash.

ROM_SysCtlGPIOAHBDisable
    // Disables a GPIO peripheral for access from the AHB.
```

ROM_SysCtlGPIOAHBEnable
// Enables a GPIO peripheral for access from the AHB.

ROM_SysCtlI2SMClkSet
// Sets the MCLK frequency provided to the I2S module.

ROM_SysCtlIntClear
// Clears system control interrupt sources.

ROM_SysCtlIntDisable
// Disables individual system control interrupt sources.

ROM_SysCtlIntEnable
// Enables individual system control interrupt sources.

ROM_SysCtlIntStatus
// Gets the current interrupt status.

ROM_SysCtlLDOGet
// Gets the output voltage of the LDO.

ROM_SysCtlLDOSet
// Sets the output voltage of the LDO.

ROM_SysCtlPeripheralClockGating
// Controls peripheral clock gating in sleep and deep-sleep mode.

ROM_SysCtlPeripheralDeepSleepDisable
// Disables a peripheral in deep-sleep mode.

ROM_SysCtlPeripheralDeepSleepEnable
// Enables a peripheral in deep-sleep mode.

ROM_SysCtlPeripheralDisable
// Disables a peripheral.

ROM_SysCtlPeripheralEnable
// Enables a peripheral.

ROM_SysCtlPeripheralPresent
// Determines if a peripheral is present.

ROM_SysCtlPeripheralReset
// Performs a software reset of a peripheral.

ROM_SysCtlPeripheralSleepDisable
// Disables a peripheral in sleep mode.

ROM_SysCtlPeripheralSleepEnable
// Enables a peripheral in sleep mode.

ROM_SysCtlPinPresent
// Determines if a pin is present.

```
ROM_SysCtlReset
    // Resets the device.

ROM_SysCtlResetCauseClear
    // Clears reset reasons.

ROM_SysCtlResetCauseGet
    // Gets the reason for a reset.

ROM_SysCtlSleep
    // Puts the processor into sleep mode.

ROM_SysCtlSRAMSizeGet
    // Gets the size of the SRAM.

ROM_SysCtlUSBPLLDisable
    // Powers down the USB PLL.

ROM_SysCtlUSBPLLEnable
    // Powers up the USB PLL.

ROM_SysTickDisable
    // Disables the SysTick counter.

ROM_SysTickEnable
    // Enables the SysTick counter.

ROM_SysTickIntDisable
    // Disables the SysTick interrupt.

ROM_SysTickIntEnable
    // Enables the SysTick interrupt.

ROM_SysTickPeriodGet
    // Gets the period of the SysTick counter.

ROM_SysTickPeriodSet
    // Sets the period of the SysTick counter.

ROM_SysTickValueGet
    // Gets the current value of the SysTick counter.

ROM_TimerConfigure
    // Configures the timer(s).

ROM_TimerControlEvent
    // Controls the event type.

ROM_TimerControlLevel
    // Controls the output level.

ROM_TimerControlStall
    // Controls the stall handling.
```

ROM_TimerControlTrigger
// Enables or disables the trigger output.

ROM_TimerDisable
// Disables the timer(s).

ROM_TimerEnable
// Enables the timer(s).

ROM_TimerIntClear
// Clears timer interrupt sources.

ROM_TimerIntDisable
// Disables individual timer interrupt sources.

ROM_TimerIntEnable
// Enables individual timer interrupt sources.

ROM_TimerIntStatus
// Gets the current interrupt status.

ROM_TimerLoadGet
// Gets the timer load value.

ROM_TimerLoadSet
// Sets the timer load value.

ROM_TimerMatchGet
// Gets the timer match value.

ROM_TimerMatchSet
// Sets the timer match value.

ROM_TimerPrescaleGet
// Get the timer prescale value.

ROM_TimerPrescaleSet
// Set the timer prescale value.

ROM_TimerRTCDisable
// Disable RTC counting.

ROM_TimerRTCEnable
// Enable RTC counting.

ROM_TimerValueGet
// Gets the current timer value.

ROM_UARTBreakCtl
// Causes a BREAK to be sent.

ROM_UARTBusy
// Determines whether the UART transmitter is busy or not.

ROM_UARTCharGet
// Waits for a character from the specified port.

ROM_UARTCharGetNonBlocking
// Receives a character from the specified port.

ROM_UARTCharPut
// Waits to send a character from the specified port.

ROM_UARTCharPutNonBlocking
// Sends a character to the specified port.

ROM_UARTCharsAvail
// Determines if there are any characters in the receive FIFO.

ROM_UARTConfigGetExpClk
// Gets the current configuration of a UART.

ROM_UARTConfigSetExpClk
// Sets the configuration of a UART.

ROM_UARTDisable
// Disables transmitting and receiving.

ROM_UARTDisableSIR
// Disables SIR (IrDA) mode on the specified UART.

ROM_UARTDMADisable
// Disable UART DMA operation.

ROM_UARTDMAEnable
// Enable UART DMA operation.

ROM_UARTEnable
// Enables transmitting and receiving.

ROM_UARTEnableSIR
// Enables SIR (IrDA) mode on the specified UART.

ROM_UARTFIFODisable
// Disables the transmit and receive FIFOs.

ROM_UARTFIFOEnable
// Enables the transmit and receive FIFOs.

ROM_UARTFIFOLevelGet
// Gets the FIFO level at which interrupts are generated.

ROM_UARTFIFOLevelSet
// Sets the FIFO level at which interrupts are generated.

ROM_UARTIntClear
// Clears UART interrupt sources.

ROM_UARTIntDisable
// Disables individual UART interrupt sources.

ROM_UARTIntEnable
// Enables individual UART interrupt sources.

ROM_UARTIntStatus
// Gets the current interrupt status.

ROM_UARTParityModeGet
// Gets the type of parity currently being used.

ROM_UARTParityModeSet
// Sets the type of parity.

ROM_UARTRxErrorClear
// Clears all reported receiver errors.

ROM_UARTRxErrorGet
// Gets current receiver errors.

ROM_UARTSpaceAvail
// Determines if there is any space in the transmit FIFO.

ROM_UARTTxIntModeGet
// Returns the current operating mode for the UART transmit interrupt.

ROM_UARTTxIntModeSet
// Sets the operating mode for the UART transmit interrupt.

ROM_uDMAChannelAttributeDisable
// Disables attributes of a uDMA channel.

ROM_uDMAChannelAttributeEnable
// Enables attributes of a uDMA channel.

ROM_uDMAChannelAttributeGet
// Gets the enabled attributes of a uDMA channel.

ROM_uDMAChannelControlSet
// Sets the control parameters for a uDMA channel.

ROM_uDMAChannelDisable
// Disables a uDMA channel for operation.

ROM_uDMAChannelEnable
// Enables a uDMA channel for operation.

ROM_uDMAChannelsEnabled
// Checks if a uDMA channel is enabled for operation.

ROM_uDMAChannelModeGet
// Gets the transfer mode for a uDMA channel.

ROM_uDMAChannelRequest
// Requests a uDMA channel to start a transfer.

ROM_uDMAChannelSelectDefault
// Select the default peripheral for a set of uDMA channels.

ROM_uDMAChannelSelectSecondary
// Select the secondary peripheral for a set of uDMA channels.

ROM_uDMAChannelSizeGet
// Gets the current transfer size for a uDMA channel.

ROM_uDMAChannelTransferSet
// Sets the transfer parameters for a uDMA channel.

ROM_uDMAControlBaseGet
// Gets the base address for the channel control table.

ROM_uDMAControlBaseSet
// Sets the base address for the channel control table.

ROM_uDMADisable
// Disables the uDMA controller for use.

ROM_uDMAEnable
// Enables the uDMA controller for use.

ROM_uDMAErrorStatusClear
// Clears the uDMA error interrupt.

ROM_uDMAErrorStatusGet
// Gets the uDMA error status.

ROM_uDMAIntClear
// Clears uDMA interrupt status.

ROM_uDMAIntStatus
// Gets the uDMA controller channel interrupt status.

ROM_UpdateEthernet
// Starts an update over the Ethernet interface.

ROM_UpdateI2C
// Starts an update over the I2C0 interface.

ROM_UpdateSSI
// Starts an update over the SSI0 interface.

ROM_UpdateUART
// Starts an update over the UART0 interface.

ROM_USBDevAddrGet
// Returns the current device address in device mode.

ROM_USBDevAddrSet
// Sets the address in device mode.

ROM_USBDevConnect
// Connects the USB controller to the bus in device mode.

ROM_USBDevDisconnect
// Removes the USB controller from the bus in device mode.

ROM_USBDevEndpointConfig
// Sets the configuration for an endpoint.

ROM_USBDevEndpointConfigGet
// Gets the current configuration for an endpoint.

ROM_USBDevEndpointDataAck
// Acknowledge that data was read from the given endpoint's FIFO in device mode.

ROM_USBDevEndpointStall
// Stalls the specified endpoint in device mode.

ROM_USBDevEndpointStallClear
// Clears the stall condition on the specified endpoint in device mode.

ROM_USBDevEndpointStatusClear
// Clears the status bits in this endpoint in device mode.

ROM_USBEndpointDataAvail
// Determine the number of bytes of data available in a given endpoint's FIFO.

ROM_USBEndpointDataGet
// Retrieves data from the given endpoint's FIFO.

ROM_USBEndpointDataPut
// Puts data into the given endpoint's FIFO.

ROM_USBEndpointDataSend
// Starts the transfer of data from an endpoint's FIFO.

ROM_USBEndpointDataToggleClear
// Sets the Data toggle on an end oint to zero.

ROM_USBEndpointDMAChannel
// Sets the DMA channel to use for a given endpoint.

ROM_USBEndpointDMADisable
// Disable DMA on a given endpoint.

ROM_USBEndpointDMAEnable
// Enable DMA on a given endpoint.

ROM_USBEndpointStatus
// Returns the current status of an endpoint.

ROM_USBFIPOAddrGet
// Returns the absolute FIFO address for a given endpoint.

ROM_USBFIPOConfigGet
// Returns the FIFO configuration for an endpoint.

ROM_USBFIPOConfigSet
// Sets the FIFO configuration for an endpoint.

ROM_USBFIPOFlush
// Forces a flush of an endpoint's FIFO.

ROM_USBFrameNumberGet
// Get the current frame number.

ROM_USBHostAddrGet
// Gets the current functional device address for an endpoint.

ROM_USBHostAddrSet
// Sets the functional address for the device that is connected to an endpoint in host mode.

ROM_USBHostEndpointConfig
// Sets the base configuration for a host endpoint.

ROM_USBHostEndpointDataAck
// Acknowledge that data was read from the given endpoint's FIFO in host mode.

ROM_USBHostEndpointDataToggle
// Sets the value data toggle on an endpoint in host mode.

ROM_USBHostEndpointStatusClear
// Clears the status bits in this endpoint in host mode.

ROM_USBHostHubAddrGet
// Get the current device hub address for this endpoint.

ROM_USBHostHubAddrSet
// Set the hub address for the device that is connected to an endpoint.

ROM_USBHostPwrDisable
// Disables the external power pin.

ROM_USBHostPwrEnable
// Enables the external power pin.

ROM_USBHostPwrFaultConfig
// Sets the configuration for USB power fault.

ROM_USBHostPwrFaultDisable
// Disables power fault detection.

ROM_USBHostPwrFaultEnable
// Enables power fault detection.

ROM_USBHostRequestIN
// Schedules a request for an IN transaction on an endpoint in host mode.

ROM_USBHostRequestStatus
// Issues a request for a status IN transaction on endpoint zero.

ROM_USBHostReset
// Handles the USB bus reset condition.

ROM_USBHostResume
// Handles the USB bus resume condition.

ROM_USBHostSpeedGet
// Returns the current speed of the USB device connected.

ROM_USBHostSuspend
// Puts the USB bus in a suspended state.

ROM_USBIntDisable
// Disables the sources for USB interrupts.

ROM_USBIntEnable
// Enables the sources for USB interrupts.

ROM_USBIntStatus
// Returns the status of the USB interrupts.

ROM_USBModeGet
// Returns the current operating mode of the controller.

ROM_USBOTGHostRequest
// This function will enable host negotiation protocol when in device mode.

ROM_WatchdogEnable
// Enables the watchdog timer.

ROM_WatchdogIntClear
// Clears the watchdog timer interrupt.

ROM_WatchdogIntEnable
// Enables the watchdog timer interrupt.

ROM_WatchdogIntStatus
// Gets the current watchdog timer interrupt status.

ROM_WatchdogLock
// Enables the watchdog timer lock mechanism.

ROM_WatchdogLockState
// Gets the state of the watchdog timer lock mechanism.

ROM_WatchdogReloadGet
// Gets the watchdog timer reload value.

ROM_WatchdogReloadSet
// Sets the watchdog timer reload value.

ROM_WatchdogResetDisable
// Disables the watchdog timer reset.

ROM_WatchdogResetEnable
// Enables the watchdog timer reset.

ROM_WatchdogRunning
// Determines if the watchdog timer is enabled.

ROM_WatchdogStallDisable
// Disables stalling of the watchdog timer during debug events.

ROM_WatchdogStallEnable
// Enables stalling of the watchdog timer during debug events.

ROM_WatchdogUnlock
// Disables the watchdog timer lock mechanism.

ROM_WatchdogValueGet
// Gets the current watchdog timer value.

C Advance Encryption Standard and Cyclic Redundancy Check Software in ROM

AES and CRC software is available in the integrated ROM of the LM3S9B90 microcontroller. For more information on this software, see *Stellaris® ROM User's Guide*.

C.1 Advanced Encryption Standard Software

The Advanced Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government. It is a strong encryption method with reasonable performance and size. AES is fast in both hardware and software, is fairly easy to implement, and requires little memory. AES is ideal for applications that can use pre-arranged keys, such as setup during manufacturing or configuration.

C.2 Cyclic Redundancy Check Software

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

D 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 | |
| System Control | | | | | | | | | | | | | | | | |
| Base 0x400F.E000 | | | | | | | | | | | | | | | | |
| DID0, type RO, offset 0x000, reset - | | | | | | | | | | | | | | | | |
| VER | | | | | | | | CLASS | | | | | | | | |
| MAJOR | | | | | | | | MINOR | | | | | | | | |
| PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | BORIOR | |
| RIS, type RO, offset 0x050, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | MOSCPUPRIS | USBPLLRIS | PLLRIS | | | | | | BORRIS |
| IMC, type R/W, offset 0x054, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | MOSCPUPIM | USBPLLIIM | PLLIIM | | | | | | BORIM |
| MISC, type R/W1C, offset 0x058, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | MOSCPUPMS | USBPLLMIS | PLLMIS | | | | | | BORMIS |
| RESC, type R/W, offset 0x05C, reset - | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | MOSCFAIL | |
| RCC, type R/W, offset 0x060, reset 0x0780.3AD1 | | | | | | | | | | | | | | | | |
| | | | | ACG | SYSDIV | | | | USESYS | | | | | | | |
| PWRDN | | | | BYPASS | XTAL | | | | OSCSRC | | | IOSCDIS | | MOSCDIS | | |
| PLLCFG, type RO, offset 0x064, reset - | | | | | | | | | | | | | | | | |
| F | | | | | | | | R | | | | | | | | |
| GPIOHBCTL, type R/W, offset 0x06C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | PORTJ | PORTH | PORTG | PORTF | PORTE | PORTD | PORTC | PORTB | PORTA |
| RCC2, type R/W, offset 0x070, reset 0x0780.6810 | | | | | | | | | | | | | | | | |
| USERCC2 | USEFRACT | SYSDIV2 | | | | | | FRACT | | | | | | | | |
| USBPWRDN | | PWRDN2 | BYPASS2 | | | | | OSCSRC2 | | | | | | | | |
| MOSCCTL, type R/W, offset 0x07C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | CVAL | |
| DSLCLKCFG, type R/W, offset 0x144, reset 0x0780.0000 | | | | | | | | | | | | | | | | |
| DSDIVORIDE | | | | | | | | DSOSCSRC | | | | | | | | |
| DSFLASHCFG, type R/W, offset 0x14C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | SHDWN | |
| PIOSCCAL, type R/W, offset 0x150, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| UTEN | | | | | | | | | | | | | | | | |
| | | | | CAL | UPDATE | | | | | UT | | | | | | |
| PIOSCSTAT, type RO, offset 0x154, reset 0x0000.0040 | | | | | | | | | | | | | | | | |
| | | | | | | | | RESULT | | | | DT | | | | |
| | | | | | | | | | | | | CT | | | | |
| I2SMCLKCFG, type R/W, offset 0x170, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| RXEN | | | | | | | | RXI | | | | RXF | | | | |
| TXEN | | | | | | | | TXI | | | | TXF | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|----|-----------|----|------------|----|-----------|----|------------|----|-----------|----|----------|----|----------|----|----------|--|----------|--|----------|--|----------|--|----------|--|----------|--|----------|--|----------|--|
| 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 | | | | | | | | | | | | | | | | |
| DD1, type RO, offset 0x004, reset - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| VER | | | | FAM | | | | PARTNO | | | | | | | | | | | | | | | | | | | | | | | |
| PINCOUNT | | | | TEMP | | | | PKG | | ROHS | | QUAL | | | | | | | | | | | | | | | | | | | |
| DC0, type RO, offset 0x008, reset 0x017F.007F | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SRAMSZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FLASHSZ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DC1, type RO, offset 0x010, reset - | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| WDT1 | | | | CAN1 | | CAN0 | | MPU | | | | HIB | | TEMPSNS | | PLL | | WDT0 | | SWO | | SWD | | JTAG | | | | | | | |
| MINSYSDIV | | | | MAXADC1SPD | | | | MAXADC0SPD | | | | SS1 | | SSIO | | TIMER3 | | TIMER2 | | TIMER1 | | TIMER0 | | | | | | | | | |
| I2C1 | | | | I2S0 | | COMP2 | | COMP1 | | COMP0 | | SS1 | | SSIO | | UART2 | | UART1 | | UART0 | | | | | | | | | | | |
| DC2, type RO, offset 0x014, reset 0x570F.5037 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EPI0 | | | | I2S0 | | COMP2 | | COMP1 | | COMP0 | | SS1 | | SSIO | | TIMER3 | | TIMER2 | | TIMER1 | | TIMER0 | | | | | | | | | |
| I2C1 | | | | I2C0 | | C10 | | C1PLUS | | C1MINUS | | C00 | | C0PLUS | | C0MINUS | | C0PLUS | | C0MINUS | | C0PLUS | | C0MINUS | | | | | | | |
| DC3, type RO, offset 0x018, reset 0xBFFF.7FC0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 32KHZ | | | | CCP5 | | CCP4 | | CCP3 | | CCP2 | | CCP1 | | CCP0 | | ADC0AIN7 | | ADC0AIN6 | | ADC0AIN5 | | ADC0AIN4 | | ADC0AIN3 | | ADC0AIN2 | | ADC0AIN1 | | ADC0AIN0 | |
| C20 | | | | C2PLUS | | C2MINUS | | C10 | | C1PLUS | | C1MINUS | | C00 | | C0PLUS | | C0MINUS | | C0PLUS | | C0MINUS | | C0PLUS | | C0MINUS | | C0PLUS | | C0MINUS | |
| DC4, type RO, offset 0x01C, reset 0x5004.F1FF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EPHY0 | | | | EMAC0 | | | | GPIOJ | | | | GPIOH | | GPIOG | | GPIOF | | GPIOE | | GPIOD | | GPIOC | | GPIOB | | GPIOA | | | | | |
| CCP7 | | | | CCP6 | | UDMA | | ROM | | GPIOJ | | GPIOH | | GPIOG | | GPIOF | | GPIOE | | GPIOD | | GPIOC | | GPIOB | | GPIOA | | | | | |
| DC5, type RO, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DC6, type RO, offset 0x024, reset 0x0000.0013 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| USB0PHY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| USB0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DC7, type RO, offset 0x028, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DMACH30 | | DMACH29 | | DMACH28 | | DMACH27 | | DMACH26 | | DMACH25 | | DMACH24 | | DMACH23 | | DMACH22 | | DMACH21 | | DMACH20 | | DMACH19 | | DMACH18 | | DMACH17 | | DMACH16 | | | |
| DMACH15 | | DMACH14 | | DMACH13 | | DMACH12 | | DMACH11 | | DMACH10 | | DMACH9 | | DMACH8 | | DMACH7 | | DMACH6 | | DMACH5 | | DMACH4 | | DMACH3 | | DMACH2 | | DMACH1 | | DMACH0 | |
| DC8, type RO, offset 0x02C, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ADC1AIN15 | | ADC1AIN14 | | ADC1AIN13 | | ADC1AIN12 | | ADC1AIN11 | | ADC1AIN10 | | ADC1AIN9 | | ADC1AIN8 | | ADC1AIN7 | | ADC1AIN6 | | ADC1AIN5 | | ADC1AIN4 | | ADC1AIN3 | | ADC1AIN2 | | ADC1AIN1 | | ADC1AIN0 | |
| ADC0AIN15 | | ADC0AIN14 | | ADC0AIN13 | | ADC0AIN12 | | ADC0AIN11 | | ADC0AIN10 | | ADC0AIN9 | | ADC0AIN8 | | ADC0AIN7 | | ADC0AIN6 | | ADC0AIN5 | | ADC0AIN4 | | ADC0AIN3 | | ADC0AIN2 | | ADC0AIN1 | | ADC0AIN0 | |
| DC9, type RO, offset 0x190, reset 0x00FF.00FF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ADC1DC7 | | | | | | | | ADC1DC6 | | ADC1DC5 | | ADC1DC4 | | ADC1DC3 | | ADC1DC2 | | ADC1DC1 | | ADC1DC0 | | | | | | | | | | | |
| ADC0DC7 | | | | | | | | ADC0DC6 | | ADC0DC5 | | ADC0DC4 | | ADC0DC3 | | ADC0DC2 | | ADC0DC1 | | ADC0DC0 | | | | | | | | | | | |
| NVMSTAT, type RO, offset 0x1A0, reset 0x0000.0001 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FWB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| RCGC0, type R/W, offset 0x100, reset 0x00000040 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| WDT1 | | | | CAN1 | | CAN0 | | HIB | | | | WDT0 | | ADC1 | | ADC0 | | | | | | | | | | | | | | | |
| MAXADC1SPD | | | | MAXADC0SPD | | | | HIB | | | | WDT0 | | ADC1 | | ADC0 | | | | | | | | | | | | | | | |
| SCGC0, type R/W, offset 0x110, reset 0x00000040 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| WDT1 | | | | CAN1 | | CAN0 | | HIB | | | | WDT0 | | ADC1 | | ADC0 | | | | | | | | | | | | | | | |
| MAXADC1SPD | | | | MAXADC0SPD | | | | HIB | | | | WDT0 | | ADC1 | | ADC0 | | | | | | | | | | | | | | | |
| DCGC0, type R/W, offset 0x120, reset 0x00000040 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| WDT1 | | | | CAN1 | | CAN0 | | HIB | | | | WDT0 | | ADC1 | | ADC0 | | | | | | | | | | | | | | | |
| MAXADC1SPD | | | | MAXADC0SPD | | | | HIB | | | | WDT0 | | ADC1 | | ADC0 | | | | | | | | | | | | | | | |
| RCGC1, type R/W, offset 0x104, reset 0x00000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EPI0 | | | | I2S0 | | COMP2 | | COMP1 | | COMP0 | | SS1 | | SSIO | | TIMER3 | | TIMER2 | | TIMER1 | | TIMER0 | | | | | | | | | |
| I2C1 | | | | I2C0 | | COMP2 | | COMP1 | | COMP0 | | SS1 | | SSIO | | TIMER3 | | TIMER2 | | TIMER1 | | TIMER0 | | | | | | | | | |
| SCGC1, type R/W, offset 0x114, reset 0x00000000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EPI0 | | | | I2S0 | | COMP2 | | COMP1 | | COMP0 | | SS1 | | SSIO | | TIMER3 | | TIMER2 | | TIMER1 | | TIMER0 | | | | | | | | | |
| I2C1 | | | | I2C0 | | COMP2 | | COMP1 | | COMP0 | | SS1 | | SSIO | | TIMER3 | | TIMER2 | | TIMER1 | | TIMER0 | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|--|-------|------|-------|----|-------|--------|--------|---------|----------|--------|--------|--------|--------|--------|--------|-------|
| 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 | |
| DCGC1, type R/W, offset 0x124, reset 0x00000000 | | | | | | | | | | | | | | | | |
| | EPI0 | | I2S0 | | COMP2 | COMP1 | COMP0 | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 | |
| | I2C1 | | I2C0 | | | | | | | SSI1 | SSI0 | | UART2 | UART1 | UART0 | |
| RCGC2, type R/W, offset 0x108, reset 0x00000000 | | | | | | | | | | | | | | | | |
| | EPHY0 | | EMAC0 | | | | | | | | | | | | USB0 | |
| | | UDMA | | | | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| SCGC2, type R/W, offset 0x118, reset 0x00000000 | | | | | | | | | | | | | | | | |
| | EPHY0 | | EMAC0 | | | | | | | | | | | | USB0 | |
| | | UDMA | | | | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| DCGC2, type R/W, offset 0x128, reset 0x00000000 | | | | | | | | | | | | | | | | |
| | EPHY0 | | EMAC0 | | | | | | | | | | | | USB0 | |
| | | UDMA | | | | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| SRCR0, type R/W, offset 0x040, reset 0x00000000 | | | | | | | | | | | | | | | | |
| | | | WDT1 | | | CAN1 | CAN0 | | | | | | | | ADC1 | ADC0 |
| | | | | | | | | | HIB | | | | WDT0 | | | |
| SRCR1, type R/W, offset 0x044, reset 0x00000000 | | | | | | | | | | | | | | | | |
| | EPI0 | | I2S0 | | COMP2 | COMP1 | COMP0 | | | | | TIMER3 | TIMER2 | TIMER1 | TIMER0 | |
| | I2C1 | | I2C0 | | | | | | | SSI1 | SSI0 | | UART2 | UART1 | UART0 | |
| SRCR2, type R/W, offset 0x048, reset 0x00000000 | | | | | | | | | | | | | | | | |
| | EPHY0 | | EMAC0 | | | | | | | | | | | | USB0 | |
| | | UDMA | | | | | | GPIOJ | GPIOH | GPIOG | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA |
| Hibernation Module | | | | | | | | | | | | | | | | |
| Base 0x400F.C000 | | | | | | | | | | | | | | | | |
| HIBRTCC, type RO, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| RTCC | | | | | | | | | | | | | | | | |
| RTCC | | | | | | | | | | | | | | | | |
| HIBRTCM0, type R/W, offset 0x004, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| RTCM0 | | | | | | | | | | | | | | | | |
| RTCM0 | | | | | | | | | | | | | | | | |
| HIBRTCM1, type R/W, offset 0x008, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| RTCM1 | | | | | | | | | | | | | | | | |
| RTCM1 | | | | | | | | | | | | | | | | |
| HIBRTCLD, type R/W, offset 0x00C, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| RTCLD | | | | | | | | | | | | | | | | |
| RTCLD | | | | | | | | | | | | | | | | |
| HIBCTL, type R/W, offset 0x010, reset 0x8000.0000 | | | | | | | | | | | | | | | | |
| WRC | | | | | | | | | | | | | | | | |
| | | | | | | VDD3ON | VABORT | CLK32EN | LOWBATEN | PINWEN | RTCWEN | CLKSEL | HIBREQ | RTCCEN | | |
| HIBIM, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCAL1 | RTCAL0 | |
| HIBRIS, type RO, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCAL1 | RTCAL0 | |
| HIBMIS, type RO, offset 0x01C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCAL1 | RTCAL0 | |
| HIBIC, type R/W1C, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | EXTW | LOWBAT | RTCAL1 | RTCAL0 | |
| HIBRTCT, type R/W, offset 0x024, reset 0x0000.7FFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| TRIM | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 |
| HIBDATA, type R/W, offset 0x030-0x12C, reset - | | | | | | | | | | | | | | | |
| RTD | | | | | | | | | | | | | | | |
| RTD | | | | | | | | | | | | | | | |
| Internal Memory | | | | | | | | | | | | | | | |
| Flash Memory Registers (Flash Control Offset) | | | | | | | | | | | | | | | |
| Base 0x400F.D000 | | | | | | | | | | | | | | | |
| FMA, type R/W, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| OFFSET | | | | | | | | | | | | | | | |
| OFFSET | | | | | | | | | | | | | | | |
| FMD, type R/W, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| FMC, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WRKEY | | | | | | | | | | | | | | | |
| COMT MERASE ERASE WRITE | | | | | | | | | | | | | | | |
| FCRIS, type RO, offset 0x00C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| PRIS ARIS | | | | | | | | | | | | | | | |
| FCIM, type R/W, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| PMASK AMASK | | | | | | | | | | | | | | | |
| FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| PMISC AMISC | | | | | | | | | | | | | | | |
| FMC2, type R/W, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WRKEY | | | | | | | | | | | | | | | |
| WRBUF | | | | | | | | | | | | | | | |
| FWBVAL, type R/W, offset 0x030, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| FWB[n] | | | | | | | | | | | | | | | |
| FWB[n] | | | | | | | | | | | | | | | |
| FWBn, type R/W, offset 0x100 - 0x13C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| Internal Memory | | | | | | | | | | | | | | | |
| Memory Registers (System Control Offset) | | | | | | | | | | | | | | | |
| Base 0x400F.E000 | | | | | | | | | | | | | | | |
| RMCTL, type R/W1C, offset 0x0F0, reset - | | | | | | | | | | | | | | | |
| BA | | | | | | | | | | | | | | | |
| RMVER, type RO, offset 0x0F4, reset 0x0202.5400 | | | | | | | | | | | | | | | |
| CONT SIZE | | | | | | | | | | | | | | | |
| VER REV | | | | | | | | | | | | | | | |
| 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 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | |
|--|----|----------|----|--------|----|---------|----|----|----|-------------|----|----------|----|-------------|----|--------|--|
| 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 | | |
| USER_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | |
| NW | | DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | |
| USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | |
| NW | | DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | |
| USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | |
| NW | | DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | |
| USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | |
| NW | | DATA | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | |
| FMPRE1, type R/W, offset 0x204, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | READ_ENABLE | | | |
| | | | | | | | | | | | | | | READ_ENABLE | | | |
| FMPRE2, type R/W, offset 0x208, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | READ_ENABLE | | | |
| | | | | | | | | | | | | | | READ_ENABLE | | | |
| 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 | | | |
| FMPPE3, type R/W, offset 0x40C, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | PROG_ENABLE | | | |
| | | | | | | | | | | | | | | PROG_ENABLE | | | |
| Micro Direct Memory Access (μDMA) | | | | | | | | | | | | | | | | | |
| μDMA Channel Control Structure (Offset from Channel Control Table Base) | | | | | | | | | | | | | | | | | |
| Base n/a | | | | | | | | | | | | | | | | | |
| DMASRCNDP, type R/W, offset 0x000, reset - | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | ADDR | | | |
| | | | | | | | | | | | | | | ADDR | | | |
| DMADSTENDP, type R/W, offset 0x004, reset - | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | ADDR | | | |
| | | | | | | | | | | | | | | ADDR | | | |
| DMACHCTL, type R/W, offset 0x008, reset - | | | | | | | | | | | | | | | | | |
| DSTINC | | DSTSIZE | | SRCINC | | SRCSIZE | | | | | | | | ARBSIZE | | | |
| ARBSIZE | | XFERSIZE | | | | | | | | NXTUSEBURST | | XFERMODE | | | | | |
| Micro Direct Memory Access (μDMA) | | | | | | | | | | | | | | | | | |
| μDMA Registers (Offset from μDMA Base Address) | | | | | | | | | | | | | | | | | |
| Base 0x400F.F000 | | | | | | | | | | | | | | | | | |
| DMASTAT, type RO, offset 0x000, reset 0x001F.0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | DMACHANS | | | |
| | | | | | | | | | | | | | | STATE | | MASTEN | |
| DMACFG, type WO, offset 0x004, reset - | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | MASTEN | |

| | | | | | | | | | | | | | | | |
|--|----|----|----|----|----|----|----|------------|----|----|----|----|----|----|--------|
| 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 |
| DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| ADDR | | | | | | | | | | | | | | | |
| DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| DMAWAITSTAT, type RO, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | WAITREQ[n] | | | | | | | |
| | | | | | | | | WAITREQ[n] | | | | | | | |
| DMASWREQ, type WO, offset 0x014, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | SWREQ[n] | | | | | | | |
| | | | | | | | | SWREQ[n] | | | | | | | |
| DMAUSEBURSTSET, type R/W, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| DMAUSEBURSTCLR, type WO, offset 0x01C, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| DMAREQMASKSET, type R/W, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| DMAREQMASKCLR, type WO, offset 0x024, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| DMAENASET, type R/W, offset 0x028, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| DMAENACL, type WO, offset 0x02C, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| DMAALTSET, type R/W, offset 0x030, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| DMAALTCLR, type WO, offset 0x034, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| DMAPRIOSET, type R/W, offset 0x038, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| | | | | | | | | SET[n] | | | | | | | |
| DMAPRIOCLR, type WO, offset 0x03C, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| | | | | | | | | CLR[n] | | | | | | | |
| DMAERRCLR, type R/W, offset 0x04C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | ERRCLR |
| DMACHALT, type R/W, offset 0x500, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | CHALT[n] | | | | | | | |
| | | | | | | | | CHALT[n] | | | | | | | |
| DMACHIS, type R/W1C, offset 0x504, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | CHIS[n] | | | | | | | |
| | | | | | | | | CHIS[n] | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|----|----|----|----|----|----|----|----|----|----|----|------|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAPeriphID0, type RO, offset 0xFE0, reset 0x0000.0030 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID0 | | | |
| DMAPeriphID1, type RO, offset 0xFE4, reset 0x0000.00B2 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID1 | | | |
| DMAPeriphID2, type RO, offset 0xFE8, reset 0x0000.000B | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID2 | | | |
| DMAPeriphID3, type RO, offset 0xFEC, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID3 | | | |
| DMAPeriphID4, type RO, offset 0xFD0, reset 0x0000.0004 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID4 | | | |
| DMACellID0, type RO, offset 0xFF0, reset 0x0000.000D | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID0 | | | |
| DMACellID1, type RO, offset 0xFF4, reset 0x0000.00F0 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID1 | | | |
| DMACellID2, type RO, offset 0xFF8, reset 0x0000.0005 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID2 | | | |
| DMACellID3, type RO, offset 0xFFC, reset 0x0000.00B1 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID3 | | | |
| General-Purpose Input/Outputs (GPIOs) | | | | | | | | | | | | | | | |
| GPIO Port A (APB) base: 0x4000.4000 | | | | | | | | | | | | | | | |
| GPIO Port A (AHB) base: 0x4005.8000 | | | | | | | | | | | | | | | |
| GPIO Port B (APB) base: 0x4000.5000 | | | | | | | | | | | | | | | |
| GPIO Port B (AHB) base: 0x4005.9000 | | | | | | | | | | | | | | | |
| GPIO Port C (APB) base: 0x4000.6000 | | | | | | | | | | | | | | | |
| GPIO Port C (AHB) base: 0x4005.A000 | | | | | | | | | | | | | | | |
| GPIO Port D (APB) base: 0x4000.7000 | | | | | | | | | | | | | | | |
| GPIO Port D (AHB) base: 0x4005.B000 | | | | | | | | | | | | | | | |
| GPIO Port E (APB) base: 0x4002.4000 | | | | | | | | | | | | | | | |
| GPIO Port E (AHB) base: 0x4005.C000 | | | | | | | | | | | | | | | |
| GPIO Port F (APB) base: 0x4002.5000 | | | | | | | | | | | | | | | |
| GPIO Port F (AHB) base: 0x4005.D000 | | | | | | | | | | | | | | | |
| GPIO Port G (APB) base: 0x4002.6000 | | | | | | | | | | | | | | | |
| GPIO Port G (AHB) base: 0x4005.E000 | | | | | | | | | | | | | | | |
| GPIO Port H (APB) base: 0x4002.7000 | | | | | | | | | | | | | | | |
| GPIO Port H (AHB) base: 0x4005.F000 | | | | | | | | | | | | | | | |
| GPIO Port J (APB) base: 0x4003.D000 | | | | | | | | | | | | | | | |
| GPIO Port J (AHB) base: 0x4006.0000 | | | | | | | | | | | | | | | |
| GPIODATA, type R/W, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DATA | | | |
| GPIODIR, type R/W, offset 0x400, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DIR | | | |
| GPIOIS, type R/W, offset 0x404, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | IS | | | |
| GPIOIBE, type R/W, offset 0x408, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | IBE | | | |

| 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 |
| GPIOIEV, type R/W, offset 0x40C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | IEV | | | |
| GPIOIM, type R/W, offset 0x410, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | IME | | | |
| GPIORIS, type RO, offset 0x414, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | RIS | | | |
| GPIOMIS, type RO, offset 0x418, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | MIS | | | |
| GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | IC | | | |
| GPIOAFSEL, type R/W, offset 0x420, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | AFSEL | | | |
| GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DRV2 | | | |
| GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DRV4 | | | |
| GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DRV8 | | | |
| GPIOODR, type R/W, offset 0x50C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | ODE | | | |
| GPIOPUR, type R/W, offset 0x510, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PUE | | | |
| GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PDE | | | |
| GPIOSLR, type R/W, offset 0x518, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SRL | | | |
| GPIODEN, type R/W, offset 0x51C, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DEN | | | |
| GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | LOCK | | | |
| | | | | | | | | | | | | LOCK | | | |
| GPIOCR, type -, offset 0x524, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CR | | | |
| GPIOAMSEL, type R/W, offset 0x528, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | GPIOAMSEL | | | |

| | | | | | | | | | | | | | | | |
|--|----|----|----|------|----|----|----|-------|----|-------|----|--------|----|--------|----|
| 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 |
| GPIOPCTL, type R/W, offset 0x52C, reset - | | | | | | | | | | | | | | | |
| PMC7 | | | | PMC6 | | | | PMC5 | | | | PMC4 | | | |
| PMC3 | | | | PMC2 | | | | PMC1 | | | | PMC0 | | | |
| GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID4 | | | |
| GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID5 | | | |
| GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID6 | | | |
| GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID7 | | | |
| GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID0 | | | |
| GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID1 | | | |
| GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID2 | | | |
| GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID3 | | | |
| GPIOPCellID0, type RO, offset 0xFF0, reset 0x0000.000D | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID0 | | | |
| GPIOPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID1 | | | |
| GPIOPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID2 | | | |
| GPIOPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID3 | | | |
| External Peripheral Interface (EPI) | | | | | | | | | | | | | | | |
| Base 0x400D.0000 | | | | | | | | | | | | | | | |
| EPICFG, type R/W, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | BLKEN | | MODE | |
| EPIBAUD, type R/W, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| COUNT1 | | | | | | | | | | | | | | | |
| COUNT0 | | | | | | | | | | | | | | | |
| EPISDRAMCFG, type R/W, offset 0x010, reset 0x42EE.0000 | | | | | | | | | | | | | | | |
| FREQ | | | | RFSH | | | | | | | | | | | |
| SLEEP | | | | | | | | SIZE | | | | | | | |
| EPIHB8CFG, type R/W, offset 0x010, reset 0x0000.FF00 | | | | | | | | | | | | | | | |
| MAXWAIT | | | | | | | | XFFEN | | XFEEN | | WRHIGH | | RDHIGH | |
| | | | | | | | | WRWS | | RDWS | | MODE | | | |

| | | | | | | | | | | | | | | | | |
|--|---------|-------|----|--------|-------|--------|---------|---------|-------|---------|--------|-------|--------|--------|----|--|
| 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 | |
| EPIHB16CFG, type R/W, offset 0x010, reset 0x0000.FF00 | | | | | | | | | | | | | | | | |
| MAXWAIT | | | | | | | | XFFEN | XFEEN | WRHIGH | RDHIGH | | | | | |
| | | | | | | | | WRWS | | RDWS | | BSEL | | MODE | | |
| EPIGPCFG, type R/W, offset 0x010, reset 0x0000.FF00 | | | | | | | | | | | | | | | | |
| CLKPIN | CLKGATE | RDYEN | | FRMPIN | FRM50 | FRMCNT | | | | RW | | | WR2CYC | RD2CYC | | |
| MAXWAIT | | | | | | | | ASIZE | | | | DSIZE | | | | |
| EPISDRAMCFG2, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| RCM | | | | | | | | | | | | | | | | |
| EPIHB8CFG2, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| WORD | | | | CSBAUD | | CSCFG | | | | | | | | | | |
| EPIHB16CFG2, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| WORD | | | | CSBAUD | | CSCFG | | | | | | | | | | |
| EPIGPCFG2, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| WORD | | | | | | | | | | | | | | | | |
| EPIADDRMAP, type R/W, offset 0x01C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | EPSZ | | EPADR | | ERSZ | | ERADR | | |
| EPIRSIZE0, type R/W, offset 0x020, reset 0x0000.0003 | | | | | | | | | | | | | | | | |
| SIZE | | | | | | | | | | | | | | | | |
| EPIRSIZE1, type R/W, offset 0x030, reset 0x0000.0003 | | | | | | | | | | | | | | | | |
| SIZE | | | | | | | | | | | | | | | | |
| EPIRADDR0, type R/W, offset 0x024, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | | |
| EPIRADDR1, type R/W, offset 0x034, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | | |
| ADDR | | | | | | | | | | | | | | | | |
| EPIRPSTD0, type R/W, offset 0x028, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | POSTCNT | | | | | | | | |
| POSTCNT | | | | | | | | | | | | | | | | |
| EPIRPSTD1, type R/W, offset 0x038, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | POSTCNT | | | | | | | | |
| POSTCNT | | | | | | | | | | | | | | | | |
| EPISTAT, type RO, offset 0x060, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CELOW | | XFFULL | XFEMPTY | INITSEQ | WBUSY | NBRBUSY | ACTIVE | | | | | |
| EPIRFIFOCNT, type RO, offset 0x06C, reset - | | | | | | | | | | | | | | | | |
| COUNT | | | | | | | | | | | | | | | | |
| EPIREADFIFO, type RO, offset 0x070, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | DATA | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | |
| EPIREADFIFO1, type RO, offset 0x074, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | DATA | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | |
|---|----|-------|----|---------|----|---------|----|---------|----|--------|----|---------|----|--------|----|---------|--|---------|--|------|--|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | |
| EPIREADFIFO2, type RO, offset 0x078, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| EPIREADFIFO3, type RO, offset 0x07C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| EPIREADFIFO4, type RO, offset 0x080, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| EPIREADFIFO5, type RO, offset 0x084, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| EPIREADFIFO6, type RO, offset 0x088, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| EPIREADFIFO7, type RO, offset 0x08C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | | | | | | | |
| EPIFIFOLVL, type R/W, offset 0x200, reset 0x0000.0033 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | WFERR | | RSERR | | | | | | | |
| WRFIFO | | | | | | | | | | RDFIFO | | | | | | | | | | | |
| EPIWFIFOCNT, type RO, offset 0x204, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| WTAV | | | | | | | | | | | | | | | | | | | | | |
| EPIIM, type R/W, offset 0x210, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | WRIM | | RDIM | | ERRIM | | | | | |
| EPIRIS, type RO, offset 0x214, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | WRRIS | | RDRIS | | ERRRIS | | | | | |
| EPIMIS, type RO, offset 0x218, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | WRMIS | | RDMIS | | ERRMIS | | | | | |
| EPIESC, type R/W1C, offset 0x21C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | WTFULL | | RSTALL | | TOUT | | | | | |
| 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 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | GPTMCFG | | | | | | | | | |
| GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | TASNAPS | | TAWOT | | TAMIE | | TACDIR | | TAAMS | | TACMR | | TAMR | |
| GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | TBSNAPS | | TBWOT | | TBMIE | | TBCDIR | | TBAMS | | TBCMR | | TBMR | |
| GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | | |
| TBPWML | | TBOTE | | TBEVENT | | TBSTALL | | TBMEN | | TAPWML | | TAOTE | | RTCEN | | TAEVENT | | TASTALL | | TAEN | |

| | | | | | | | | | | | | | | | | |
|--|----|----|----|----|---------|---------|---------|---------|----|----|----|---------|---------|---------|---------|----------|
| 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 | |
| GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | TBMIM | CBEIM | CBMIM | TBTOIM | | | | TAMIM | RTCIM | CAEIM | CAMIM | TATOIM |
| GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | TBMRIS | CBERIS | CBMRIS | TBTORIS | | | | TAMRIS | RTCRIIS | CAERIS | CAMRIS | TATORIS |
| GPTMMIS, type RO, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | TBMMIS | CBEMIS | CBMMIS | TBTOMIS | | | | TAMMIS | RTCMIS | CAEMIS | CAMMIS | TATOMIS |
| GPTMICR, type W1C, offset 0x024, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | TBMCINT | CBECINT | CBMCINT | TBTCINT | | | | TAMCINT | RTCCINT | CAECINT | CAMCINT | TATOCINT |
| GPTMTAILR, type R/W, offset 0x028, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TAILRH |
| | | | | | | | | | | | | | | | | TAILRL |
| GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TBILRL |
| GPTMTAMATCHR, type R/W, offset 0x030, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TAMRH |
| | | | | | | | | | | | | | | | | TAMRL |
| GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TBMRL |
| GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TAPSR |
| GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TBPSR |
| GPTMTAR, type RO, offset 0x048, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TARH |
| | | | | | | | | | | | | | | | | TARL |
| GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TBRL |
| GPTMTAV, type RO, offset 0x050, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TAVH |
| | | | | | | | | | | | | | | | | TAVL |
| GPTMTBV, type RO, offset 0x054, reset 0x0000.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | TBVL |
| Watchdog Timers | | | | | | | | | | | | | | | | |
| WDT0 base: 0x4000.0000 | | | | | | | | | | | | | | | | |
| WDT1 base: 0x4000.1000 | | | | | | | | | | | | | | | | |
| WDTLOAD, type R/W, offset 0x000, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | WDTLOAD |
| | | | | | | | | | | | | | | | | WDTLOAD |
| WDTVALUE, type RO, offset 0x004, reset 0xFFFF.FFFF | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | WDTVALUE |
| | | | | | | | | | | | | | | | | WDTVALUE |

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|----|----|----|----|-------|----|-------|----|
| 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 |
| WDCTL, type R/W, offset 0x008, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1) | | | | | | | | | | | | | | | |
| WRC | | | | | | | | | | | | RESEN | | INTEN | |
| WDTCR, type WO, offset 0x00C, reset - | | | | | | | | | | | | | | | |
| WDTCR | | | | | | | | | | | | | | | |
| WDTCR | | | | | | | | | | | | | | | |
| WDTRIS, type RO, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTRIS | | | | | | | | | | | | | | | |
| WDTMIS, type RO, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTMIS | | | | | | | | | | | | | | | |
| WDTEST, type R/W, offset 0x418, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTEST | | | | | | | | | | | | | | | |
| WDTEST | | | | | | | | | | | | | | | |
| WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTLOCK | | | | | | | | | | | | | | | |
| WDTLOCK | | | | | | | | | | | | | | | |
| WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTPeriphID4 | | | | | | | | | | | | | | | |
| WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTPeriphID5 | | | | | | | | | | | | | | | |
| WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTPeriphID6 | | | | | | | | | | | | | | | |
| WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| WDTPeriphID7 | | | | | | | | | | | | | | | |
| WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005 | | | | | | | | | | | | | | | |
| WDTPeriphID0 | | | | | | | | | | | | | | | |
| WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018 | | | | | | | | | | | | | | | |
| WDTPeriphID1 | | | | | | | | | | | | | | | |
| WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 | | | | | | | | | | | | | | | |
| WDTPeriphID2 | | | | | | | | | | | | | | | |
| WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| WDTPeriphID3 | | | | | | | | | | | | | | | |
| WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D | | | | | | | | | | | | | | | |
| WDTPCellID0 | | | | | | | | | | | | | | | |
| WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 | | | | | | | | | | | | | | | |
| WDTPCellID1 | | | | | | | | | | | | | | | |
| WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0006 | | | | | | | | | | | | | | | |
| WDTPCellID2 | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|--|-----|------|----|----------|-----|------|----|--------|--------|--------|--------|---------|---------|---------|---------|
| 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 |
| WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 | | | | | | | | | | | | | | | |
| CID3 | | | | | | | | | | | | | | | |
| Analog-to-Digital Converter (ADC) ADC0 base: 0x4003.8000 ADC1 base: 0x4003.9000 | | | | | | | | | | | | | | | |
| ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | ASEN3 | ASEN2 | ASEN1 | ASEN0 |
| ADCRIS, type RO, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | INR3 | INR2 | INR1 | INR0 |
| ADCIM, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DCONSS3 | DCONSS2 | DCONSS1 | DCONSS0 |
| | | | | | | | | | | | | MASK3 | MASK2 | MASK1 | MASK0 |
| ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DCINSS3 | DCINSS2 | DCINSS1 | DCINSS0 |
| | | | | | | | | | | | | IN3 | IN2 | IN1 | IN0 |
| ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | OV3 | OV2 | OV1 | OV0 |
| ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| EM3 | | | | EM2 | | | | EM1 | | | | EM0 | | | |
| ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | UV3 | UV2 | UV1 | UV0 |
| ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210 | | | | | | | | | | | | | | | |
| SS3 | | | | SS2 | | | | SS1 | | | | SS0 | | | |
| ADCPSSI, type WO, offset 0x028, reset - | | | | | | | | | | | | | | | |
| GSYNC | | | | SYNCWAIT | | | | | | | | | | | |
| | | | | | | | | | | | | SS3 | SS2 | SS1 | SS0 |
| ADCSAC, type R/W, offset 0x030, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | AVG | | | |
| ADCDCISC, type R/W1C, offset 0x034, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | DCINT7 | DCINT6 | DCINT5 | DCINT4 | DCINT3 | DCINT2 | DCINT1 | DCINT0 |
| ADCCTL, type R/W, offset 0x038, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | VREF | | | |
| ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| MUX7 | | | | MUX6 | | | | MUX5 | | | | MUX4 | | | |
| MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| 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 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|--|-----|------|----|---------|-----|------|----|---------|-----|------|----|---------|-----|------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCSSFIFO1, type RO, offset 0x088, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| ADCSSFIFO2, type RO, offset 0x088, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| ADCSSFIFO3, type RO, offset 0x0A8, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100 | | | | | | | | | | | | | | | |
| FULL | | | | EMPTY | | | | HPTR | | | | TPTR | | | |
| ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100 | | | | | | | | | | | | | | | |
| FULL | | | | EMPTY | | | | HPTR | | | | TPTR | | | |
| ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100 | | | | | | | | | | | | | | | |
| FULL | | | | EMPTY | | | | HPTR | | | | TPTR | | | |
| ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100 | | | | | | | | | | | | | | | |
| FULL | | | | EMPTY | | | | HPTR | | | | TPTR | | | |
| ADCSSOP0, type R/W, offset 0x050, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| S7DCOP | | | | S6DCOP | | | | S5DCOP | | | | S4DCOP | | | |
| S3DCOP | | | | S2DCOP | | | | S1DCOP | | | | S0DCOP | | | |
| ADCSSDC0, type R/W, offset 0x054, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| S7DCSEL | | | | S6DCSEL | | | | S5DCSEL | | | | S4DCSEL | | | |
| S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |
| ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| MUX3 | | | | MUX2 | | | | MUX1 | | | | MUX0 | | | |
| ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| 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 | | | | | | | | | | | | | | | |
| TS3 | IE3 | END3 | D3 | TS2 | IE2 | END2 | D2 | TS1 | IE1 | END1 | D1 | TS0 | IE0 | END0 | D0 |
| ADCSSOP1, type R/W, offset 0x070, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| S3DCOP | | | | S2DCOP | | | | S1DCOP | | | | S0DCOP | | | |
| ADCSSOP2, type R/W, offset 0x090, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| S3DCOP | | | | S2DCOP | | | | S1DCOP | | | | S0DCOP | | | |
| ADCSSDC1, type R/W, offset 0x074, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |
| ADCSSDC2, type R/W, offset 0x094, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| S3DCSEL | | | | S2DCSEL | | | | S1DCSEL | | | | S0DCSEL | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|---|----|----|----|-----|-----|-----|----|---------|---------|---------|---------|---------|---------|---------|---------|--|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| ADCSSMUX3, type R/W, offset 0x0A0, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | MUX0 | | | | |
| ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TS0 | IE0 | END0 | D0 | |
| ADCSSOP3, type R/W, offset 0x0B0, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | S0DCOP | |
| ADCSSDC3, type R/W, offset 0x0B4, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | S0DCSEL | | | | |
| ADCDCRIC, type R/W, offset 0xD00, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | DCTRIG7 | DCTRIG6 | DCTRIG5 | DCTRIG4 | DCTRIG3 | DCTRIG2 | DCTRIG1 | DCTRIG0 | |
| | | | | | | | | DCINT7 | DCINT6 | DCINT5 | DCINT4 | DCINT3 | DCINT2 | DCINT1 | DCINT0 | |
| ADCDCCTL0, type R/W, offset 0xE00, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCCTL1, type R/W, offset 0xE04, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCCTL2, type R/W, offset 0xE08, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCCTL3, type R/W, offset 0xE0C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCCTL4, type R/W, offset 0xE10, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCCTL5, type R/W, offset 0xE14, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCCTL6, type R/W, offset 0xE18, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCCTL7, type R/W, offset 0xE1C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | CTE | CTC | CTM | | | | | CIE | CIC | CIM | | | |
| ADCDCOMP0, type R/W, offset 0xE40, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | |
| | | | | | | | | | | | | COMP0 | | | | |
| ADCDCOMP1, type R/W, offset 0xE44, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | |
| | | | | | | | | | | | | COMP0 | | | | |
| ADCDCOMP2, type R/W, offset 0xE48, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | |
| | | | | | | | | | | | | COMP0 | | | | |
| ADCDCOMP3, type R/W, offset 0xE4C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | |
| | | | | | | | | | | | | COMP0 | | | | |

| | | | | | | | | | | | | | | | | | | | | |
|---|---------|---------|----|-----|-------|-------|-------|-------|-------|-------|-------|----------|--------|--------|--------|----------|------|-----|-----|-----|
| 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 | | | | | |
| ADCDCCMP4, type R/W, offset 0xE50, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | | | | | |
| | | | | | | | | | | | | COMP0 | | | | | | | | |
| ADCDCCMP5, type R/W, offset 0xE54, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | | | | | |
| | | | | | | | | | | | | COMP0 | | | | | | | | |
| ADCDCCMP6, type R/W, offset 0xE58, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | | | | | |
| | | | | | | | | | | | | COMP0 | | | | | | | | |
| ADCDCCMP7, type R/W, offset 0xE5C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COMP1 | | | | | | | | |
| | | | | | | | | | | | | COMP0 | | | | | | | | |
| 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 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | OE | BE | PE | FE | DATA | | | | |
| UARTSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register) | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | OE | BE | PE | FE | |
| UARTSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Error Clear Register) | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | DATA | | | | |
| UARTFR, type RO, offset 0x018, reset 0x0000.0090 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | RI | TXFE | RXFF | TXFF | RXFE | BUSY | DCD | DSR | CTS |
| UARTILPR, type R/W, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | ILPDVSR | | | | |
| UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | DIVINT | | | | |
| UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | DIVFRAC | | | | |
| UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPS | WLEN | | FEN | STP2 | EPS | PEN | BRK | |
| UARTCTL, type R/W, offset 0x030, reset 0x0000.0300 | | | | | | | | | | | | | | | | | | | | |
| CTSEN | RTSEN | | | RTS | DTR | RXE | TXE | LBE | LIN | HSE | EOT | SMART | SIRLP | SIREN | UARTEN | | | | | |
| UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012 | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | RXIFLSEL | | | | TXIFLSEL | | | | |
| UARTIM, type R/W, offset 0x038, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | | | |
| LME5IM | LME1IM | LMSBIM | | | OEIM | BEIM | PEIM | FEIM | RTIM | TXIM | RXIM | DSRIM | DCDIM | CTSIM | RIIM | | | | | |
| UARTRIS, type RO, offset 0x03C, reset 0x0000.000F | | | | | | | | | | | | | | | | | | | | |
| LME5RIS | LME1RIS | LMSBRIS | | | OERIS | BERIS | PERIS | FERIS | RTRIS | TXRIS | RXRIS | DSRRIS | DCDRIS | CTSRIS | RIRIS | | | | | |

| | | | | | | | | | | | | | | | |
|--|---------|---------|----|----|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|
| 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 |
| UARTMIS, type RO, offset 0x040, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| LME5MIS | LME1MIS | LMSBMIS | | | OEMIS | BEMIS | PEMIS | FEMIS | RTMIS | TXMIS | RXMIS | DSRMIS | DCDMIS | CTSMIS | RIMIS |
| UARTICR, type W1C, offset 0x044, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| LME5MIC | LME1MIC | LMSBMIC | | | OEIC | BEIC | PEIC | FEIC | RTIC | TXIC | RXIC | DSRMIC | DCDMIC | CTSMIC | RIMIC |
| UARTDMACTL, type R/W, offset 0x048, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | DMAERR | TXDMAE | RXDMAE |
| UARTLCTL, type R/W, offset 0x090, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | BLER | | | | | MASTER |
| UARTLSS, type RO, offset 0x094, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| TSS | | | | | | | | | | | | | | | |
| UARTLTIM, type RO, offset 0x098, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| TIMER | | | | | | | | | | | | | | | |
| UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID4 |
| UARTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID5 |
| UARTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID6 |
| UARTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID7 |
| UARTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0060 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID0 |
| UARTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID1 |
| UARTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID2 |
| UARTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PID3 |
| UARTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | CID0 |
| UARTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | CID1 |
| UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | 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 |
| UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID3 | | | |
| Synchronous Serial Interface (SSI) | | | | | | | | | | | | | | | |
| SSI0 base: 0x4000.8000 | | | | | | | | | | | | | | | |
| SSI1 base: 0x4000.9000 | | | | | | | | | | | | | | | |
| SSICR0, type R/W, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| SCR | | | | | | SPH | | SPO | | FRF | | DSS | | | |
| SSICR1, type R/W, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | EOT | | SOD | MS | SSE | LBM |
| SSIDR, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| SSISR, type RO, offset 0x00C, reset 0x0000.0003 | | | | | | | | | | | | | | | |
| | | | | | | | | | | BSY | RFF | RNE | TNF | TFE | |
| SSICPSR, type R/W, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| CPSDVSR | | | | | | | | | | | | | | | |
| SSIIM, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | TXIM | RXIM | RTIM | RORIM | | |
| SSIRIS, type RO, offset 0x018, reset 0x0000.0008 | | | | | | | | | | | | | | | |
| | | | | | | | | | | TXRIS | RXRIS | RTRIS | RORRIS | | |
| SSIMIS, type RO, offset 0x01C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | TXMIS | RXMIS | RTMIS | RORMIS | | |
| SSIICR, type W1C, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | RTIC | | RORIC | |
| SSIDMACTL, type R/W, offset 0x024, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXDMAE | | RXDMAE | |
| SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID4 | | | |
| SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID5 | | | |
| SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID6 | | | |
| SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID7 | | | |
| SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID0 | | | |

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|--------|------|--------|---------|--------|-------|-------|-----|
| 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 |
| SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID1 | | | |
| SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID2 | | | |
| SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PID3 | | | |
| SSIPCellID0, type RO, offset 0xFF0, reset 0x0000.000D | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID0 | | | |
| SSIPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID1 | | | |
| SSIPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID2 | | | |
| SSIPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CID3 | | | |
| Inter-Integrated Circuit (I²C) Interface | | | | | | | | | | | | | | | |
| I²C Master | | | | | | | | | | | | | | | |
| I2C Master 0 base: 0x4002.0000 | | | | | | | | | | | | | | | |
| I2C Master 1 base: 0x4002.1000 | | | | | | | | | | | | | | | |
| I2CMSA, type R/W, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SA | | R/S | |
| I2CMCS, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register) | | | | | | | | | | | | | | | |
| | | | | | | | | BUSBSY | IDLE | ARBLST | DATAACK | ADRACK | ERROR | BUSY | |
| I2CMCS, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Control Register) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | ACK | STOP | START | RUN |
| I2CMDR, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DATA | | | |
| I2CMTPR, type R/W, offset 0x00C, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TPR | | | |
| I2CMIMR, type R/W, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | IM | |
| I2CMRIS, type RO, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | RIS | |
| I2CMMIS, type RO, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | MIS | |
| I2CMICR, type WO, offset 0x01C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | IC | |

| | | | | | | | | | | | | | | | | | | |
|--|----|----|----|-----|-----|-----|-----|------|-----|-----|----|---------|----|----------|----|---------|--|------|
| 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 | | | |
| I2CMCR, type R/W, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SFE | | MFE | | | | LPBK |
| Inter-Integrated Circuit (I²C) Interface | | | | | | | | | | | | | | | | | | |
| I²C Slave | | | | | | | | | | | | | | | | | | |
| I2C Slave 0 base: 0x4002.0800 | | | | | | | | | | | | | | | | | | |
| I2C Slave 1 base: 0x4002.1800 | | | | | | | | | | | | | | | | | | |
| I2CSOAR, type R/W, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | OAR | | | | | | |
| I2CSCSR, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register) | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | FBR | | TREQ | | RREQ | | |
| I2CSCSR, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Control Register) | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DA | | | | | | |
| I2CSDR, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | DATA | | | | | | |
| I2CSIMR, type R/W, offset 0x00C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | STOPIM | | STARTIM | | DATAIM | | |
| I2CSRIS, type RO, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | STOPRIS | | STARTRIS | | DATARIS | | |
| I2CSMIS, type RO, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | STOPMIS | | STARTMIS | | DATAMIS | | |
| I2CSICR, type WO, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | STOPIC | | STARTIC | | DATAIC | | |
| Inter-Integrated Circuit Sound (I²S) Interface | | | | | | | | | | | | | | | | | | |
| Base 0x4005.4000 | | | | | | | | | | | | | | | | | | |
| I2STXFIFO, type WO, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| TXFIFO | | | | | | | | | | | | | | | | | | |
| TXFIFO | | | | | | | | | | | | | | | | | | |
| I2STXFIFOCFG, type R/W, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | CSS | | LRS | | | | |
| I2STXCFG, type R/W, offset 0x008, reset 0x1400.7DF0 | | | | | | | | | | | | | | | | | | |
| | | | | JST | DLY | SCP | LRP | WM | FMT | MSL | | | | | | | | |
| SSZ | | | | | | | | SDSZ | | | | | | | | | | |
| I2STXLIMIT, type R/W, offset 0x00C, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | LIMIT | | | | | | |
| I2STXISM, type R/W, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | FFI | | FFM | | | | |
| I2STXLEV, type RO, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | LEVEL | | | | | | |

| | | | | | | | | | | | | | | | |
|--|----|----|----|-------|----|----|----|------|----|--------------------|----|--------------------|----|----|----|
| 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 |
| I2SRXFIFO, type RO, offset 0x800, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| RXFIFO | | | | | | | | | | | | | | | |
| RXFIFO | | | | | | | | | | | | | | | |
| I2SRXFIFOCFG, type R/W, offset 0x804, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| FMM CSS LRS | | | | | | | | | | | | | | | |
| I2SRXCFG, type R/W, offset 0x808, reset 0x1400.7DF0 | | | | | | | | | | | | | | | |
| JST DLY SCP LRP | | | | RM | | | | MSL | | | | | | | |
| SSZ | | | | | | | | SDSZ | | | | | | | |
| I2SRXLIMIT, type R/W, offset 0x80C, reset 0x0000.7FFF | | | | | | | | | | | | | | | |
| LIMIT | | | | | | | | | | | | | | | |
| I2SRXISM, type R/W, offset 0x810, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| FFI FFM | | | | | | | | | | | | | | | |
| I2SRXLEV, type RO, offset 0x818, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| LEVEL | | | | | | | | | | | | | | | |
| I2SCFG, type R/W, offset 0xC00, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | RXSLV TXSLV | | | | | |
| | | | | | | | | | | | | RXEN TXEN | | | |
| I2SIM, type R/W, offset 0xC10, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | RXREIM RXSRIM | | | | | |
| | | | | | | | | | | | | TXWEIM TXSRIM | | | |
| I2SRIS, type RO, offset 0xC14, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | RXRERIS RXSRRIS | | | | | |
| | | | | | | | | | | | | TXWERIS TXSRRIS | | | |
| I2SMIS, type RO, offset 0xC18, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | RXREMIS RXSRMIS | | | | | |
| | | | | | | | | | | | | TXWEMIS TXSRMIS | | | |
| I2SIC, type WO, offset 0xC1C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | RXREIC | | | | | |
| | | | | | | | | | | | | TXWEIC | | | |
| Controller Area Network (CAN) Module | | | | | | | | | | | | | | | |
| CAN0 base: 0x4004.0000 | | | | | | | | | | | | | | | |
| CAN1 base: 0x4004.1000 | | | | | | | | | | | | | | | |
| CANCTL, type R/W, offset 0x000, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| TEST CCE DAR EIE SIE IE INIT | | | | | | | | | | | | | | | |
| CANSTS, type R/W, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| BOFF EWARN EPASS RXOK TXOK LEC | | | | | | | | | | | | | | | |
| CANERR, type RO, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| RP | | | | REC | | | | TEC | | | | | | | |
| CANBIT, type R/W, offset 0x00C, reset 0x0000.2301 | | | | | | | | | | | | | | | |
| TSEG2 | | | | TSEG1 | | | | SJW | | | | BRP | | | |
| CANINT, type RO, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| INTID | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|--|--------|--------|--------|-------|------|------|-------|--------|------|-----|---------|-----------|-----------------|-------|-------|
| 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 |
| CANTST, type R/W, offset 0x014, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | RX | TX | | LBACK | SILENT | BASIC | | |
| CANBRPE, type R/W, offset 0x018, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | BRPE | |
| CANIF1CRQ, type R/W, offset 0x020, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| | BUSY | | | | | | | | | | | | | MNUM | |
| CANIF2CRQ, type R/W, offset 0x080, reset 0x0000.0001 | | | | | | | | | | | | | | | |
| | BUSY | | | | | | | | | | | | | MNUM | |
| CANIF1CMSK, type R/W, offset 0x024, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | WRNRD | MASK | ARB | CONTROL | CLRINTPND | NEWDAT / TXRQST | DATAA | DATAB |
| CANIF2CMSK, type R/W, offset 0x084, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | WRNRD | MASK | ARB | CONTROL | CLRINTPND | NEWDAT / TXRQST | DATAA | DATAB |
| CANIF1MSK1, type R/W, offset 0x028, reset 0x0000.FFFF | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | MSK |
| CANIF2MSK1, type R/W, offset 0x088, reset 0x0000.FFFF | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | MSK |
| CANIF1MSK2, type R/W, offset 0x02C, reset 0x0000.FFFF | | | | | | | | | | | | | | | |
| | MXTD | MDIR | | | | | | | | | | | | | MSK |
| CANIF2MSK2, type R/W, offset 0x08C, reset 0x0000.FFFF | | | | | | | | | | | | | | | |
| | MXTD | MDIR | | | | | | | | | | | | | MSK |
| CANIF1ARB1, type R/W, offset 0x030, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | ID |
| CANIF2ARB1, type R/W, offset 0x090, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | ID |
| CANIF1ARB2, type R/W, offset 0x034, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | MSGVAL | XTD | DIR | | | | | | | | | | | | ID |
| CANIF2ARB2, type R/W, offset 0x094, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | MSGVAL | XTD | DIR | | | | | | | | | | | | ID |
| CANIF1MCTL, type R/W, offset 0x038, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | NEWDAT | MSGLST | INTPND | UMASK | TXIE | RXIE | RMTEN | TXRQST | EOB | | | | | | DLC |
| CANIF2MCTL, type R/W, offset 0x098, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | NEWDAT | MSGLST | INTPND | UMASK | TXIE | RXIE | RMTEN | TXRQST | EOB | | | | | | DLC |
| CANIF1DA1, type R/W, offset 0x03C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | DATA |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|--|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CANIF1DA2, type R/W, offset 0x040, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| CANIF1DB1, type R/W, offset 0x044, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| CANIF1DB2, type R/W, offset 0x048, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| CANIF2DA1, type R/W, offset 0x09C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| CANIF2DA2, type R/W, offset 0x0A0, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| CANIF2DB1, type R/W, offset 0x0A4, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| CANIF2DB2, type R/W, offset 0x0A8, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| DATA | | | | | | | | | | | | | | | |
| CANTXRQ1, type RO, offset 0x100, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| TXRQST | | | | | | | | | | | | | | | |
| CANTXRQ2, type RO, offset 0x104, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| TXRQST | | | | | | | | | | | | | | | |
| CANNWDA1, type RO, offset 0x120, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| NEWDAT | | | | | | | | | | | | | | | |
| CANNWDA2, type RO, offset 0x124, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| NEWDAT | | | | | | | | | | | | | | | |
| CANMSG1INT, type RO, offset 0x140, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| INTPND | | | | | | | | | | | | | | | |
| CANMSG2INT, type RO, offset 0x144, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| INTPND | | | | | | | | | | | | | | | |
| CANMSG1VAL, type RO, offset 0x160, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| MSGVAL | | | | | | | | | | | | | | | |
| CANMSG2VAL, type RO, offset 0x164, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| MSGVAL | | | | | | | | | | | | | | | |
| Ethernet Controller | | | | | | | | | | | | | | | |
| Ethernet MAC (Ethernet Offset) | | | | | | | | | | | | | | | |
| Base 0x4004.8000 | | | | | | | | | | | | | | | |
| MACRIS/MACIACK, type R/W1C, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| PHYINT MDINT RXER FOV TXEMP TXER RXINT | | | | | | | | | | | | | | | |

| 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 |
| 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 |
| 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 |
| MACLED, type R/W, offset 0x040, reset 0x0000.0010 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | LED1 |
| | | | | | | | | | | | | | | | LED0 |
| MDIX, type R/W, offset 0x044, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | EN |
| Ethernet Controller | | | | | | | | | | | | | | | |
| MII Management (Accessed through the MACMCTL register) | | | | | | | | | | | | | | | |
| MR0, type R/W, address 0x00, reset 0x3100 | | | | | | | | | | | | | | | |
| RESET | LOOPBK | SPEEDSL | ANEGEN | PWRDN | ISO | RANEG | DUPLEX | COLT | | | | | | | |

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 |
| MR1, type RO, address 0x01, reset 0x7849 | | | | | | | | | | | | | | | |
| | 100X_F | 100X_H | 10T_F | 10T_H | | | | | | ANEGC | RFAULT | ANEGA | LINK | JAB | EXTD |
| MR2, type RO, address 0x02, reset 0x0161 | | | | | | | | | | | | | | | |
| OUI[21:6] | | | | | | | | | | | | | | | |
| MR3, type RO, address 0x03, reset 0xB410 | | | | | | | | | | | | | | | |
| 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 | | | | | | | | S |
| MR6, type RO, address 0x06, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | PDF | LPNPA | | PRX | LPANEGA |
| MR16, type RO, address 0x10, reset 0x0040 | | | | | | | | | | | | | | | |
| SR | | | | | | | | | | | | | | | |
| MR17, type R/W, address 0x11, reset 0x0002 | | | | | | | | | | | | | | | |
| | FASTRIP | EDPD | | LSQE | | | FASTEST | | | | | | FGLS | ENON | |
| MR27, type RO, address 0x1B, reset - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | XPOL | | | | |
| MR29, type RO, address 0x1D, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EONIS | ANCOMPIS | RFLTIS | LDIS | LPACKIS | PDFIS | PRXIS | |
| MR30, type R/W, address 0x1E, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EONIM | ANCOMPIM | RFLTIM | LDIM | LPACKIM | PDFIM | PRXIM | |
| MR31, type R/W, address 0x1F, reset 0x00040 | | | | | | | | | | | | | | | |
| | | | AUTODONE | | | | | | | | | SPEED | | | SCRDIS |
| Universal Serial Bus (USB) Controller | | | | | | | | | | | | | | | |
| Base 0x4005.0000 | | | | | | | | | | | | | | | |
| USBFADDR, type R/W, offset 0x000, reset 0x00 | | | | | | | | | | | | | | | |
| FUNCADDR | | | | | | | | | | | | | | | |
| USBPOWER, type R/W, offset 0x001, reset 0x20 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | RESET | RESUME | SUSPEND | PWRDNPHY |
| USBPOWER, type R/W, offset 0x001, reset 0x20 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | ISOUP | SOFTCONN | | | RESET | RESUME | SUSPEND | PWRDNPHY |
| USBTXIS, type RO, offset 0x002, reset 0x0000 | | | | | | | | | | | | | | | |
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| USBRXIS, type RO, offset 0x004, reset 0x0000 | | | | | | | | | | | | | | | |
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | |
| USBTXIE, type R/W, offset 0x006, reset 0xFFFF | | | | | | | | | | | | | | | |
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |
| USBRXIE, type R/W, offset 0x008, reset 0xFFFF | | | | | | | | | | | | | | | |
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | |
| USBIS, type RO, offset 0x00A, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | VBUSERR | SESREQ | DISCON | CONN | SOF | BABBLE | RESUME | |
| USBIS, type RO, offset 0x00A, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SOF | RESET | RESUME | SUSPEND |
| USBIE, type R/W, offset 0x00B, reset 0x06 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | VBUSERR | SESREQ | DISCON | CONN | SOF | BABBLE | RESUME | |
| USBIE, type R/W, offset 0x00B, reset 0x06 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SOF | RESET | RESUME | SUSPEND |
| USBFRAME, type RO, offset 0x00C, reset 0x0000 | | | | | | | | | | | | | | | |
| Frame | | | | | | | | | | | | | | | |
| USBEPIDX, type R/W, offset 0x00E, reset 0x00 | | | | | | | | | | | | | | | |
| EPIDX | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|--|----|----|----|----|----|----|----|--------|---------|---------|----|----|----|----|----|
| 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 |
| USBTEST, type R/W, offset 0x00F, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | FORCEH | FIFOACC | FORCEFS | | | | | |
| USBTEST, type R/W, offset 0x00F, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | FIFOACC | FORCEFS | | | | | |
| USBFIFO0, type R/W, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO1, type R/W, offset 0x024, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO2, type R/W, offset 0x028, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO3, type R/W, offset 0x02C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO4, type R/W, offset 0x030, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO5, type R/W, offset 0x034, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO6, type R/W, offset 0x038, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO7, type R/W, offset 0x03C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO8, type R/W, offset 0x040, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO9, type R/W, offset 0x044, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO10, type R/W, offset 0x048, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO11, type R/W, offset 0x04C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO12, type R/W, offset 0x050, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO13, type R/W, offset 0x054, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO14, type R/W, offset 0x058, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| USBFIFO15, type R/W, offset 0x05C, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |
| | | | | | | | | EPDATA | | | | | | | |

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 |
| USBDEVCTL, type R/W, offset 0x060, reset 0x80 | | | | | | | | | | | | | | | |
| | | | | | | | | DEV | FSDEV | LSDEV | VBUS | | HOST | HOSTREQ | SESSION |
| USBTXFIFOSZ, type R/W, offset 0x062, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | DPB | | SIZE | | |
| USBRXFIFOSZ, type R/W, offset 0x063, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | DPB | | SIZE | | |
| USBTXFIFOADD, type R/W, offset 0x064, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBRXFIFOADD, type R/W, offset 0x066, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBCONTIM, type R/W, offset 0x07A, reset 0x5C | | | | | | | | | | | | | | | |
| | | | | | | | | | | | WTCON | | WTID | | |
| USBVPLEN, type R/W, offset 0x07B, reset 0x3C | | | | | | | | | | | | | | | |
| | | | | | | | | | | | VPLEN | | | | |
| USBFSEOF, type R/W, offset 0x07D, reset 0x77 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | FSEOFG | | | | |
| USBLSEOF, type R/W, offset 0x07E, reset 0x72 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | LSEOFG | | | | |
| USBTXFUNCADDR0, type R/W, offset 0x080, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR1, type R/W, offset 0x088, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR2, type R/W, offset 0x090, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR3, type R/W, offset 0x098, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR4, type R/W, offset 0x0A0, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR5, type R/W, offset 0x0A8, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR6, type R/W, offset 0x0B0, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR7, type R/W, offset 0x0B8, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR8, type R/W, offset 0x0C0, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR9, type R/W, offset 0x0C8, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR10, type R/W, offset 0x0D0, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR11, type R/W, offset 0x0D8, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR12, type R/W, offset 0x0E0, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR13, type R/W, offset 0x0E8, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR14, type R/W, offset 0x0F0, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXFUNCADDR15, type R/W, offset 0x0F8, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | ADDR | | | | |
| USBTXHUBADDR0, type R/W, offset 0x082, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | ADDR | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
|--|----|----|----|----|----|----|----|----------|----|----|----|----|----|----|----|--|------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| USBTXHUBADDR1, type R/W, offset 0x08A, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR2, type R/W, offset 0x092, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR3, type R/W, offset 0x09A, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR4, type R/W, offset 0x0A2, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR5, type R/W, offset 0x0AA, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR6, type R/W, offset 0x0B2, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR7, type R/W, offset 0x0BA, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR8, type R/W, offset 0x0C2, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR9, type R/W, offset 0x0CA, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR10, type R/W, offset 0x0D2, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR11, type R/W, offset 0x0DA, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR12, type R/W, offset 0x0E2, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR13, type R/W, offset 0x0EA, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR14, type R/W, offset 0x0F2, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBADDR15, type R/W, offset 0x0FA, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBTXHUBPORT0, type R/W, offset 0x083, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT1, type R/W, offset 0x08B, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT2, type R/W, offset 0x093, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT3, type R/W, offset 0x09B, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT4, type R/W, offset 0x0A3, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT5, type R/W, offset 0x0AB, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT6, type R/W, offset 0x0B3, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT7, type R/W, offset 0x0BB, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT8, type R/W, offset 0x0C3, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT9, type R/W, offset 0x0CB, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXHUBPORT10, type R/W, offset 0x0D3, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |

| 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 |
| USBTXHUBPORT11, type R/W, offset 0x0DB, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | PORT | | | | | | | |
| USBTXHUBPORT12, type R/W, offset 0x0E3, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | PORT | | | | | | | |
| USBTXHUBPORT13, type R/W, offset 0x0EB, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | PORT | | | | | | | |
| USBTXHUBPORT14, type R/W, offset 0x0F3, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | PORT | | | | | | | |
| USBTXHUBPORT15, type R/W, offset 0x0FB, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | PORT | | | | | | | |
| USBRXFUNCADDR1, type R/W, offset 0x08C, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR2, type R/W, offset 0x094, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR3, type R/W, offset 0x09C, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR4, type R/W, offset 0x0A4, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR5, type R/W, offset 0x0AC, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR6, type R/W, offset 0x0B4, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR7, type R/W, offset 0x0BC, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR8, type R/W, offset 0x0C4, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR9, type R/W, offset 0x0CC, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR10, type R/W, offset 0x0D4, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR11, type R/W, offset 0x0DC, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR12, type R/W, offset 0x0E4, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR13, type R/W, offset 0x0EC, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR14, type R/W, offset 0x0F4, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXFUNCADDR15, type R/W, offset 0x0FC, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | ADDR | | | | | | | |
| USBRXHUBADDR1, type R/W, offset 0x08E, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | ADDR | | | | | |
| USBRXHUBADDR2, type R/W, offset 0x096, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | ADDR | | | | | |
| USBRXHUBADDR3, type R/W, offset 0x09E, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | ADDR | | | | | |
| USBRXHUBADDR4, type R/W, offset 0x0A6, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | ADDR | | | | | |
| USBRXHUBADDR5, type R/W, offset 0x0AE, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | ADDR | | | | | |
| USBRXHUBADDR6, type R/W, offset 0x0B6, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | ADDR | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
|---|----|----|----|----|----|----|----|----------|----|----|----|----|----|----|----|--|---------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| USBXHUBADDR7, type R/W, offset 0x0BE, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR8, type R/W, offset 0x0C6, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR9, type R/W, offset 0x0CE, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR10, type R/W, offset 0x0D6, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR11, type R/W, offset 0x0DE, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR12, type R/W, offset 0x0E6, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR13, type R/W, offset 0x0EE, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR14, type R/W, offset 0x0F6, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBADDR15, type R/W, offset 0x0FE, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | MULTTRAN | | | | | | | | | ADDR |
| USBXHUBPORT1, type R/W, offset 0x08F, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT2, type R/W, offset 0x097, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT3, type R/W, offset 0x09F, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT4, type R/W, offset 0x0A7, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT5, type R/W, offset 0x0AF, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT6, type R/W, offset 0x0B7, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT7, type R/W, offset 0x0BF, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT8, type R/W, offset 0x0C7, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT9, type R/W, offset 0x0CF, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT10, type R/W, offset 0x0D7, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT11, type R/W, offset 0x0DF, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT12, type R/W, offset 0x0E7, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT13, type R/W, offset 0x0EF, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT14, type R/W, offset 0x0F7, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBXHUBPORT15, type R/W, offset 0x0FF, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | PORT |
| USBTXMAXP1, type R/W, offset 0x110, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | MAXLOAD |
| USBTXMAXP2, type R/W, offset 0x120, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | MAXLOAD |

| 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 |
| USBTXMAXP3, type R/W, offset 0x130, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP4, type R/W, offset 0x140, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP5, type R/W, offset 0x150, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP6, type R/W, offset 0x160, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP7, type R/W, offset 0x170, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP8, type R/W, offset 0x180, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP9, type R/W, offset 0x190, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP10, type R/W, offset 0x1A0, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP11, type R/W, offset 0x1B0, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP12, type R/W, offset 0x1C0, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP13, type R/W, offset 0x1D0, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP14, type R/W, offset 0x1E0, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBTXMAXP15, type R/W, offset 0x1F0, reset 0x0000 | | | | | | | | | | | | | | | |
| MAXLOAD | | | | | | | | | | | | | | | |
| USBCSRL0, type W1C, offset 0x102, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| NAKTO STATUS REQPKT ERROR SETUP STALLED TXRDY RXRDY | | | | | | | | | | | | | | | |
| USBCSRL0, type W1C, offset 0x102, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| SETENDC RXRDYC STALL SETEND DATAEND STALLED TXRDY RXRDY | | | | | | | | | | | | | | | |
| USBCSRH0, type W1C, offset 0x103, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| DTWE DT FLUSH | | | | | | | | | | | | | | | |
| USBCSRH0, type W1C, offset 0x103, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| FLUSH | | | | | | | | | | | | | | | |
| USBCOUNT0, type RO, offset 0x108, reset 0x00 | | | | | | | | | | | | | | | |
| COUNT | | | | | | | | | | | | | | | |
| USBTYPE0, type R/W, offset 0x10A, reset 0x00 | | | | | | | | | | | | | | | |
| SPEED | | | | | | | | | | | | | | | |
| USBNAKLMT, type R/W, offset 0x10B, reset 0x00 | | | | | | | | | | | | | | | |
| NAKLMT | | | | | | | | | | | | | | | |
| USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY | | | | | | | | | | | | | | | |
| USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY | | | | | | | | | | | | | | | |
| USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY | | | | | | | | | | | | | | | |
| USBTXCSRL4, type R/W, offset 0x142, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY | | | | | | | | | | | | | | | |
| USBTXCSRL5, type R/W, offset 0x152, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY | | | | | | | | | | | | | | | |
| USBTXCSRL6, type R/W, offset 0x162, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY | | | | | | | | | | | | | | | |

| 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 |
| USBTXCSRL7, type R/W, offset 0x172, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL8, type R/W, offset 0x182, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL9, type R/W, offset 0x192, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL10, type R/W, offset 0x1A2, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL11, type R/W, offset 0x1B2, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL12, type R/W, offset 0x1C2, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL13, type R/W, offset 0x1D2, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL14, type R/W, offset 0x1E2, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL15, type R/W, offset 0x1F2, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | NAKTO | CLRDT | STALLED | SETUP | FLUSH | ERROR | FIFONE | TXRDY |
| USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL4, type R/W, offset 0x142, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL5, type R/W, offset 0x152, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL6, type R/W, offset 0x162, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL7, type R/W, offset 0x172, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL8, type R/W, offset 0x182, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL9, type R/W, offset 0x192, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL10, type R/W, offset 0x1A2, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL11, type R/W, offset 0x1B2, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL12, type R/W, offset 0x1C2, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL13, type R/W, offset 0x1D2, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL14, type R/W, offset 0x1E2, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRL15, type R/W, offset 0x1F2, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | | CLRDT | STALLED | STALL | FLUSH | UNDRN | FIFONE | TXRDY |
| USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |

| 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 |
| USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH4, type R/W, offset 0x143, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH5, type R/W, offset 0x153, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH6, type R/W, offset 0x163, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH7, type R/W, offset 0x173, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH8, type R/W, offset 0x183, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH9, type R/W, offset 0x193, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH10, type R/W, offset 0x1A3, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH11, type R/W, offset 0x1B3, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH12, type R/W, offset 0x1C3, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH13, type R/W, offset 0x1D3, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH14, type R/W, offset 0x1E3, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH15, type R/W, offset 0x1F3, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | | MODE | DMAEN | FDT | DMAMOD | DTWE | DT |
| USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH4, type R/W, offset 0x143, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH5, type R/W, offset 0x153, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH6, type R/W, offset 0x163, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH7, type R/W, offset 0x173, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH8, type R/W, offset 0x183, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH9, type R/W, offset 0x193, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH10, type R/W, offset 0x1A3, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH11, type R/W, offset 0x1B3, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH12, type R/W, offset 0x1C3, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH13, type R/W, offset 0x1D3, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|---------|---------|--------|-------|-----------------|--------|------|-------|
| 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 |
| USBTXCSRH14, type R/W, offset 0x1E3, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBTXCSRH15, type R/W, offset 0x1F3, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOSET | ISO | MODE | DMAEN | FDT | DMAMOD | | |
| USBRXMAXP1, type R/W, offset 0x114, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP2, type R/W, offset 0x124, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP3, type R/W, offset 0x134, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP4, type R/W, offset 0x144, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP5, type R/W, offset 0x154, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP6, type R/W, offset 0x164, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP7, type R/W, offset 0x174, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP8, type R/W, offset 0x184, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP9, type R/W, offset 0x194, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP10, type R/W, offset 0x1A4, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP11, type R/W, offset 0x1B4, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP12, type R/W, offset 0x1C4, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP13, type R/W, offset 0x1D4, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP14, type R/W, offset 0x1E4, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXMAXP15, type R/W, offset 0x1F4, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | MAXLOAD | | | | | | | |
| USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL4, type R/W, offset 0x146, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL5, type R/W, offset 0x156, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL6, type R/W, offset 0x166, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL7, type R/W, offset 0x176, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|-------|---------|--------|-------|-----------------|-------|------|-------|
| 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 |
| USBRXCSRL8, type R/W, offset 0x186, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL9, type R/W, offset 0x196, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL10, type R/W, offset 0x1A6, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL11, type R/W, offset 0x1B6, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL12, type R/W, offset 0x1C6, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL13, type R/W, offset 0x1D6, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL14, type R/W, offset 0x1E6, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL15, type R/W, offset 0x1F6, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | REQPKT | FLUSH | DATAERR / NAKTO | ERROR | FULL | RXRDY |
| USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL4, type R/W, offset 0x146, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL5, type R/W, offset 0x156, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL6, type R/W, offset 0x166, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL7, type R/W, offset 0x176, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL8, type R/W, offset 0x186, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL9, type R/W, offset 0x196, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL10, type R/W, offset 0x1A6, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL11, type R/W, offset 0x1B6, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL12, type R/W, offset 0x1C6, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL13, type R/W, offset 0x1D6, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL14, type R/W, offset 0x1E6, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |
| USBRXCSRL15, type R/W, offset 0x1F6, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | CLRDT | STALLED | STALL | FLUSH | DATAERR | OVER | FULL | RXRDY |

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|--------|--------|-------|------------------|--------|------|----|----|
| 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 |
| USBXCSRH1, type R/W, offset 0x117, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH2, type R/W, offset 0x127, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH3, type R/W, offset 0x137, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH4, type R/W, offset 0x147, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH5, type R/W, offset 0x157, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH6, type R/W, offset 0x167, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH7, type R/W, offset 0x177, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH8, type R/W, offset 0x187, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH9, type R/W, offset 0x197, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH10, type R/W, offset 0x1A7, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH11, type R/W, offset 0x1B7, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH12, type R/W, offset 0x1C7, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH13, type R/W, offset 0x1D7, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH14, type R/W, offset 0x1E7, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH15, type R/W, offset 0x1F7, reset 0x00 (OTG A / Host Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | AUTORQ | DMAEN | PIDERR | DMAMOD | DTWE | DT | |
| USBXCSRH1, type R/W, offset 0x117, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH2, type R/W, offset 0x127, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH3, type R/W, offset 0x137, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH4, type R/W, offset 0x147, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH5, type R/W, offset 0x157, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH6, type R/W, offset 0x167, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH7, type R/W, offset 0x177, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH8, type R/W, offset 0x187, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |

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 |
| USBXCSRH9, type R/W, offset 0x197, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH10, type R/W, offset 0x1A7, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH11, type R/W, offset 0x1B7, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH12, type R/W, offset 0x1C7, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH13, type R/W, offset 0x1D7, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH14, type R/W, offset 0x1E7, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCSRH15, type R/W, offset 0x1F7, reset 0x00 (OTG B / Device Mode) | | | | | | | | | | | | | | | |
| | | | | | | | | AUTOCL | ISO | DMAEN | DISNYET / PIDERR | DMAMOD | | | |
| USBXCOUNT1, type RO, offset 0x118, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT2, type RO, offset 0x128, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT3, type RO, offset 0x138, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT4, type RO, offset 0x148, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT5, type RO, offset 0x158, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT6, type RO, offset 0x168, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT7, type RO, offset 0x178, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT8, type RO, offset 0x188, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT9, type RO, offset 0x198, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT10, type RO, offset 0x1A8, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT11, type RO, offset 0x1B8, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT12, type RO, offset 0x1C8, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT13, type RO, offset 0x1D8, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT14, type RO, offset 0x1E8, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBXCOUNT15, type RO, offset 0x1F8, reset 0x0000 | | | | | | | | | | | | | | | |
| | | | | | | | | COUNT | | | | | | | |
| USBTXTYPE1, type R/W, offset 0x11A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE2, type R/W, offset 0x12A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|-----------------|----|-------|----|-----|----|----|----|
| 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 |
| USBTXTYPE3, type R/W, offset 0x13A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE4, type R/W, offset 0x14A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE5, type R/W, offset 0x15A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE6, type R/W, offset 0x16A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE7, type R/W, offset 0x17A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE8, type R/W, offset 0x18A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE9, type R/W, offset 0x19A, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE10, type R/W, offset 0x1AA, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE11, type R/W, offset 0x1BA, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE12, type R/W, offset 0x1CA, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE13, type R/W, offset 0x1DA, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE14, type R/W, offset 0x1EA, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXTYPE15, type R/W, offset 0x1FA, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | SPEED | | PROTO | | TEP | | | |
| USBTXINTERVAL1, type R/W, offset 0x11B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL2, type R/W, offset 0x12B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL3, type R/W, offset 0x13B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL4, type R/W, offset 0x14B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL5, type R/W, offset 0x15B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL6, type R/W, offset 0x16B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL7, type R/W, offset 0x17B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL8, type R/W, offset 0x18B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL9, type R/W, offset 0x19B, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL10, type R/W, offset 0x1AB, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL11, type R/W, offset 0x1BB, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL12, type R/W, offset 0x1CB, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |
| USBTXINTERVAL13, type R/W, offset 0x1DB, reset 0x00 | | | | | | | | | | | | | | | |
| | | | | | | | | TXPOLL / NAKLMT | | | | | | | |

| | | | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|----|----|----|----|-----------------|----|-------|----|-----|--|
| 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 | | |
| USBTXINTERVAL14, type R/W, offset 0x1EB, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBTXINTERVAL15, type R/W, offset 0x1FB, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXTYPE1, type R/W, offset 0x11C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE2, type R/W, offset 0x12C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE3, type R/W, offset 0x13C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE4, type R/W, offset 0x14C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE5, type R/W, offset 0x15C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE6, type R/W, offset 0x16C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE7, type R/W, offset 0x17C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE8, type R/W, offset 0x18C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE9, type R/W, offset 0x19C, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE10, type R/W, offset 0x1AC, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE11, type R/W, offset 0x1BC, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE12, type R/W, offset 0x1CC, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE13, type R/W, offset 0x1DC, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE14, type R/W, offset 0x1EC, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXTYPE15, type R/W, offset 0x1FC, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | SPEED | | PROTO | | TEP | |
| USBRXINTERVAL1, type R/W, offset 0x11D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL2, type R/W, offset 0x12D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL3, type R/W, offset 0x13D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL4, type R/W, offset 0x14D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL5, type R/W, offset 0x15D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL6, type R/W, offset 0x16D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL7, type R/W, offset 0x17D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL8, type R/W, offset 0x18D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL9, type R/W, offset 0x19D, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |

| | | | | | | | | | | | | | | | | | |
|--|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----------------|---------|---------|--------|--------|------|
| 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 | | |
| USBRXINTERVAL10, type R/W, offset 0x1AD, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL11, type R/W, offset 0x1BD, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL12, type R/W, offset 0x1CD, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL13, type R/W, offset 0x1DD, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL14, type R/W, offset 0x1ED, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRXINTERVAL15, type R/W, offset 0x1FD, reset 0x00 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | TXPOLL / NAKLMT | | | | | |
| USBRQPKTCOUNT1, type R/W, offset 0x304, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT2, type R/W, offset 0x308, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT3, type R/W, offset 0x30C, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT4, type R/W, offset 0x310, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT5, type R/W, offset 0x314, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT6, type R/W, offset 0x318, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT7, type R/W, offset 0x31C, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT8, type R/W, offset 0x320, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT9, type R/W, offset 0x324, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT10, type R/W, offset 0x328, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT11, type R/W, offset 0x32C, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT12, type R/W, offset 0x330, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT13, type R/W, offset 0x334, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT14, type R/W, offset 0x338, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBRQPKTCOUNT15, type R/W, offset 0x33C, reset 0x0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | COUNT | | | | | |
| USBXDPKTBUFDIS, type R/W, offset 0x340, reset 0x0000 | | | | | | | | | | | | | | | | | |
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | | | |
| USBTXDPKTBUFDIS, type R/W, offset 0x342, reset 0x0000 | | | | | | | | | | | | | | | | | |
| EP15 | EP14 | EP13 | EP12 | EP11 | EP10 | EP9 | EP8 | EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | | | |
| USBEP, type R/W, offset 0x400, reset 0x0000.0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | PFLTACT | PFLTAEN | PFLTSEN | PFLTEN | EPENDE | EPEN |
| USBEP, type RO, offset 0x404, reset 0x0000.0000 | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PF | | |

| 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 | |
| USBEPICIM, type R/W, offset 0x408, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PF | |
| USBEPICISC, type R/W, offset 0x40C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | PF | |
| USBDRRIS, type RO, offset 0x410, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | RESUME | |
| USBDRIIM, type R/W, offset 0x414, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | RESUME | |
| USBDRISC, type W1C, offset 0x418, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | RESUME | |
| USBGPCS, type R/W, offset 0x41C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | DEVMODOTG | DEVMOD | | |
| USBVDC, type R/W, offset 0x430, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | VBDEN | |
| USBVDCRIS, type RO, offset 0x434, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | VD | |
| USBVDCIM, type R/W, offset 0x438, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | VD | |
| USBVDCISC, type R/W, offset 0x43C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | VD | |
| USBIDVRIS, type RO, offset 0x444, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | ID | |
| USBIDVIM, type R/W, offset 0x448, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | ID | |
| USBIDVISC, type R/W1C, offset 0x44C, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | ID | |
| USBDMASEL, type R/W, offset 0x450, reset 0x0033.2211 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |
| Analog Comparators | | | | | | | | | | | | | | | | |
| Base 0x4003.C000 | | | | | | | | | | | | | | | | |
| ACMIS, type R/W1C, offset 0x000, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | IN2 | IN1 | IN0 |
| ACRIS, type RO, offset 0x004, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | IN2 | IN1 | IN0 |
| ACINTEN, type R/W, offset 0x008, reset 0x0000.0000 | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | IN2 | IN1 | IN0 |

| | | | | | | | | | | | | | | | |
|--|----|----|----|----|------|-------|----|--------|------|--------|----|------|----|------|------|
| 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 |
| ACREFCTL, type R/W, offset 0x010, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | EN | RNG | | | | | | | VREF |
| ACSTAT0, type RO, offset 0x020, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | OVAL |
| ACSTAT1, type RO, offset 0x040, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | OVAL |
| ACSTAT2, type RO, offset 0x060, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | OVAL |
| ACCTL0, type R/W, offset 0x024, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | TOEN | ASRCP | | TSLVAL | TSEN | ISLVAL | | ISEN | | CINV | |
| ACCTL1, type R/W, offset 0x044, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | TOEN | ASRCP | | TSLVAL | TSEN | ISLVAL | | ISEN | | CINV | |
| ACCTL2, type R/W, offset 0x064, reset 0x0000.0000 | | | | | | | | | | | | | | | |
| | | | | | TOEN | ASRCP | | TSLVAL | TSEN | ISLVAL | | ISEN | | CINV | |

E Ordering and Contact Information

E.1 Ordering Information

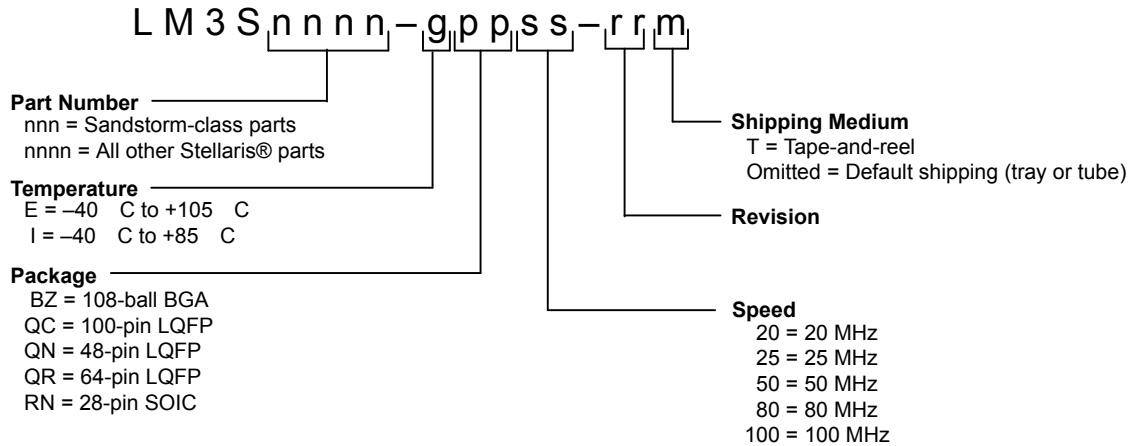


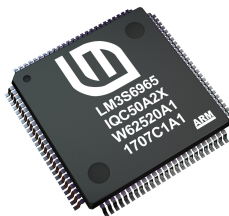
Table E-1. Part Ordering Information

| Orderable Part Number | Description |
|-----------------------|-------------------------------------|
| LM3S9B90-IQC80-B1 | Stellaris® LM3S9B90 Microcontroller |
| LM3S9B90-IQC80-B1T | Stellaris® LM3S9B90 Microcontroller |

E.2 Part Markings

The Stellaris® microcontrollers are marked with an identifying number. This code contains the following information:

- The first line indicates the part number. In the example figure below, this is the LM3S6965.
- The first seven characters in the second line indicate the temperature, package, speed, and revision. In the example figure below, this is an Industrial temperature (I), 100-pin LQFP package (QC), 50-MHz (50), revision A2 (A2) device.
- The remaining characters contain internal tracking numbers.



E.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:
http://www.luminarymicro.com/products/reference_design_kits/
- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris® microcontrollers before purchase:
<http://www.luminarymicro.com/products/kits.html>
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box:
http://www.luminarymicro.com/products/development_kits.html

See the website for the latest tools available, or ask your distributor.

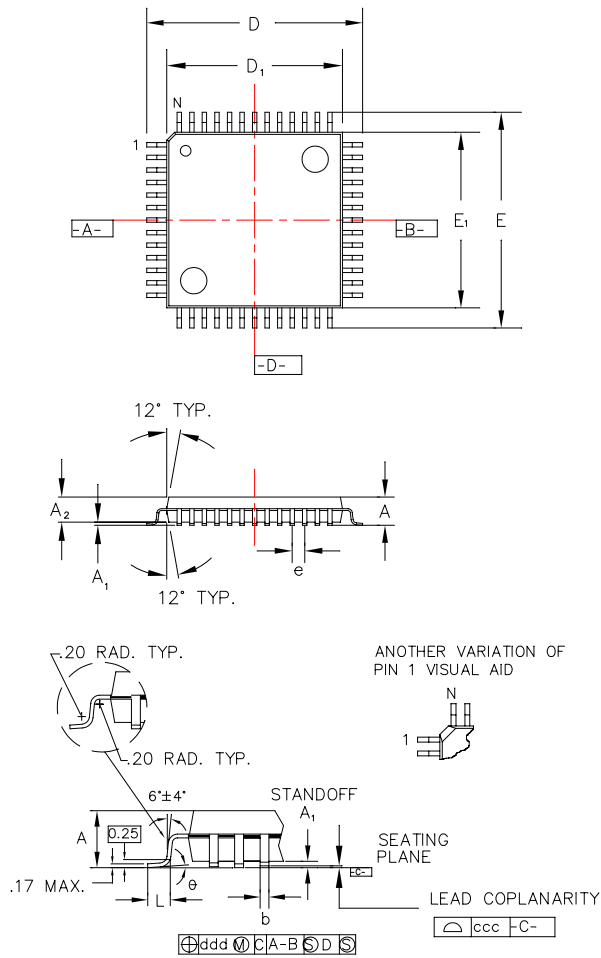
E.4 Support Information

For support on Stellaris® products, contact:

support_lmi@ti.com +1-512-279-8800, ext. 3

F Package Information

Figure F-1. 100-Pin LQFP Package



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 |

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products

| | |
|-----------------------------|--|
| Amplifiers | amplifier.ti.com |
| Data Converters | dataconverter.ti.com |
| DLP® Products | www.dlp.com |
| DSP | dsp.ti.com |
| Clocks and Timers | www.ti.com/clocks |
| Interface | interface.ti.com |
| Logic | logic.ti.com |
| Power Mgmt | power.ti.com |
| Microcontrollers | microcontroller.ti.com |
| RFID | www.ti-rfid.com |
| RF/IF and ZigBee® Solutions | www.ti.com/lprf |

Applications

| | |
|--------------------|--|
| Audio | www.ti.com/audio |
| Automotive | www.ti.com/automotive |
| Broadband | www.ti.com/broadband |
| Digital Control | www.ti.com/digitalcontrol |
| Medical | www.ti.com/medical |
| Military | www.ti.com/military |
| Optical Networking | www.ti.com/opticalnetwork |
| Security | www.ti.com/security |
| Telephony | www.ti.com/telephony |
| Video & Imaging | www.ti.com/video |
| Wireless | www.ti.com/wireless |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2009, Texas Instruments Incorporated